HP Service Manager

Software Version: Service Manager 9.40; Universal CMDB 10.20 or later

For the supported Windows® and Unix® operating systems

Universal CMDB Integration Guide (Using Service Manager Enhanced Generic Adapter)

Document Release Date: March 2015

Software Release Date: January 2015

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® is a trademark of Adobe Systems Incorporated.

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

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

UNIX® is a registered trademark of The Open Group.

Linux® is the registered trademark of Linus Torvalds in the U.S. and other countries.

For a complete list of open source and third party acknowledgements, visit the HP Software Support Online web site and search for the product manual called HP Service Manager Open Source and Third Party License Agreements.

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: **https://softwaresupport.hp.com**

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 website at: **https://softwaresupport.hp.com**

This website 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 website 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 website. 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 website is **http://h20230.www2.hp.com/sc/solutions/index.jsp**

Contents

Chapter 1: Introduction

This chapter provides an overview of the HP Universal CMDB (UCMDB) - HP Service Manager (SM) integration (also referred to as the Universal CMDB (UCMDB) integration or UCMDB-SM integration throughout this document).

This chapter includes:

Who Should Read this Guide

This guide is intended for a system implementer or system administrator who will be establishing and maintaining a connection between the HP Universal CMDB (UCMDB) and Service Manager (SM) systems. This guide assumes that you have administrative access to both systems. The procedures in this guide may duplicate information that is available in your UCMDB and Service Manager help systems, but is provided here for your convenience.

Note: This document provides instructions on setting up an integration between the two products by using the Service Manager Enhanced Generic Adapter, which is available only in UCMDB 10.20 or later. Service Manger 9.40 can still work with the old XSLT-based Service Manager 9-x adapter. For information about setting up the integration using this XSLT-based adapter, see the *UCMDB-SM Integration Guide (Using Service Manager 9-x Adapter)*.

Purpose of the Integration

An integration between HP Universal CMDB (UCMDB) and HP Service Manager enables you to share information about the actual state of a configuration item (CI) between your UCMDB system and a Service Manager system. CIs commonly include IT services, hardware and software. Any organization that wants to implement the best practices Configuration Management and Change Management ITIL processes can use this integration to verify that CIs actually have the attribute values the organization has agreed to support.

You can use this integration to automate the creation of Service Manager change or incident records to update or rollback CIs that have unexpected attribute values. Service Manager allows you to programmatically define what actions you want to take whenever a CI's actual state does not match the expected state as defined in the CI record.

The integration offers several different ways for users to view CI actual state information:

- By default, the integration automatically updates the managed fields of Service Manager CI records as part of the regular UCMDB synchronization schedule. You can choose the option to configure the integration to automatically create change or incident records instead.
- A Service Manager user can view the current actual state of a CI by looking at the Actual State section in the CI record. When you open the Actual State section, Service Manager makes a web services request to UCMDB and displays all CI attributes the request returns. Service Manager only makes the web service call when you open this section.
- A Service Manager user can use the **View in UCMDB** option to log in to the UCMDB system and view the current CI attributes from UCMDB. The Service Manager user must have a valid UCMDB user name and password to log in to the UCMDB system.

Supported Use Cases

This section describes use cases that are supported by the UCMDB-SM integration. The supported use cases provide the core business processes that are enabled by the UCMDB-SM integration.

There are four main business use cases supported by the UCMDB-SM integration. They are as follows:

- Planned Change: A change created in SM through the formal SM change process.
- Unplanned Change: A change or incident that occurred in SM and does not conform to the formal SM change process.
- Retrieving SM Ticket Information: The ability to view SM ticket information in UCMDB.
- Actual State: The ability to view the UCMDB CI information in SM.

All of the use cases provide important functionalities that enable the user to perform ITIL (IT Infrastructure Library) processes. The ITIL processes refer to a set of best practices that define and outline how organizations should manage their IT.

Enabling ITIL Processes

By activating CI push from UCMDB to SM the user facilitates ITIL processes such as Incident, Problem and Change Management in SM.

SM utilizes the data pushed from UCMDB in the following modules:

- Incident Management: the Service Desk operator (SD Agent) selects the “Service” and the “Affected CI” for the specific Incident record.
- Problem Management: the SD agent selects the “Service”, “Affected CI(s)”, and the “Primary CI” for the specific Problem record.
- Change Management: the SD agent selects the “Service” and the “Affected CI(s)” for the specific Change record.

In each of the previously mentioned ITIL processes, SM utilizes CI information for Service, Affected CIs and Primary CIs that all originate in UCMDB.

Managing Planned Changes

The purpose of the “Planned Change” use case is to provide IT organizations a formal process by which changes to the IT infrastructure are introduced after thorough review and analysis. This is performed according to the “Change Management” process defined in ITIL.

A “Planned Change” is initiated by the SM user through the formal “Change Management” process module in SM. This is followed by the actual change implementation.

The actual changes are discovered by a discovery tool such as HP DDMA, and then updated in UCMDB and the relevant modifications are pushed to SM. Once the user has validated the change, the user closes the relevant planned change in SM.

Managing Unplanned Changes

The purpose of the “Unplanned Change” use case is to provide IT organizations a formal process by which all changes that occur to the IT infrastructure are both logged and conventionalized through the organizations formal approval process.

An “Unplanned Change” is a change that is recognized by a Discovery tool such as DDMA. The change is first updated and visible in UCMDB and then the data is pushed to SM. SM recognizes the change and as a result an “Incident” or “Change” record is generated.

These Changes are seen also in the SM “Pending Changes” section in the Configuration Item form, once approved they are moved to the SM “Historic Changes” section.

Retrieving Service Manager Ticket Information

Retrieving SM ticket information from within UCMDB provides all HP Software applications users with access to this information by using UCMDB's federation capabilities and supporting APIs. These applications include Business Service Management (BSM), Asset Manager (AM), Operations Orchestration (OO), etc.

SM ticket data is accessed from within UCMDB using UCMDB federation capabilities. SM ticket data includes Incident, Problem and Change records as well as a key set of their attributes.

UCMDB enables users to create reports/views that combine the federated ticket data from SM with CI information from UCMDB.

Retrieving Actual State of UCMDB CIs

The purpose of “Actual State” is to enable SM users insight into CIs’ current state as detected by “Discovery Tools” and populated in UCMDB. This state provides up-to-date information that may vary from the information displayed in SM both in content and in scope.

The “Actual State” of the CI is displayed in SM in order to enable the user to validate the current state of the CI that resides in UCMDB or in another data repository.

SM users retrieve the Actual State of CIs from UCMDB or additional data sources by viewing the CI's Actual State section in the SM Configuration Item form.

Accessing UCMDB CIs from Service Manager

SM users can open the UCMDB User Interface in the context of a specific CI, by clicking the **View in UCMDB** button in the SM CI record. When the user clicks the **View in UCMDB** button, a UCMDB login screen is displayed; After the user enters a UCMDB username and password, UCMDB displays a topological view of the specific CI together with all related CIs that are linked to it.

Tip: You can configure Lightweight Single Sign-On (LW-SSO) for the integration, so that Service Manager web client users can bypass the UCMDB login screen after clicking the **View in UCMDB** button. For more information, see "How to Enable Lightweight Single Sign-On (LW-SSO) Configuration" on page 102.

If the UCMDB Browser URL is specified in the SM System Information Record, this button is replaced by the **View in UCMDB Browser** button. When you click the **View in UCMDB Browser** button, a UCMDB

Browser login screen is displayed; After you enter a UCMDB Browser username and password, the CI is displayed in the UCMDB Browser user interface.

Core Features

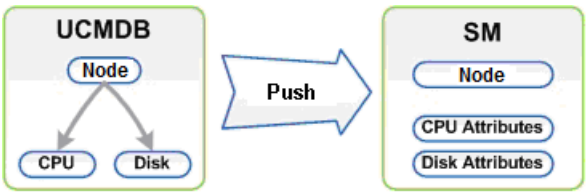
This section explains the rudimentary concepts behind the Federation, Push, and Population features as they pertain to the integration.

This section includes:

Push

UCMDB can automatically discover most types of CIs available in Service Manager. This integration enables you to push these types of CIs from UCMDB to Service Manager.

The following figure shows how data is pushed from UCMDB to Service Manager (SM). The data is physically pushed (copied) from UCMDB to SM. Once the data is physically located in SM, the data is utilized by the SM user that consumes this information in various SM processes.



Note: CI Type and Attribute Push

Only information that is physically present in UCMDB can be pushed to SM.

Federation

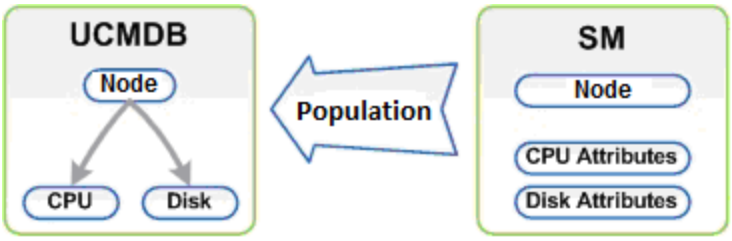
With the federation feature, UCMDB pulls various ticket information (for example, Incident, Problem, and Change ticket information) from SM. This enables users to see Ticket information in UCMDB as Ticket CIs that are connected to the relevant Nodes.

When data is federated (reflected or mirrored) from SM to UCMDB, the data is not physically present in UCMDB; instead it is passed over to UCMDB through Web Services.

Population

You can also use this integration to populate those types of CIs that UCMDB cannot automatically discover or CIs that have been created in Service Manager before you have a UCMDB system deployed. For more information, see "When Do I Need the Population Feature?" on page 106.

Population is the reverse of Push. The following figure shows how data is populated from SM to UCMDB. One SM CI record with multiple attributes is transferred to UCMDB as multiple CI records.



How CI information is Synchronized Between UCMDB and Service Manager

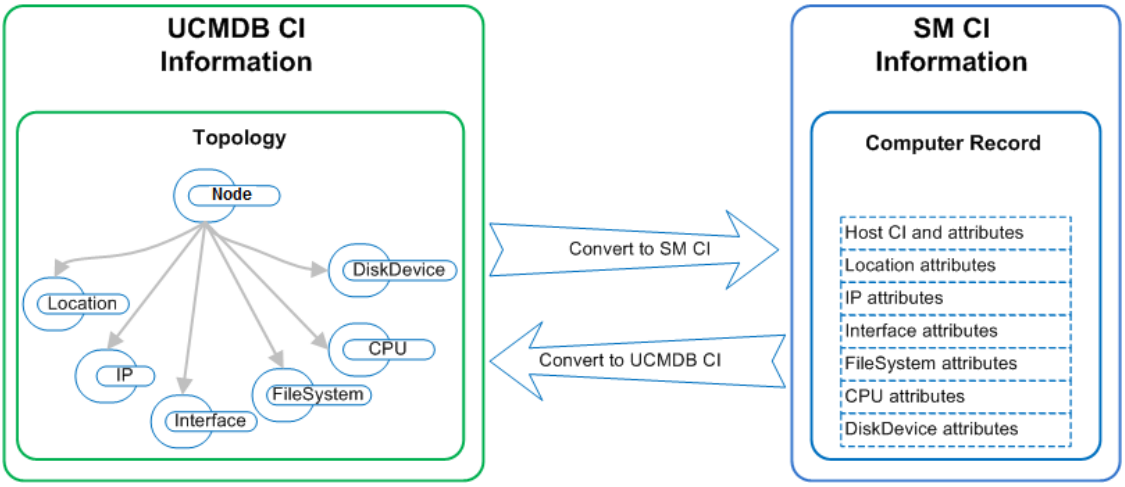
This section explains how CI information is transferred between the UCMDB and Service Manager systems.

This section includes:

CI Information Usage

When referring to the concept of CI information it is important to make the distinction between a UCMDB CI and a Service Manager (SM) CI. The UCMDB model represents a topology that contains a number of CI types and relationships.

The UCMDB topology can be represented in Service Manager as a single entity. Multiple CIs from UCMDB and their attributes are merged into a single record in SM and the relevant UCMDB attributes are mapped to their appropriate counterparts in the SM record.



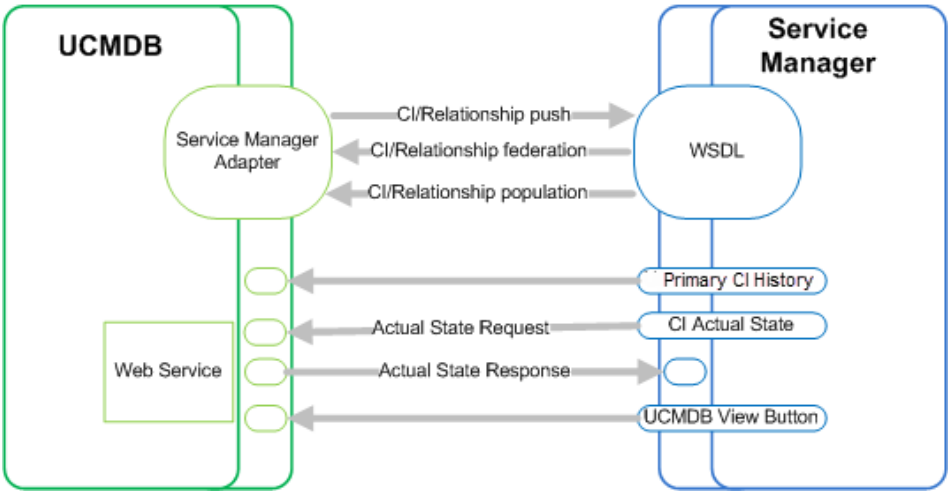
The above figure shows the correlation between the UCMDB topological model and its representation of the Computer Instance together with its parallel representation in SM. The SM computer CI contains all of the UCMDB information that is passed through the integration.

In the push flow, in the UCMDB topological view several CIs such as Node, IP, Interface, Location, File System, CPU, Disk Device and their Relationships are converted into a single SM computer record with the IP, MAC Address and Location, File System, CPU and Disk Device attributes.

In the population flow, the conversion is reversed.

High-Level Components of the Integration

The following diagram shows the high-level components of the integration, and illustrates the interactions between UCMDB and Service Manager.

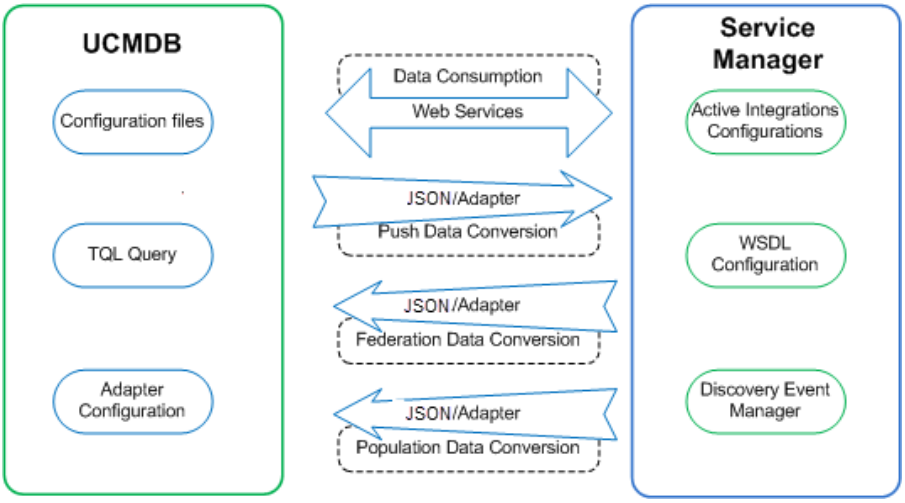


Relationships Between Integration Components

The following figure illustrates the relationships between the Service Manager Adapter components in UCMDB and the associated components in Service Manager.

The Service Manager Adapter includes configuration files, which are used to map UCMDB entities to their counterparts in Service Manager during data push, as well as map Service Manager CIs to UCMDB entities during population.

The configuration files utilize UCMDB queries that define a superset of data relevant for the integration.



What Information Is Stored in UCMDB

Your UCMDB system stores the actual state of CIs and CI relationships as CI attributes. Typically, UCMDB uses one or more integrations and discovery mechanisms (feeders) to automatically detect CI attribute values. The UCMDB-SM integration only uses a subset of the CI attributes available in a UCMDB system.

For more information, see "Tailoring the Integration" on page 113.

What Information Is Stored in Service Manager

Your Service Manager system stores the managed or expected state of CIs and CI relationships as attribute values in a CI record. To be part of the integration, a CI attribute in your UCMDB system must map to a managed field in the Service Manager CI record. You can add, remove, or update the managed fields that are part of the integration by tailoring the Service Manager web services that manage the integration.

Service Manager runs according to a set of rules that define what actions you want the system to take whenever a CI's actual state does not match the expected state as defined in the CI record. You define these rules from the Discovery Event Manager (DEM) in Service Manager where you can do the following:

- Automatically update a CI record to match the attribute values listed in the actual state. (This is the default behavior.)
- Automatically create a change record to review the differences between the actual state and the managed state.
- Automatically create an incident record to review the differences between the actual state and the managed state.

Chapter 2: Integration Setup

Before implementing the integration in your production environment, you can set up the integration in a test environment using the out-of-the-box integration configurations. This chapter describes the basic integration setup tasks without any tailoring or multi-tenancy configurations. It covers the following topics:

Tip: Before you proceed to implementing the integration in your production environment, you can refer to the following chapters for further information:

- "Multi-Tenancy (Multi-Company) Setup" on page 68, which describes how you set up the integration in multi-tenancy mode.
- "Standards and Best Practices" on page 89, which describes best practices for implementing the integration and also provides Frequently-Asked-Questions information.
- "Tailoring the Integration" on page 113, which describes how you can tailor the integration to better suit your business needs.
- "Troubleshooting" on page 189, which provides information on troubleshooting data push and population issues.

Integration Requirements

The supported product versions of this integration are listed in the following table.

Supported product versions

Service Manager	UCMDB
9.40	10.20 or later

Note: This document describes the integration based on the Service Manager Enhanced Generic Adapter, which was introduced in UCMDB 10.20. Service Manager 9.40 can still work with the old XSLT-based Service Manager 9.x adapter. For information about setting up the integration using this XSLT-based adapter, see the *UCMDB-SM Integration Guide (Using Service Manager 9-x Adapter)*.

You must set up the following required components to establish an integration between UCMDB and Service Manager.

- HP Universal CMDB installation
  Add a UCMDB Probe for the population feature if you do not already have one.
- HP Service Manager installation
  Add the UCMDB URL to the System Information Record. See "How to Add the UCMDB Connection Information" on page 32.
- Network connection between the HP Universal CMDB and HP Service Manager systems.

For instructions on installing and configuring your systems, see the UCMDB and Service Manager documentation.

How to Migrate Your Integration

Starting with version 10.20, UCMDB has introduced an adapter named Service Manager Enhanced Generic Adapter, which is based on the enhanced UCMDB integration framework. This enhanced adapter can work with Service Manager (SM) to additionally provide the following major features that ServiceManagerAdapter9-x does not support:

- A Visual Mapping tool: a tool that provides a graphic user interface for easy field mapping between UCMDB and SM, without the need to work with XSLT mapping files as you do with the previous adapters. This tool still provides an XML editor, which allows you to directly edit the mapping file code lines.

- Centralized CI type customization. For more information, see "Centralized CI Management" on page 36.

For more information about the Visual Mapping tool and the Generic Adapter framework, refer to the *Universal CMDB Help Center*.

To use the Service Manager Enhanced Generic Adapter, existing customers need to migrate their product systems and integration configurations. The migration process is as follows:

1. "Upgrade Service Manager to Version 9.40" below
2. "Upgrade UCMDB to Version 10.20" below
3. "Enable the RESTful APIs for Custom CI Types in Service Manager" on the next page
4. "Reconfigure an Integration Point Using the Service Manager Enhanced Generic Adapter in UCMDB" on page 23
5. "Update the Configurations for Custom CI Types in UCMDB" on page 23

Upgrade Service Manager to Version 9.40

If you want to use the Service Manager Enhanced Generic Adapter (ServiceManagerEnhancedAdapter9-x) for the UCMDB-SM integration, you must upgrade Service Manager to version 9.40, which introduced the following features that are required for the adapter to work:

- Restful APIs for Push, Population, and Federation
- Restful APIs for Device Type retrieval
- Restful APIs for Device Type synchronization

For information about how to upgrade Service Manager to version 9.40, see the *Service Manager 9.40 Installation and Upgrade Documentation Center*.

Upgrade UCMDB to Version 10.20

The Service Manager Enhanced Generic Adapter (ServiceManagerEnhancedAdapter9-x) also relies on many new features introduced in UCMDB 10.20 to work.

For information about how to upgrade your UCMDB system to version 10.20, see the *Universal CMDB 10.20 Deployment Guide*.

Note: If you have upgraded one or both of your product systems but still want to stay with an old adapter, refer to the previous UCMDB-SM integration documentation that is based on your specific adapter.

Enable the RESTful APIs for Custom CI Types in Service Manager

If your existing UCMDB-SM integration environment uses any custom CI types, you need to update the External Access Definition of each custom CI type in Service Manager to enable its RESTful API. The Service Manager Enhanced Generic Adapter requires the ucmdbIntegration RESTful APIs to work.

The following table describes the parameters that you need to update in each External Access Definition.

Parameter	Description	Example Value
RESTful Enabled	Enables the RESTful API (must be set to true to enable the RESTful API)	true
Resource Collection Name	Specifies a unique name for the resource collection. The following convention is recommended: <WS Object Name>+'s'	ucmdbATMs
Resource Name	Specifies a name for the resource. The following convention is recommended: <WS Object Name>	ucmdbATM
Unique Keys	Unique keys of the relevant device table	logical.name
Max Records Returned in Query	Maximum number of returned records in one query	1000
Resource Collection Actions - POST	Action to invoke for a POST request on the resource collection	Create
Resource Actions - POST	Action to invoke for a POST request on the resource	Create
Resource Actions - PUT	Action to invoke for a PUT request on the resource	Update
Resource Actions - DELETE	Action to invoke for a DELETE request on the resource	Delete

To access these parameters, follow these steps:

1. Open the External Access Definition for the custom CI type.
   a. Click **Tailoring > Web Services** > **Web Service Configuration** to open the External Access Definition form.

   b. In the Service Name field, type `ucmdbIntegration`.

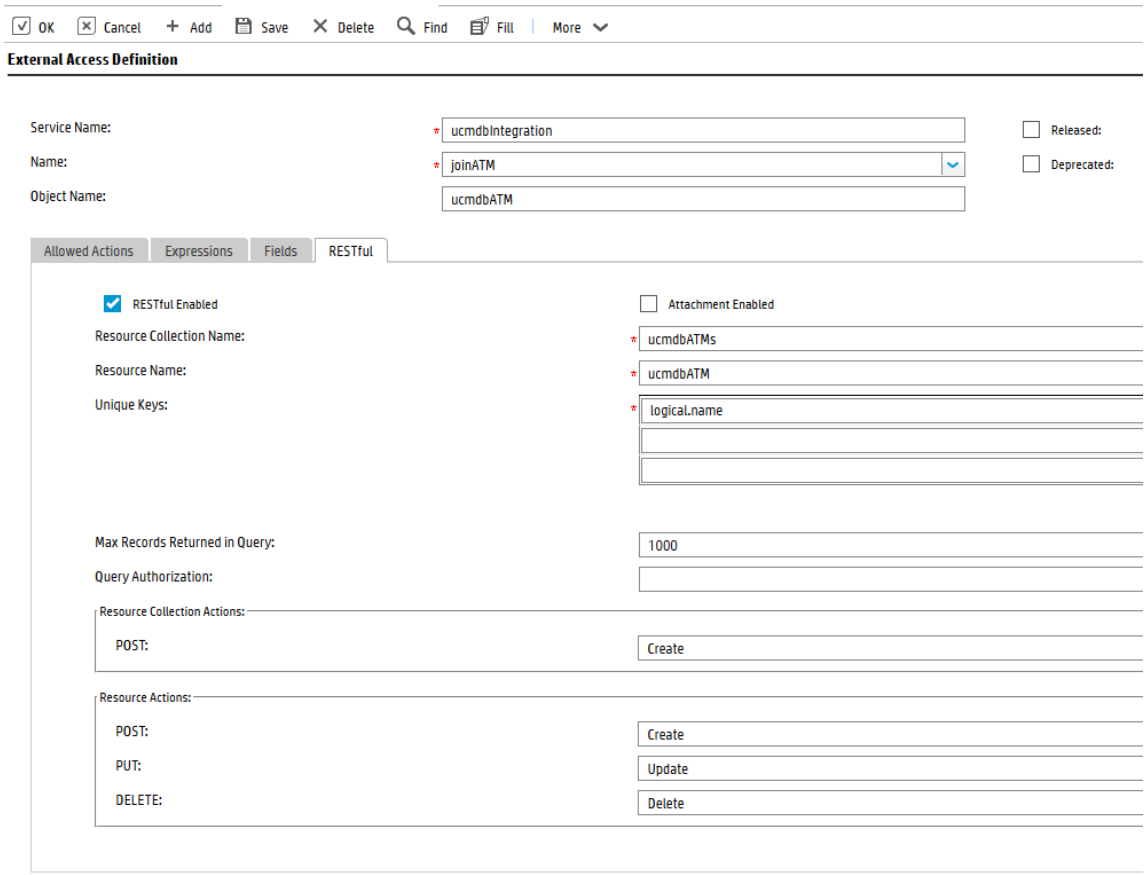
   c. In the Name field, select the table for the custom CI type.

   d. In the Object Name field, type the relevant web service object name.

   e. Click **Search**.

2. Click the **Restful** tab, and update the parameters.

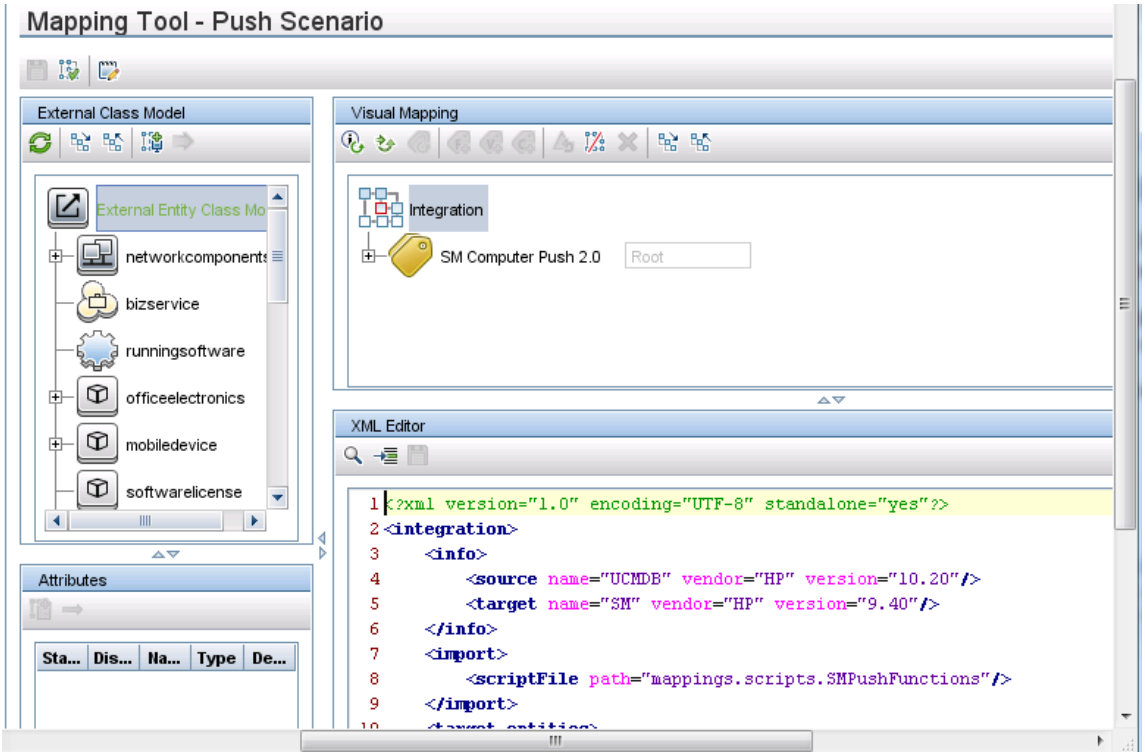
   As an example, the following figure shows the external access definition of a custom CI type named ATM Machine.



Reconfigure an Integration Point Using the Service Manager Enhanced Generic Adapter in UCMDB

To use the Visual Mapping tool, you need to enable the Service Manager Enhanced Generic Adapter (ServiceManagerEnhancedAdapater 9.x) on the UCMDB side. To do so, set up an integration point to use this adapter, and activate the integration point. For details, see "How to Create an Integration Point in UCMDB" on page 33.

Once this adapter is successfully enabled, you should be able to open one of the out-of-box mapping files (such as "SM Computer Push 2.0.xml") in the Visual Mapping tool interface without any problems. The following figure shows the Visual Mapping tool interface in which the "SM Computer Push 2.0.xml" file is open.



For details on how to use the Visual Mapping tool, refer to the *Universal CMDB 9.40 Data Flow Management Guide*.

Update the Configurations for Custom CI Types in UCMDB

The configurations for all out-of-the-box CI types in UCMDB 10.20 are already based on the Service Manager Enhanced Generic Adapter. However, you still need to perform the following tasks to manually update the configurations for your custom CI types in UCMDB.

Task 1. Convert the mapping scripts from XSLT to XML and Groovy.

The old Service Manager adapters use XSLT scripts for field mapping between UCMDB and SM. Most of the mapping scripts were developed for the four typical mapping scenarios described in the following table.

Scenario	Notes
One to One field mapping	• One field in one side is mapped to one field in the other side • The value is synchronized between UCMDB and SM directly (without value conversion)
One to Many field mapping	• One field in one side is mapped to many fields in the other side • The value of the one field is calculated based on the values of many fields
Many to Many field mapping	• Multiple fields are defined as a child CI type of the relevant CI type • A single record has multiple child instances
Value Conversion	• The value of one field is converted to another value based on a specific algorithm • Value conversion can happen on all kinds of field mappings (one to one, one to many, and many to many)

The Service Manager Enhanced Generic Adapter uses XML and Groovy mapping scripts. Old mapping scripts for the out-of-the-box CI types have been converted to XML and Groovy by default. However, you still need to convert your existing custom scripts to XML and Groovy.

To convert a custom mapping script, you need to identify its mapping scenario (One to One, One to Many, Many to Many, or Value Conversion) first; then you can convert the script by referring to the following sample scripts for your specific scenario.

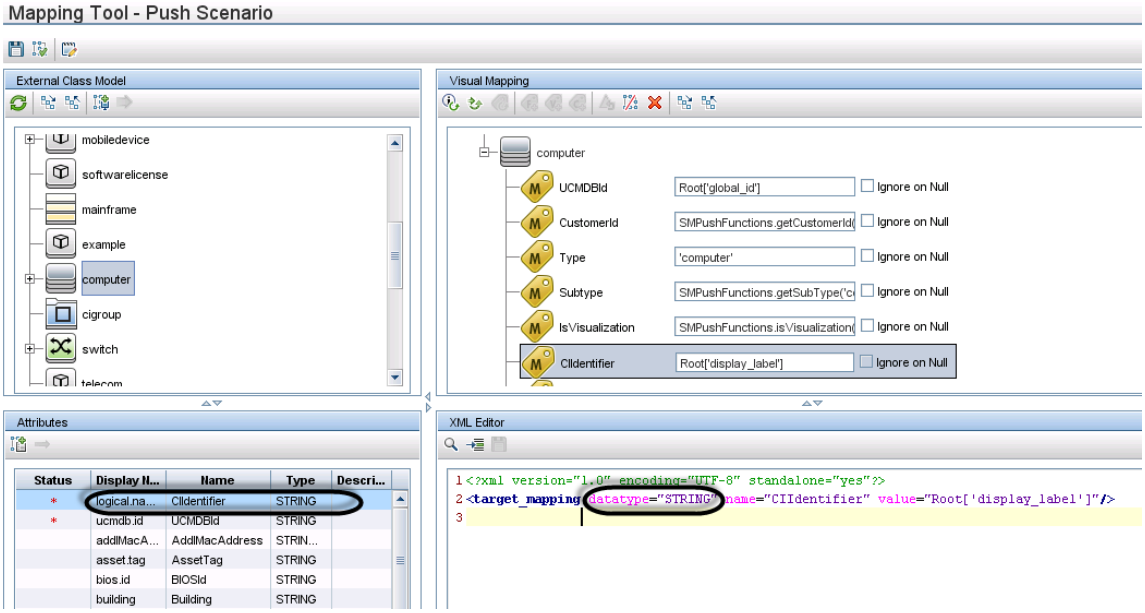
Tip: In all mapping scenarios, you can use the Visual Mapping tool to generate an XML script skeleton. Compared with the use of an old XSLT adapter, this is much easier as you can drag and drop the mapped fields to generate the script instead of writing every code line from scratch. For details about how to use XML and Groovy scripts in the UCMDB integration framework, see the *HP Universal CMDB 10.20 Developer Reference Guide.*

Tip: The Service Manager Enhanced Generic Adapter has four out-of-the-box Groovy scripts, which you can use as a reference:

- SMUtils.groovy: defines common methods used for push, population, and federation.
- SMPushFunctions.groovy: defines the methods used in the out-of-the-box push mapping scripts.
- SMPopulateFunctions.groovy: defines the methods used in the out-of-the-box population mapping scripts.
- SMFederationFunctions.groovy: defines the methods used in the out-of-the-box federation mapping scripts.

Additionally, the SMFederationConverter.groovy script defines value conversions for federation.

Field mapping must use the correct data type for each attribute. It is convenient to determine the data type for an attribute by using the Visual Mapping tool. The following figure illustrates the data type for the CIIdentifier attribute. You can either find the data type from the External Class Model Attributes pane or simply drag and drop the attribute from that pane to the Visual Mapping area to automatically get the correct data type.



One to One Field Mapping

Sample Script in XSLT:

```
<Type>computer</Type>
```

Or

```
<UCMDBId><xsl:value-of select="@id"/></UCMDBId>
```

Or

```
<xsl:for-each select="@display_label">
     <CIIdentifier><xsl:value-of select="."/></CIIdentifier>

</xsl:for-each>
```

Sample Script in XML (Groovy is not required in this scenario):

```
<target_mapping datatype="STRING" name="CIIdentifier" value="Root['display_label']
    "/>
```

Where: The **datatype** must be the correct data type defined in Service Manager for the attribute. You can find the data type from the Visual Mapping tool interface.

One to Many Field Mapping

Sample Script in XSLT:

```
<xsl:variable name="prefix" select="'Value&gt;'"/>
    <xsl:variable name="suffix" select="'&lt;/Value'"/>
    <Subtype>
        <xsl:choose>
        <xsl:when test="contains(@node_role,concat($prefix,'desktop',$suffix))
    ">Desktop</xsl:when>
        <xsl:when test="@os_family">
        <xsl:value-of select="@os_family"/>
        </xsl:when>
        <xsl:otherwise>Server</xsl:otherwise>
        </xsl:choose>
     </Subtype>
```

Sample Script in XML:

```
<target_mapping datatype="STRING" name="Subtype" value="SMPushFunctions.getSubType
    ('computer',Root['node_role'],Root['os_family'])"/>
```

Sample Script in Groovy:

You must develop a groovy function to implement the mapping logic. Usually the relevant fields in the counterpart system will be used as the parameters of the Groovy function.

Many to Many Field Mapping

Sample Script in XSLT:

```
<xsl:for-each select="cpus">
        <cpu>
            <xsl:for-each select="cpu">
                <cpu>
                    <CpuID><xsl:value-of select="@cpu_id"/></CpuID>
```

```
                    <CpuName><xsl:value-of select="@name"/></CpuName>
                    <CpuClockSpeed>
                          <xsl:value-of select="@cpu_clock_speed"/>
                    </CpuClockSpeed>
                </cpu>
            </xsl:for-each>
        </cpu>
    </xsl:for-each>
```

Sample Script in XML:

```
<for-each-source-entity count-index="i" source-entities="Root.Cpu">
        <target_entity name="cpu">
              <target_mapping datatype="STRING" name="CpuID" value="Root.Cpu[i]
    ['cpu_id']"/>
              <target_mapping datatype="STRING" name="CpuName" value="Root.Cpu[i]
    ['name']"/>
              <target_mapping datatype="STRING" name="CpuClockSpeed" value="Root.Cpu
    [i]['cpu_clock_speed']"/>
        </target_entity>
     </for-each-source-entity>
```

Where: <*source-entities*> (**Root Cpu**) comes from the local TQL query structure, while <*target_entity name*> (**cpu**) comes from the <cpu> tag in the XSLT file.

Note: Groovy scripts are not required for this scenario.

Value Conversion

Sample Script in XSLT:

```
<xsl:if test="@customer_id">
        <xsl:variable name="ucmdbCustomerId" select="@customer_id"/>
        <xsl:variable name="tenantMappingEntry" select="document('SM_MT_
    mapping.xml')/list['TenantMapping']/entry[@ucmdb=$ucmdbCustomerId]"/>
        <xsl:choose>
            <xsl:when test="$tenantMappingEntry">
                <CustomerId><xsl:value-of
    select="$tenantMappingEntry/@sm"/></CustomerId>
            </xsl:when>
            <xsl:otherwise>
                <CustomerId><xsl:value-of select="@customer_id"/></CustomerId>
            </xsl:otherwise>
        </xsl:choose>
    </xsl:if>
```

Or

```
<xsl:if test="contains(@node_role,concat($prefix,'virtualized_system',$suffix))">
            <IsVisualization>true</IsVisualization>
    </xsl:if>
```

Sample Script in XML:

```
<target_mapping datatype="STRING" name="CustomerId"
   value="SMPushFunctions.getCustomerId(CustomerInformation)"/>
```

Or

```
<target_mapping datatype="BOOLEAN" name="IsVisualization"
   value="SMPushFunctions.isVisualization(Root['node_role'])"/>
```

Note: Usually the relevant field in the counterpart system will be used as one of the parameters of the Groovy function.

Sample Scripts in Groovy:

```
/**
        * Priority Mapping
        * [uCMDB value : SM value ]
        */
       private static final def PriorityMapping = [
           "1_critical":"1",
           "2_high":"2",
           "3_average":"3",
           "4_low":"4"];

/**


     * Convert the Priority value from uCMDB to SM
        *
        * @param priority uCMDB Priority value
        * @return SM Priority value
        */
       public static String convertPriority(String priority, DataAdapterLogger log)
   {
           return convertEnumValue(priority,"Priority");
       }
```

Task 2. Update the configuration files.

Update the files as described in the following table.

File	Description
icon.properties	This configuration file defines the UCMDB icons that will be displayed in the mapping tool for Service Manager Device Types. If you have defined any custom device types in SM, you can update this configuration file to assign them appropriate icons. For more information, see the *Universal CMDB Data Flow Management Guide*.
sm.properties	This is the integration adapter configuration file. If you have modified the out-of-the-box settings in your old sm.properties file (for example, the number of concurrent threads), you need to update this configuration file accordingly for those options that are still valid in the Service Manager Enhanced Generic Adapter. For more information, see "How to Update the Integration Adapter Configuration File (sm.properties)" on page 129.
smFedConf.xml	This configuration file is used for federation and is introduced by the Service Manager Enhanced Generic Adapter. If you have any custom logic for the federation feature, such as custom fields for federation CIs, you need to update this configuration file. For more information, see "Troubleshooting Federation Issues" on page 204.
smPopConf.xml	This configuration file is used for population and replaces the old smPopConfFile.xml configuration file. If you have any custom logic for the population feature, such as custom query conditions, you need to update this configuration file. For more information, see the following topics: "What Is the Purpose of the <container> Element in the Population Configuration File (smPopConf.xml)?" on page 110 "No TQL Query Configured in smPopConf.xml " on page 200
smPushConf.xml	This configuration file is used for relationship data push and replaces the old smSyncConfFile.xml configuration file. For more information about this file, see the following topics: "How to Map a Relationship Type Query to the Service Manager Web Service Object" on page 176 "Query not Configured in smPushConf.xml" on page 194

Task 3. Enable Push, Population and Federation for CI types.

Once you have completed the tasks described above, you must edit your integration point to add Push and Population jobs for the CI types, as well as enable Federation for supported CI types. For details, see the following topics:

"How to Create an Integration Point in UCMDB" on page 33

"How to Define Data Push Jobs in UCMDB" on page 43

"How to Define Population Jobs in UCMDB" on page 38

"How to Add an Attribute of a Supported CI Type for Federation" on page 182

Integration Setup Overview

The integration requires setup on both the UCMDB and Service Manager systems.

This task includes the following steps:

1. Set up the Service Manager system.
   See "HP Service Manager Setup" below.

2. Set up the UCMDB system.
   See "HP Universal CMDB Setup" on page 33.

3. Run the UCMDB population jobs to synchronize CIs to UCMDB.
   See "Populating UCMDB with Service Manager CI Data" on page 38.

4. Run the UCMDB data push jobs to transfer CIs to Service Manager.
   See "Populating UCMDB with Service Manager CI Data" on page 38.

HP Service Manager Setup

You must complete the following tasks from your Service Manager system to support the integration.

1. Create a dedicated integration user account in Service Manager.
   See "How to Create an Integration User Account" on the next page.

2. Add the UCMDB connection information to the system information record.
   See "How to Add the UCMDB Connection Information" on page 32.

How to Create an Integration User Account

This integration requires an administrator user account for UCMDB to connect to Service Manager. The user account must already exist in both UCMDB and Service Manager.

Note: The integration user account must have the **RESTful API** capability word in Service Manager, if the integration uses the Service Manager Enhanced Generic Adapter, which requires the Service Manager ucmdbIntegration RESTful APIs to work.

To create a dedicated integration user account in Service Manager:

1. Log in to Service Manager as a system administrator.
2. Type `contacts` in the Service Manager command line, and press ENTER.
3. Create a new contact record for the integration user account.
   a. In the Contact Name field, type a name. For example, `UCMDB`.
   b. Click **Add**, and then **OK**.
4. Type `operator` in the Service Manager command line, and press ENTER.
5. In the Login Name field, type the username of an existing system administrator account, and click **Search**.
   The system administrator account is displayed.
6. Create a new user account based on the existing one.
   a. Change the Login Name to the integration account name that you want (for example, **ucmdb**).
   b. Type a Full Name. For example, `UCMDB`.
   c. In the Contact ID field, click the **Fill** icon and select the contact record that you have just created.
   d. Click **Add**.
   e. Select the **Security** tab, and change the password.
   f. Select the **Startup** tab, and add the **RESTful API** capability word for the operator.
   g. Click **OK**.

The integration user account is created. Later you will need to add this user account (username/password) in UCMDB, and then specify this user account in the Credentials ID field when

creating an integration point in UCMDB. See "How to Create an Integration Point in UCMDB" on the next page.

How to Add the UCMDB Connection Information

Service Manager requires the UCMDB connection information to obtain CI attribute information from the UCMDB system, and display it in the Actual State section in the Service Manager configuration item form.

Caution: If you do not specify the correct connection information, an error, instead of UCMDB CI information, will be displayed in the Actual State section.

To add UCMDB connection information in Service Manager:

1. Log in to Service Manager as a system administrator.
2. Click **System Administration** > **Base System Configuration** > **Miscellaneous** > **System Information Record**.
3. Click the **Active Integrations** tab.
4. Select the **HP Universal CMDB** option.
5. In the **UCMDB webservice URL** field, type the URL to the HP Universal CMDB web service API. The URL has the following format:
   http://*<UCMDB server name>*:*<port>*/axis2/services/ucmdbSMService

   Replace *<UCMDB server name>* with the host name of your UCMDB server, and replace *<port>* with the communications port that your UCMDB server uses.
6. In the **UserId** and **Password** fields, type the user credentials that are required to manage CIs on the UCMDB system. For example, the out-of-the-box administrator credentials are **admin/admin**.
7. Optionally, if you want to enable an integration to UCMDB Browser, in the **UCMDB Browser URL** field, type your UCMDB Browser URL in the following format:
   http://*<UCMDB browser server name>*:*<port>*/ucmdb-browser

   For example: http://myucmdbbrowserserver:8081/ucmdb-browser

   Note: If you specify the UCMDB Browser URL here, the **View in UCMDB Browser** button will

> replace the **View in UCMDB** button in CI records synchronized from UCMDB; only when you leave this field empty, the **View in UCMDB** button will appear.

8. Click **Save**. Service Manager displays the message: Information record updated.
9. Log out of the Service Manager system.
10. Log back into the Service Manager system with an administrator account.

The Actual State section and the **View in UCMDB Browser** or **View in UCMDB** button will be available in CI records pushed from UCMDB.

HP Universal CMDB Setup

You must complete the following task from your UCMDB system to support the integration:

Create an integration point between UCMDB and Service Manager. See "How to Create an Integration Point in UCMDB" below.

How to Create an Integration Point in UCMDB

A default UCMDB installation already includes the ServiceManagerEnhancedAdapter9-x package. To use the integration package, you must create an integration point that lists the connection properties for the integration.

> **Caution:** For data population, this integration supports the use of only one probe for your Service Manager system. In other words, you cannot run population jobs on different probes by setting up multiple integration points with different probes for your Service Manager system. Only one probe is allowed for one Service Manager system.

To create an integration point, follow these steps:

1. Log in to UCMDB as an administrator.
2. Add the integration user account that you created in Service Manager.
   a. Click **Administration** > **Users and Roles**.
   b. Click the **Add New User** icon  .

c. For User Name and Password, type the user name and password that you created in Service Manager. See "How to Create an Integration User Account" on page 31.

d. Click **Next**, and then select **Admin** from the Role List .

e. Click **Finish**. The integration user account is added.

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

4. Click the **New Integration Point** icon  .
UCMDB displays a New Integration Point properties window.

5. Complete the integration and adapter property fields as described in the following table.

Field name	Is required?	Description
Integration Name	Yes	Type the name (unique key) of the integration point. For example, **sm_integration**.
Integration Description	No	Type a description for the integration point.
Adapter	Yes	Select **HP Software Products** > **Service Manager** > **ServiceManagerEnhancedAdapter9.x**.
Is Integration Activated	Yes	Enable this option to indicate the integration point is active.
Hostname/IP	Yes	Type the hostname or IP address of the Service Manager server. For example, `localhost`.
Port	Yes	Type the communications port of the Service Manager server. For example, `13080`.

, continued

Field name	Is required?	Description

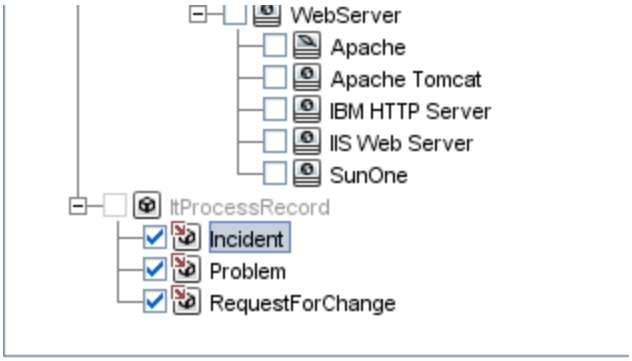
URL Override	No	This field value (if any) supersedes the Hostname/IP and Port settings described above.
		
		Use this field If you want UCMDB to connect to Service Manager in any combinations of the following ways:
		
		■ Connect to Service Manager over HTTPS or over both HTTP and HTTPS
		
		■ Connect to multiple Service Manager server nodes (horizontally scaled environment)
		
		■ Connect to one single Service Manager server node through multiple ports (vertically scaled environment)
		
		For more information, see "Push in Clustered Environments" on page 93.
		
		Type one or more Service Manager web services URLs (separated by a semicolon) in this field.
		
		The following are two example values of this field (each URL must use this format: `http(s)://<hostname>:<port>/SM/9/rest`):
		
		■ `https://localhost:13443/SM/9/rest`
		
		■ `http://localhost:13080/SM/9/rest;`
		`https://localhost:13443/SM/9/rest;`
		`http://smfpe04:13080/SM/9/rest`

Credentials ID	Yes	Click **Generic Protocol**, click the **Add** icon to add the integration user account that you created, and then select it. This account must exist in both Service Manager and UCMDB. See "How to Create an Integration User Account" on page 31.
Data Flow Probe	Yes	Select the name of the Data Flow Probe used to run population jobs. You should have already added the data flow probe for the integration after installing UCMDB. See "Integration Requirements" on page 19.

6. Click **Test Connection** to make sure that a successful connection has been established.

7. Click **OK**.
   The integration point is created and its details are displayed.

8. Click the **Federation** tab, and complete the following configuration.
   a. In Supported and Selected CI Types, select the following CI types as needed from **ItProcessRecord**:
      i. Incident
      ii. Problem
      iii. Request for Change



   b. For each CI type you selected (Incident, Problem, or RequestForChange), select **Retrieve CIs of selected CI Type** for **CI Type Retrieval Mode**.

9. Click the **Population** and **Data Push** tabs to view the default integration job details.

   Note: UCMDB creates several default population and data push jobs when creating an integration point. If needed, you can create a new job for the integration point. For information about creating integration jobs, see "How to Define Data Push Jobs in UCMDB" on page 43 and "How to Define Population Jobs in UCMDB" on page 38.

10. Click the **Save Integration Point** icon .

Centralized CI Management

The Service Manager Enhanced Generic Adapter allows centralized CI type management. Centralized CI management enables you to manage CIs from UCMDB without the need to do much tailoring work on the Service Manager (SM) side. You can then synchronize CIs from UCMDB to SM through data push.

Managing Out-Of-the-Box CI Types

Service Manager and UCMDB provide out-of-the-box CI types, as well as mapping files for them. To modify an out-of-the-box CI type (for example, adding new attributes according to your business needs), you only need to open the Visual Mapping tool, manually add new attributes to this CI type, and then apply the update without leaving UCMDB. The new attribute will be automatically added to the relevant web service object on the SM side.

Managing New Custom CI Types

When you create a new CI type in UCMDB, all relevant SM objects (DBDICT, JoinDef, Web Service API, DEM Rule, and so on) for the new CI type are automatically created on the SM side, except the format (the new CI type will use **configurationItem** as the default format).

To add a new CI type according to your business needs, you only need to add a new CI type and create a TQL query in UCMDB, and then create a matching SM CI type by using the Visual Mapping tool without the need to leave UCMDB.

Note: The population feature enables you to synchronize CIs from SM to UCMDB. For information about the circumstances under which you need to use the population feature, see "When Do I Need the Population Feature?" on page 106.

Visual Mapping Tool

The Service Manager Enhanced Generic Adapter comes with a Visual Mapping tool, which provides a graphic user interface for you to configure value and field mappings for complex data push or population configurations.

The following table describes the panes of this graphic user interface.

Pane	Description
Local Query	Displays a hierarchical tree structure of the local TQL query in the CMDB.
Local Query Attributes	This is the Attributes pane for the Local Query pane. It displays attributes of a local integration TQL query.
Visual Mapping	Allows you to establish mapping for items you select from the Local Query pane and the External Class Model pane through a drag and drop.
XML Editor	Allows to you edit XML mapping files in a text editor.

Pane	Description
External Class Model	Displays a hierarchical tree structure of the external class model. For the UCMDB-SM integration, this pane displays supported CI types from the Service Manager (SM) side.
External Class Model Attributes	The Attributes pane for the External Class Model pane displays attributes of an external class model. For the UCMDB-SM integration, this pane displays attributes from the SM side.

Populating UCMDB with Service Manager CI Data

In addition to pushing CI data from UCMDB to Service Manager, this integration also supports the population of CI data (including CIs and CI relationships) from Service Manager to UCMDB. The integration can then update the list of CIs in UCMDB if new CIs or new attribute values are found in Service Manager. The population of data from Service Manager to UCMDB is defined in the Integration Studio in UCMDB. You can manually run the population jobs, however HP recommends that you schedule these jobs to keep your CIs and CI attributes up to date.

This task includes the following steps:

1. Define CI/CI Relationship population jobs in UCMDB.
   See "How to Define Population Jobs in UCMDB" below.
2. View the transferred CI/CI Relationship data in UCMDB.
   See "View Service Manager CI Data in UCMDB" on page 41.
3. Schedule CI population jobs to keep CIs and CI attributes up to date.
   See "How to Schedule CI Population Jobs" on page 41.

How to Define Population Jobs in UCMDB

A CI or relationship population job copies certain types of CIs or relationships from Service Manager to UCMDB.

To define a CI or CI relationship population job, follow these steps:

1. Log in to UCMDB as an administrator.
2. Navigate to **Data Flow Management** > **Integration Studio**. UCMDB displays a list of existing integration points.
3. Open the integration point that you created for Service Manager.
4. Click the **Population** tab, and add a new job as follows.

   Note: UCMDB creates several default population and data push jobs when creating an integration point. The following table lists the default population jobs and their queries. If needed, you can create, update or remove queries for each job.

Queries for CI / CI relationship population

Integration Job	Queries for CI / CI relationship population
SM Configuration Item Population job	Out-of-the-box, the following queries are available for this job, which populates UCMDB with CI records from Service Manager: ■ **SM Business Service Population 2.0**: Populates UCMDB with CIs of the bizservice type. ■ **SM RunningSoftware Population 2.0**: Populates UCMDB with CIs of the RunningSoftware type. ■ **SM Computer Population**: Populates UCMDB with CIs of the computer type. ■ **SM CLIP Down Time Population 2.0**: Populates UCMDB with CIs of the planned outage type.

Queries for CI / CI relationship population, continued

Integration Job	Queries for CI / CI relationship population
SM Relations Population job	Out-of-the-box, the following queries are defined for this job, which populates UCMDB with CI Relationship records from Service Manager: ■ **SM Biz To Biz With Containment 2.0**: Populates UCMDB with CI relationships in which a bizservice CI contains another. ■ **SM Biz To Biz With Usage 2.0**: Populates UCMDB with CI relationships in which a bizservice CI uses another. ■ **SM Biz To Computer With Containment 2.0**: Populates UCMDB with CI relationships in which a bizservice CI contains a computer CI. ■ **SM Biz To Computer With Usage 2.0**: Populates UCMDB with CI relationships in which a bizservice CI uses a computer CI. ■ **SM Computer To Computer With Connects 2.0**: Populates UCMDB with CI relationships in which a computer CI connects to another. ■ **SM Biz To Software With Containment 2.0**: Populates UCMDB with CI relationships in which a bizservice CI contains a software CI. ■ **SM Biz To Software With Usage 2.0**: Populates UCMDB with CI relationships in which a bizservice CI uses a software CI. ■ **SM Computer Composition Software 2.0**: Populates UCMDB with CI relationships in which a computer CI connects to a software CI. ■ **SM CI Connection Down Time CI 2.0**: Populates UCMDB with CI relationships in which a CI connects to a down time CI.

a. Click the **New Integration Job** icon  .

b. Type a Name for the integration job. For example, **CI_Population_Job1**.

c. Click the **Add** icon  to add existing queries to the job (see the table above).

d. Select the **Allow Integration Job to delete removed data** check box for the query.

e. Click **OK** to save the job.

5. Run the job manually to see if the integration job works properly.
   a. To populate UCMDB with all relevant data for the job, click the  icon.
   b. To populate UCMDB with only CI data changes since the job last ran, click the  icon.
6. Wait for the job to complete, and click the **Refresh** icon multiple times as needed until the job is completed.

   Note: When the job is completed, the job status becomes one of the following: Succeeded, Passed with failures, or Failed.
7. Click the **Statistics** tab to view the results. If the job failed, click the **Query Status** tab and **Job Errors** tab for more information. For details, see "Troubleshooting Population Issues" on page 198.
8. Click **OK**.

If the job is completed successfully, you can view the transferred CI data in UCMDB and schedule the job so that it can run automatically.

View Service Manager CI Data in UCMDB

After a population job is successfully completed, you can search for the Service Manager CI records in UCMDB, and verify that their attributes are correctly populated.

The Service Manager **CI Identifier**field is populated to the **Name** field on the Configuration Item Properties pane in UCMDB.

Note: To see the entire attribute mappings of a CI type, you can open its population XML file (for example, **SM Business Service Population.xml**), where the UCMDB attribute field names and the mapped Service Manager web service field caption names are defined. For more information, see "Tailoring the Integration" on page 113.

How to Schedule CI Population Jobs

You can schedule CI population jobs to match the discovery/maintenance schedule of your Service Manager feeders. For example, if your Service Manager feeders send CI data updates on a daily schedule, then the population jobs must also run on a daily schedule. By using a matching schedule you can ensure that your UCMDB system always has the most current CI data.

This task includes the following steps:

1. Log in to UCMDB as an administrator.
2. Navigate to **Data Flow Management** > **Integration Studio**. UCMDB displays a list of integration points.
3. Open the integration point for Service Manager.
4. Click the **Population** tab, and select a population job from the list.
5. Click the **Edit Integration Job** icon.
6. Select the **Scheduler enabled** option.
7. Select the scheduling options that you want to use. For example, select Repeat every: **Day** and Ends: **Never**.
8. Select a Time Zone.
9. Click **OK**.

Pushing UCMDB CI Data to Service Manager

The integration requires a one-time transfer of CIs from UCMDB to Service Manager to populate the Service Manager system with CIs. The integration will then update the list of CIs in Service Manager when UCMDB discovers new CIs or new attribute values. The integration accomplishes the push of CI data using data push jobs in the UCMDB system. HP recommends that you schedule these jobs to keep your CIs and CI attributes up to date.

Centralized CI Management

The Service Manager Enhanced Generic Adapter allows centralized CI type management.When you create a new CI type in UCMDB, all relevant SM objects (DBDICT, JoinDef, Web Service API, DEM Rule, and so on) for the new CI type are automatically created on the SM side, except the format (the new CI type will use **configurationItemNode** as the default format). Centralized CI management enables you manage CIs in UCMDB without manually

This task includes the following steps:

1. Define CI/CI Relationship data push jobs.
   See "How to Define Data Push Jobs in UCMDB" on the next page.

2. View the CI/CI Relationship data pushed from UCMDB.
   See "How to View UCMDB CI Data in Service Manager" on page 47.

3. Schedule data push jobs to keep CI/CI Relationship data up to date.
   See "How to Schedule Data Push Jobs" on page 46.

How to Define Data Push Jobs in UCMDB

Data push jobs copy CI or CI Relationship records from your UCMDB system to your Service Manager system.

To define a CI or CI relationship push job, follow these steps:

1. Log in 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 that you created for Service Manager. For example, **sm_integration**.

4. Click the **Data Push** tab.

5. Follow these steps to add a new data push job:

   a. Click the **New Integration Job** icon  .

> **Note:** UCMDB creates a default data push job when creating an integration point. The following table lists the default data push job and its queries. If needed, you can create, update, or remove queries for the push job. To access these out-of-the-box queries for push, go to **Modeling** > **Modeling Studio** > **Resources**, select **Queries** for Resource Type, and then navigate to **Root** > **Integration** > **Push**. For information about tailoring data push queries, see "How to Create a Query to Synchronize the CI Type" on page 159.

Queries for CI / CI Relationship Push

Integration job	Queries
Integration job	Queries

Queries for CI / CI Relationship Push , continued

Integration job	Queries
SM Push job	Out-of-the-box, the following queries are available for this job, which pushes CI records from UCMDB to Service Manager: ■ **SM Mainframe Push 2.0**: pushes CIs of the mainframe type. ■ SM Business Element Push 2.0: pushes CIs of the business application, and infrastructure service types ■ **SM Network Component Push 2.0**: pushes CIs of the network component type. ■ **SM Running Software Push 2.0**: pushes CIs of the running software type. ■ **SM Business Service Push 2.0**: pushes CIs of the business service type. ■ **SM Computer Push 2.0**: pushes CIs of the computer type. ■ **SM Storage Push 2.0**: pushes CIs of the storage type. ■ **SM Switch Push 2.0**: pushes CIs of the switch type. ■ **SM Net Printer Push 2.0**: pushes CIs of the net printer type. ■ **SM Cluster Push 2.0**: pushes CIs of the cluster type. ■ **SM Mobile Device Push 2.0**: pushes CIs of the mobile device type. ■ **SM Local Printer Push 2.0**: pushes CIs of the local printer type. Out-of-the-box, the following queries are available for this job, which pushes CI Relationship records from UCMDB to Service Manager: ■ **SM Layer2 Topology Relations Push 2.0**: pushes compound CI relationships between nodes. ■ **SM Business Service Relations Push 2.0**: pushes CI relationships whose upstream CI type is business service. ■ **SM CRG Relations Push 2.0**: pushes CI relationships whose upstream CI type is cluster. ■ **SM Node Relations Push 2.0**: pushes direct CI relationships whose upstream CI

Queries for CI / CI Relationship Push , continued

Integration job	Queries
	type is node.

b. In the **Name** field, type a unique name for the job. For example, **CI_Push_Job1**.

c. Click the **Add Query** icon to add existing queries to the job.

d. Select the **Allow Integration Job to delete removed data** option for each query.

e. Click **OK** to save the job.

6. Run the job manually to verify that the integration job works properly.

Caution: If you have a huge amount of CI data in your UCMDB system, and this is your first time to push CI /CI Relationship data to Service Manager, it is recommended to select the "Add the record" option instead of "Open a change" or "Open an incident" for "Action if matching record does not exist" in each Discovery Event Manager Rules definition. Failure to do so may cause unnecessary performance problems. For details, see "How to Add Discovery Event Manager Rules" on page 136.

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

b. To push only data that was changed since the job last ran, click the icon.

Tip: You can stop a running push job by pressing the **Stops the selected job** icon .

7. Wait for the job to complete, and click the **Refresh** icon multiple times as needed until the job is completed.

Note: When the job is completed, the job status becomes one of the following depending on the results: `Completed successfully`, `Completed`, or `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 details, see "Troubleshooting Data Push Issues" on page 189.

9. Click **OK**.

If the job is completed successfully, you can view the UCMDB CI data in Service Manager, and schedule the job so that it can run automatically.

How to Schedule Data Push Jobs

It is a best practice to schedule the data push jobs to match the discovery schedule of your Service Manager feeders. For example, if your Service Manager feeders send CI data updates on a daily schedule, the data push jobs must also run on a daily schedule. By using a matching schedule you can ensure that your Service Manager system always has the most current CI data.

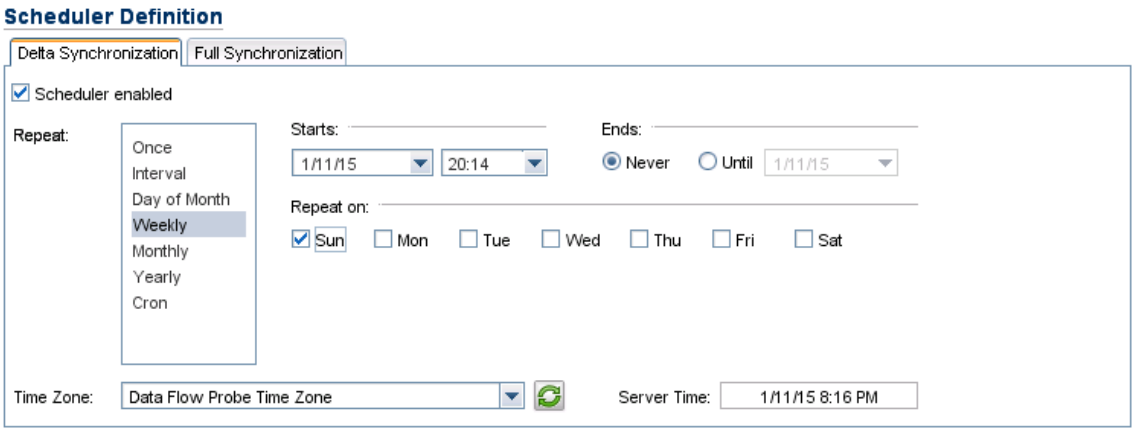
UCMDB allows you to schedule updates directly from a data push job. This task includes the following steps:

1. Log in to UCMDB as an administrator.
2. Navigate to **Data Flow Management** > **Integration Studio**. UCMDB displays a list of integration points.
3. Select the integration point that you created for Service Manager. For example, **SM Integration**.
4. Click the **Data Push** tab.
5. Select a push job. For example, **SM Configuration Item Push Job 2.0**.
6. Click the **Edit Integration Job** icon  .

   Tip: UCMDB allows you to define two different schedules for two types of data push: Delta Sync, and Full Sync. For recommendations on push scheduling, see "Push Scheduling Recommendations" on page 92.

7. Define a schedule for Delta Sync.
   a. Click the **Delta Synchronization** tab.
   b. Select the **Scheduler enabled** option.

c. Select the scheduling options that you want to use.



8. Click the **Full Synchronization** tab, and select the scheduling options that you want to use.
9. Click **OK** to save the data push job.
10. Repeat step 6 to step 9 for the rest of data push jobs of the integration point.
11. Save the integration point.

How to View UCMDB CI Data in Service Manager

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

CI records pushed from UCMDB contains a **View in UCMDB** or **View in UCMDB Browser** button, which enables you to access UCMDB or UCMDB Browser to view the CI information.

Note:

- If you specified the UCMDB Browser URL in the System Information Record in SM, the **View in UCMDB Browser** button is displayed; otherwise the **View in UCMDB** button is displayed.
- The UCMDB Browser is a lightweight UI designed for simple access to UCMDB configuration information. This is a tool for searching, locating and consuming configuration related data. It is an optional add-on to UCMDB. For more information, refer to the UCMDB Browser documentation.

To view UCMDB CI data in Service Manager:

1. Log in to Service Manager as a system administrator.
2. Navigate to **Configuration Management** > **Search CIs**.
3. Open a CI record pushed from UCMDB.
4. If the **View in UCMDB** button is available, view the CI record in UCMDB.
   a. Click the **View in UCMDB** button. The UCMDB login screen opens.
   b. Type a UCMDB username and password to log in.

   The CI record opens in UCMDB. You can view its properties.

   Note: You can enable Lightweight Single Sign-On (LW-SSO) for the integration so that Service Manager web client users can bypass the UCMDB login screen. For details, see "How to Enable Lightweight Single Sign-On (LW-SSO) Configuration" on page 102.
5. If the **View in UCMDB Browser** button is available, view the CI record in the UCMDB Browser.
   a. Click the **View in UCMDB Browser** button. The UCMDB Browser login screen opens.
   b. Type a UCMDB Browser username and password to log in. The CI record opens in the UCMDB Browser. You can view its properties and other information.
6. Open the **Actual State** section.
   Service Manager sends a web services request to UCMDB and displays all CI attributes that the request returns.

   Note: The web services request uses the UCMDB webservice URL and account (for example, admin/admin) defined in the System Information Record in Service Manager. See "How to Add the UCMDB Connection Information" on page 32.

How to View the Change History of the Primary CI of a Problem Record

When integrated with UCMDB Browser, Service Manager displays a **Primary CI History in UCMDB** section in a problem record whose primary CI is synchronized from UCMDB. You can view the CI changes on that primary CI for root cause investigation.

For information about how to enable an integration with UCMDB Browser, see "How to Add the UCMDB Connection Information" on page 32.

To view the primary CI change history, follow these steps:

1. Log in to Service Manager.
2. Navigate to **Problem Management**, and perform a search to open a problem record whose Primary CI is synchronized from UCMDB.
3. Click the **Primary CI History in UCMDB** tab.

Federating Service Manager Ticket Data to UCMDB

Federation does not physically copy data from Service Manager (SM) to UCMDB; it only retrieves SM data for displaying in UCMDB. Out-of-the-box, the UCMDB-SM integration supports federation for the Incident, Problem, and RequestForChange external CI types in UCMDB. If you have enabled these CI types for federation when creating your integration point, in UCMDB you can retrieve Incident, Problem, and Change record data from SM.

Out-of-the-box, each of the CI types (Incident, Problem, or Change) supports federation of a subset of their attributes in Service Manager; however, the integration can federate more attributes if tailored as such.

This section includes:

- "Federation Queries" below
- "Examples of Using Federation" on the next page
- "How to Add an Attribute of a Supported CI Type for Federation" on page 182

Federation Queries

Federation uses queries to determine what data to retrieve from Service Manager. To retrieve specific ticket data from Service Manager, you need to create a TQL query first.

The ServiceManagerEnhancedGenericAdapter9-x provides a bunch of sample federation queries, which you can use as a reference. They are available from the following path in UCMDB: **Modeling** > **Modeling Studio** > **Resources** > **Integration** > **Service Manager** > **Federation**.

Examples of Using Federation

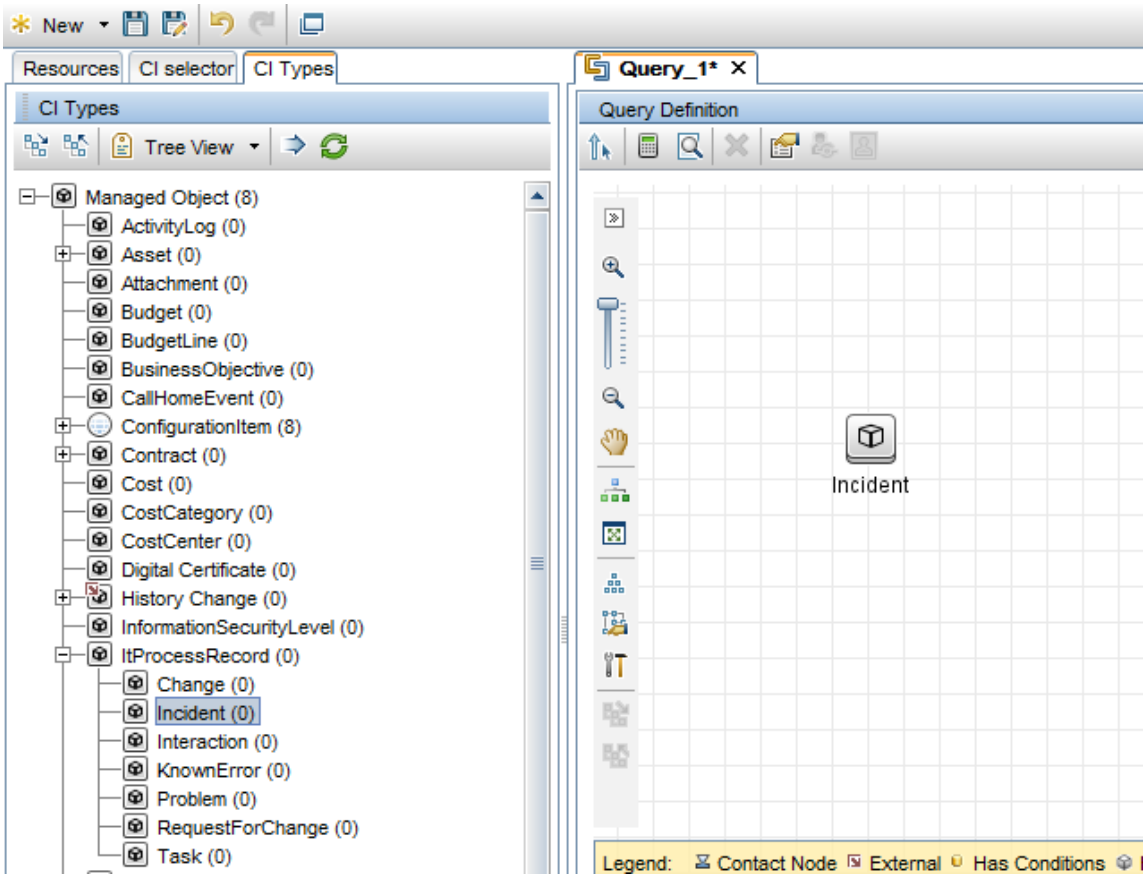
You can use the federation feature in many different ways. The following are only examples of using the feature.

This section describes four examples:

Example 1: Federate All SM Incident Tickets

This example illustrates how you retrieve information of all Incident records that exist in Service Manager.

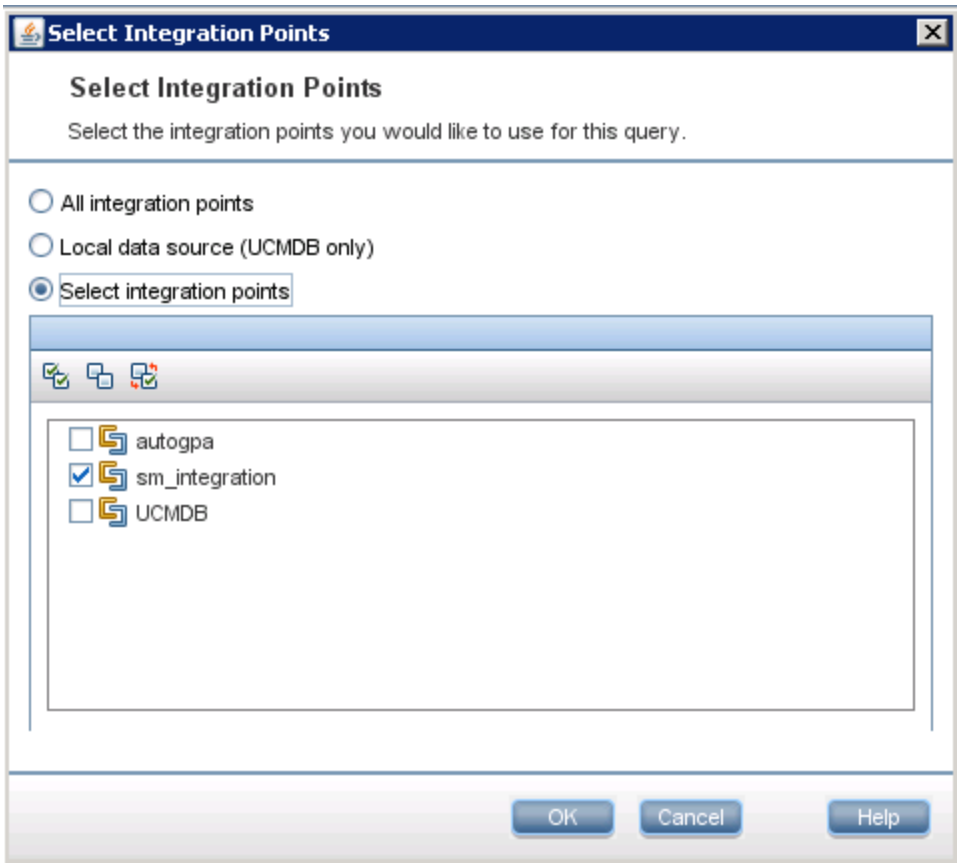
1. Log in to UCMDB as an administrator.
2. Navigate to **Modeling** > **Modeling Studio** > **Resources**.
3. For Resource Type, select **Queries** from the list.
4. Click **New** > **Query**.
5. On the **CI Types** tab, navigate to **ItProcessRecord** > **Incident**, and drag it to the query pane on the right side.



6. Specify Service Manager as the data source for the Incident query node.
   a. Select the Incident query node, click the **Data Sources** tab on the lower right pane, and then click **Edit**.

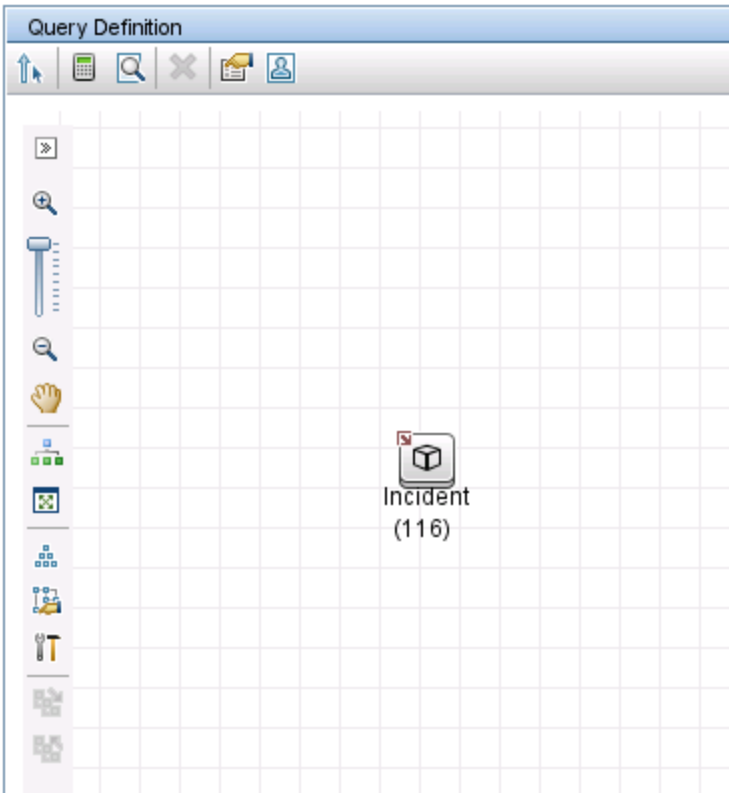
   b. Select the **Select integration points** option, and then select your integration point name (for

example, **sm_integration**). Click **OK**.

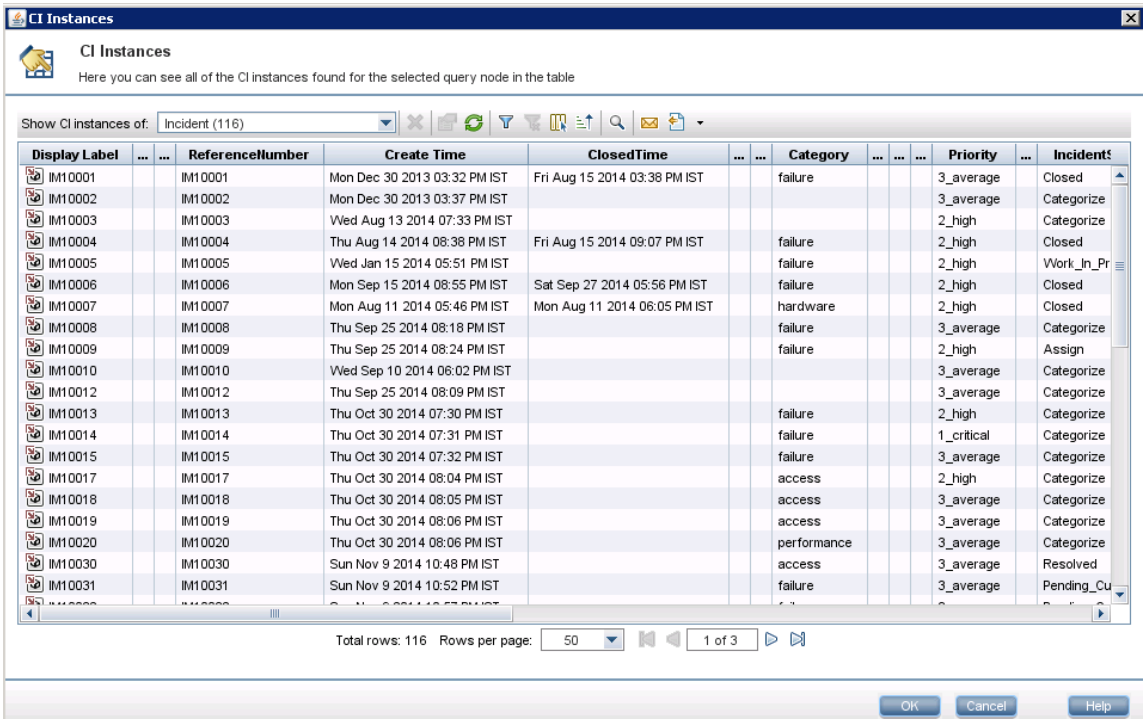


7. Click the **Save** icon , and then type a query name and select a location in which to save the query (for example, select the **Root** folder).

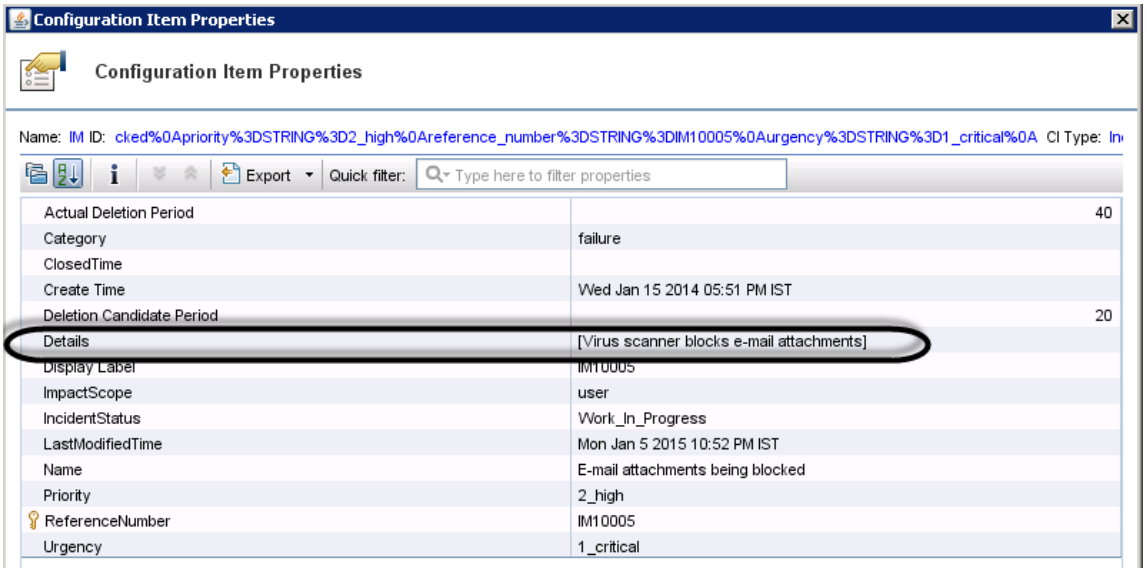
8. Select the Incident query node, and then click the **Calculate Query Result Count** icon . UCMDB returns the query result count. For example, the following figure shows that there are a total of 116 Incident records in Service Manager.



9. Right-click the Incident query node, and select **Show Element Instances**. UCMDB displays a list of all Incident records that exist in Service Manager.



10. Select an Incident record from the list, and click the **Properties** icon  to view its details.

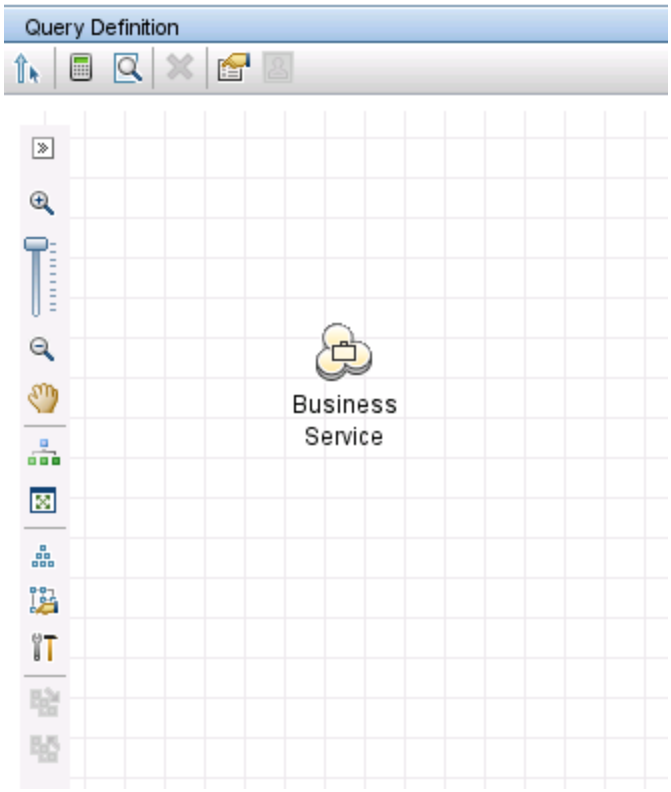


Example 2: Federate SM Incident Records that Affect a UCMDB Business Service CI

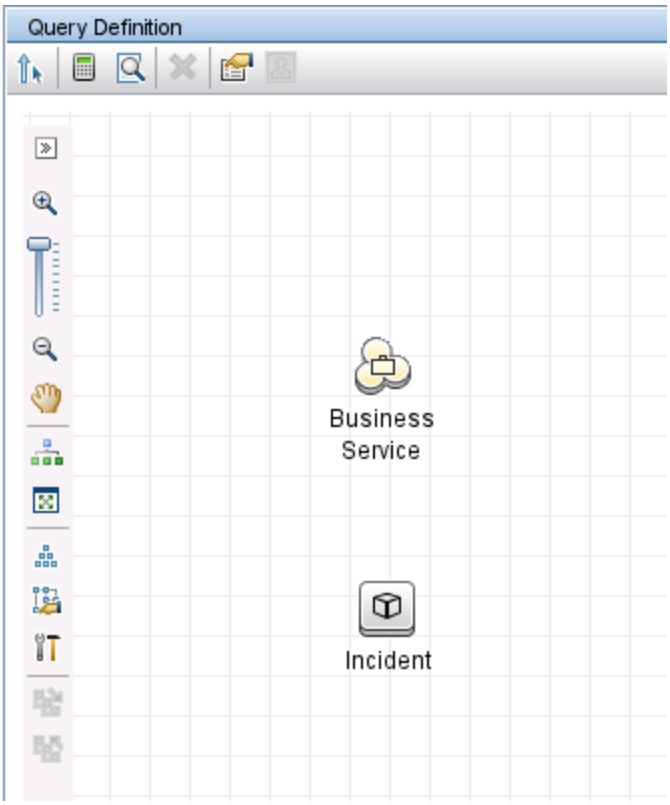
The following example illustrates how to federate a list of Service Manager Incident records whose Affected Service or Affected CI field contains a UCMDB Business Service CI.

In this example, IM10005 in SM has an affected service named **bizservice1**, which was pushed from UCMDB.

1. Log in to UCMDB as an administrator.
2. Navigate to **Modeling** > **Modeling Studio** > **Resources**.
3. For Resource Type, select **Queries** from the list.
4. Click **New** > **Query.**
5. On the **CI Type** tab, navigate to **ConfigurationItem** > **BusinessElement** > **Service** > **BusinessService**, and drag it to the query pane on the right side.



6. Navigate to **ItProcessRecord** > **Incident**, and drag the icon to the query pane.

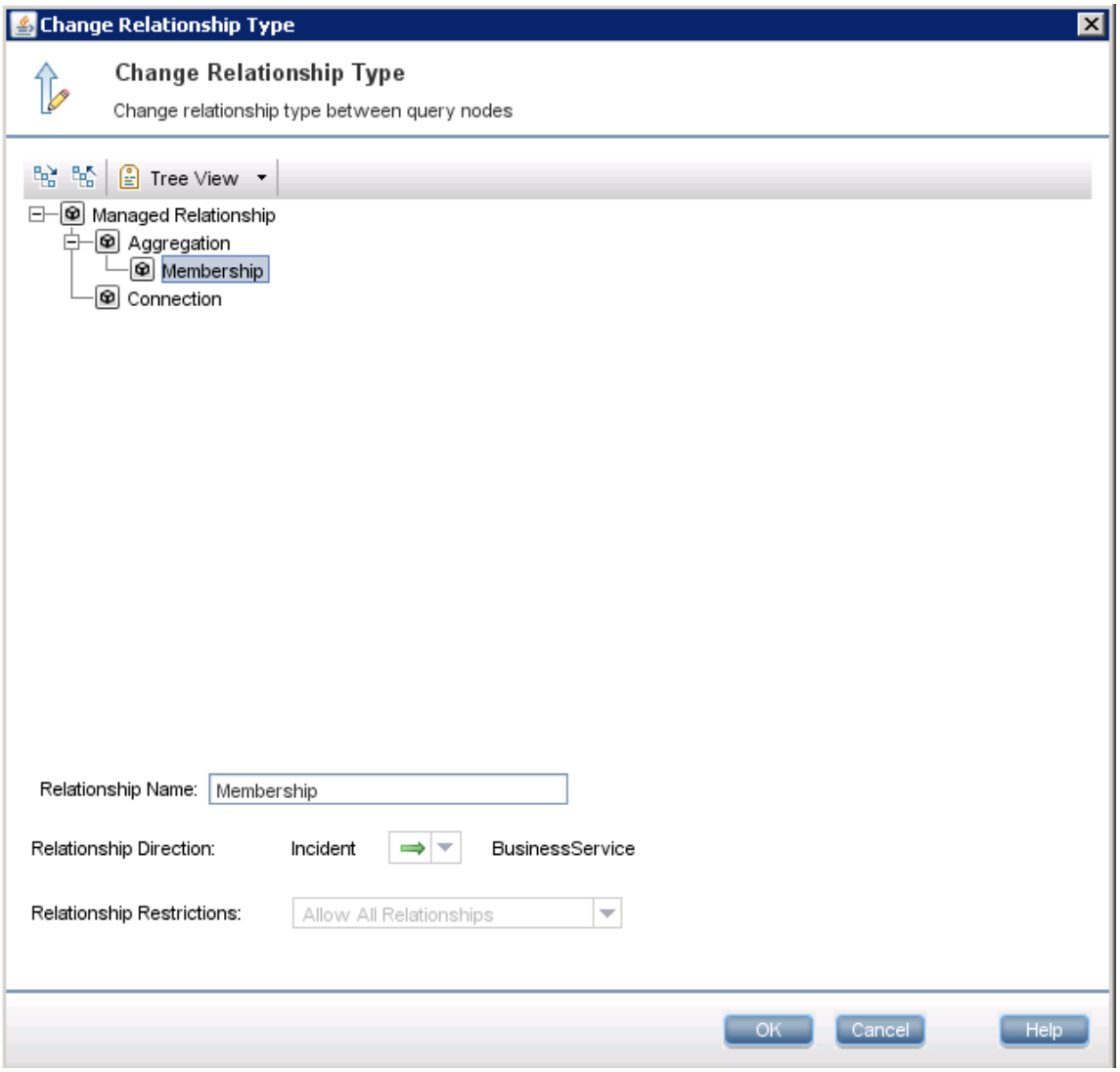


7. Click the **Create Relationship** icon  .

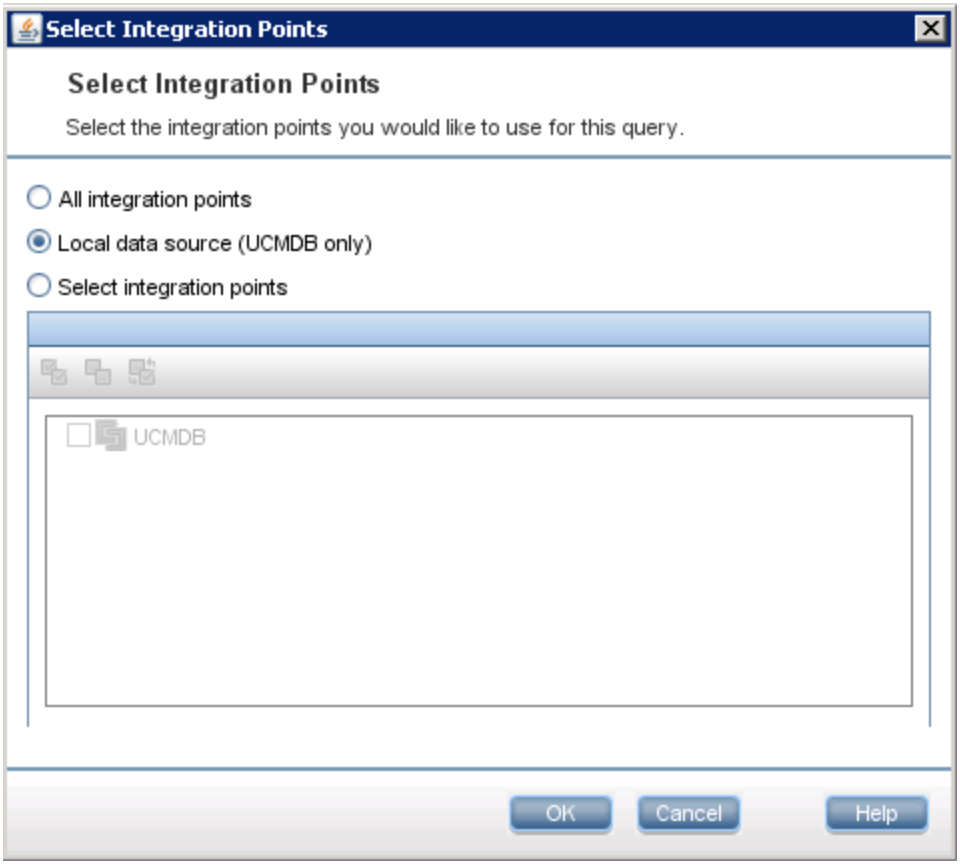
8. Select the Incident query node, and drag the arrow from this node to the BusinessService node to create a regular relationship between the nodes.
   a. Select **Regular Relationship**, and click **OK**.

   b. Select **Membership**, and optionally enter a relationship name (for example, **Membership**). Click

OK.

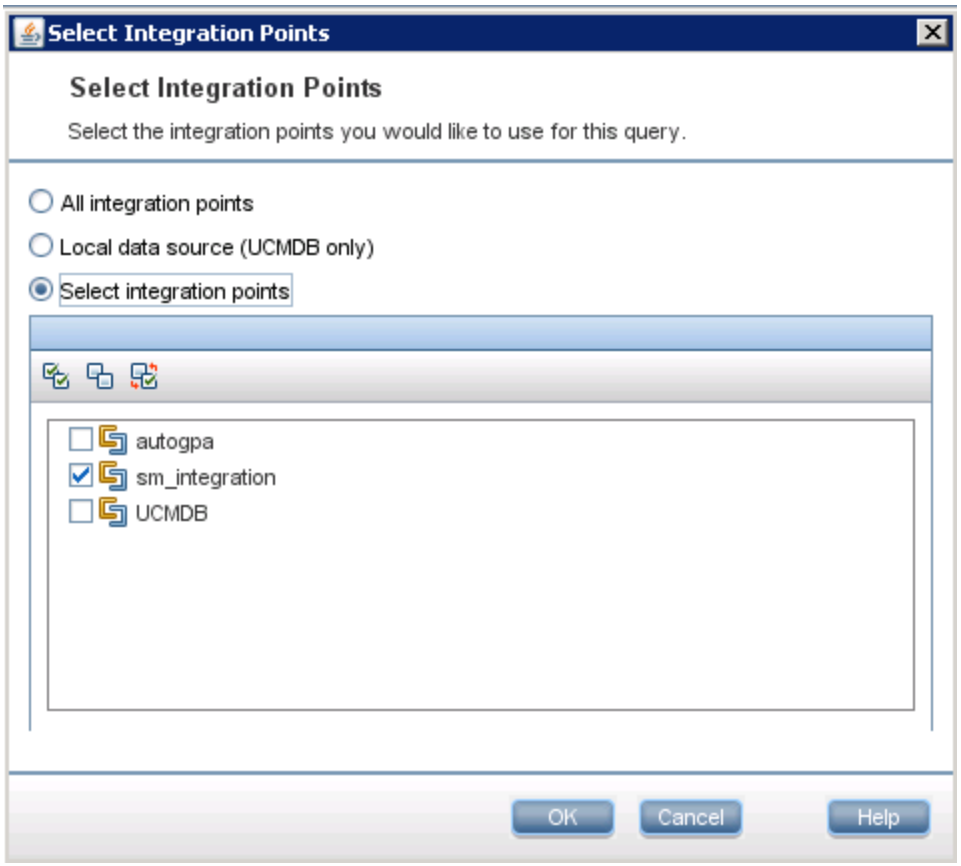


9. Specify UCMDB as the data source for the BusinessService query node. To do this, follow these steps:
   a. Select the **BusinessService** query node.
   b. On the lower right pane, click the **Data Sources** tab and then click **Edit**.
   c. Make sure that the **Local data source (UCMDB only)** option is selected.

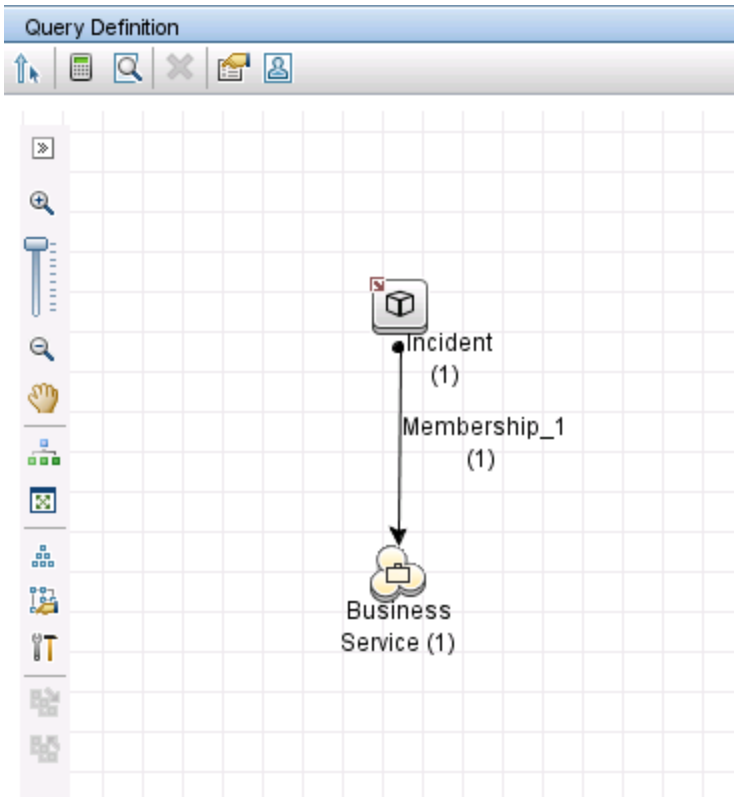


d. Click **OK**.

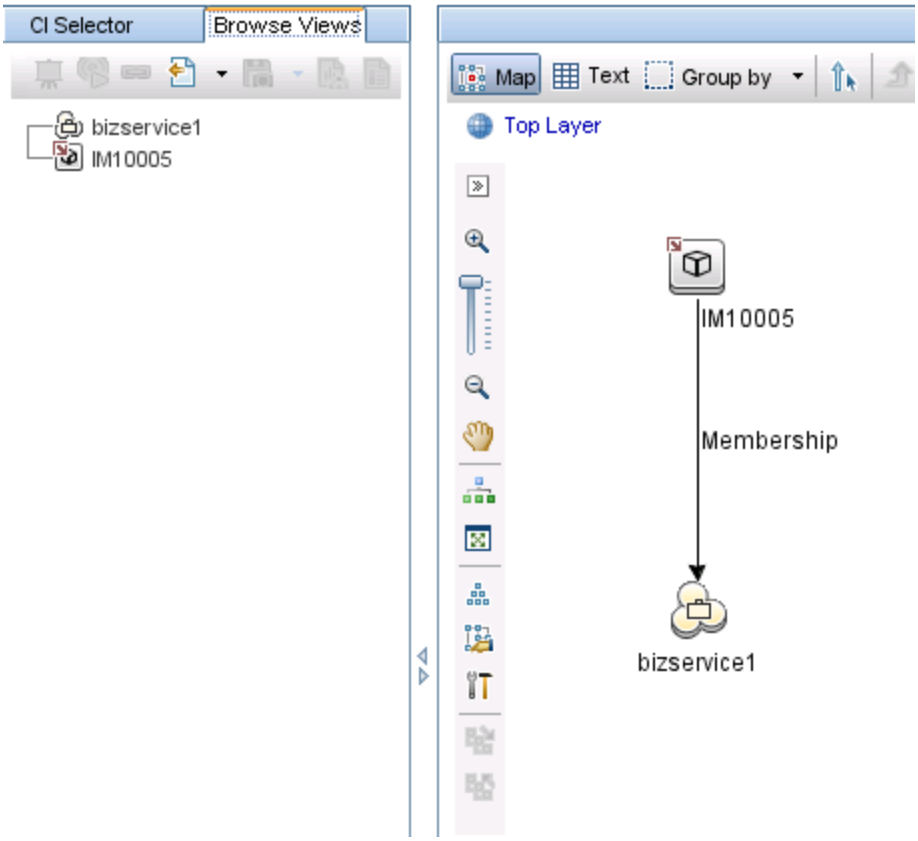
10. Repeat the steps above to specify your integration point as the data source for the Incident query node (for example, **sm_integration**).



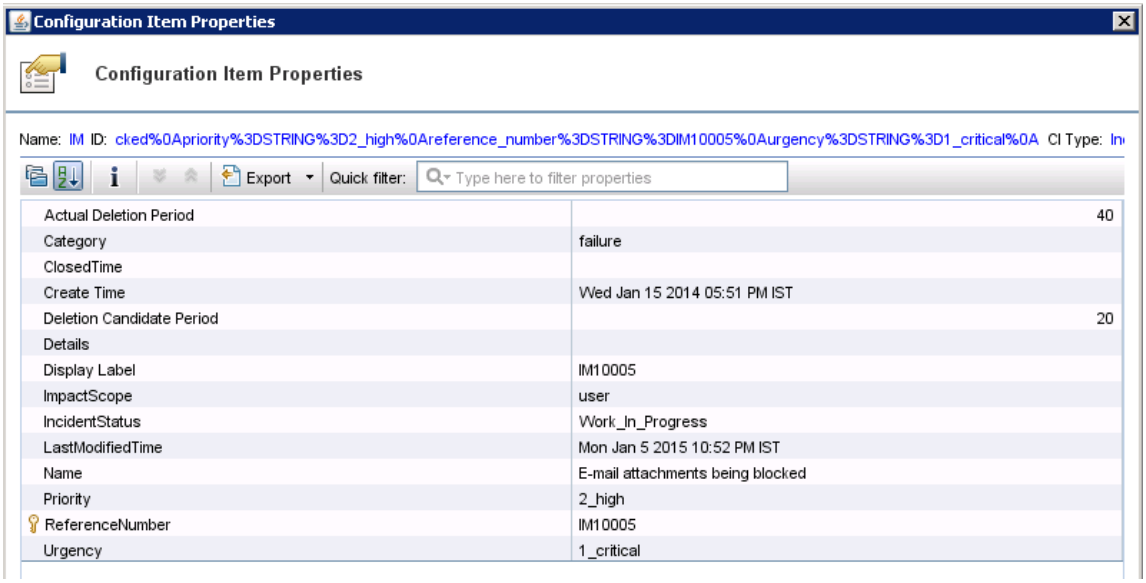
11. Click the **Calculate Query Result Count** icon  . The number of SM Incidents and the number of their affected UCMDB Business Service CIs are displayed.



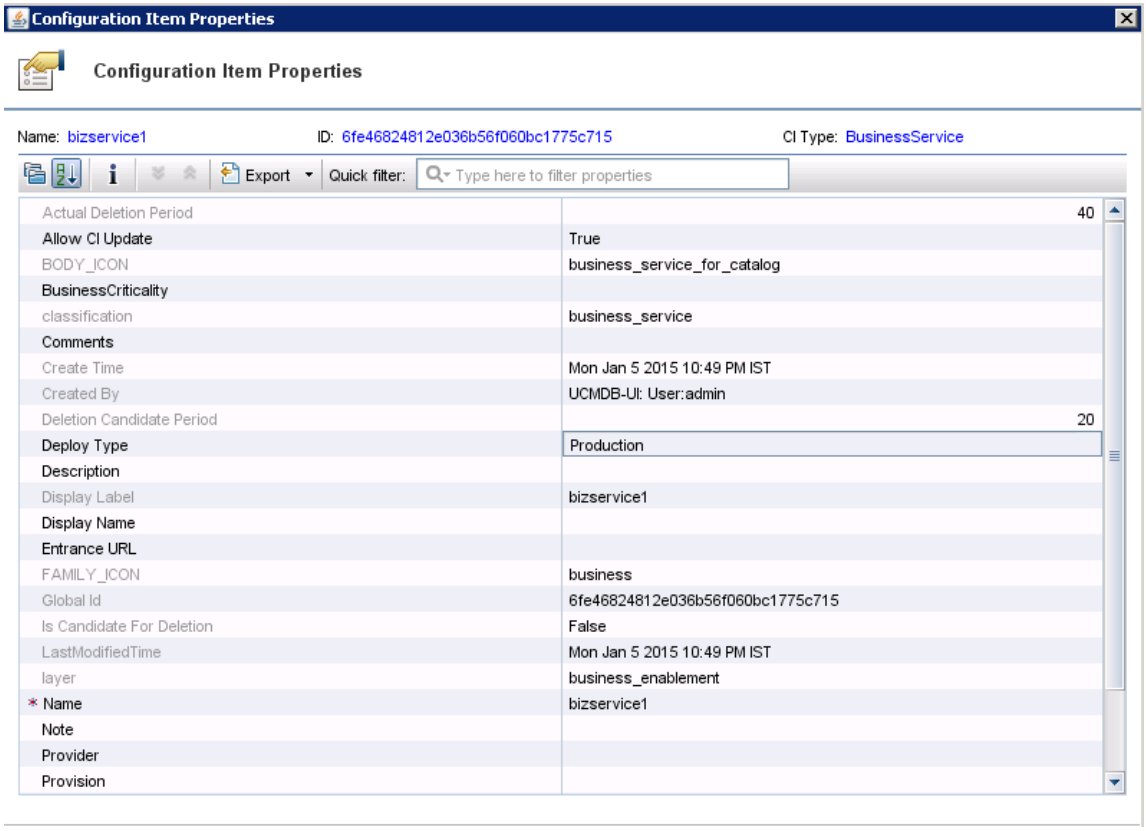
12. Click the **Preview** icon to view the query results  .



13. Select each SM Incident record from either the CI Selector pane or the query pane, and click the **Properties** icon to view its details  .



14. Select each UCMDB CI record from the CI Selector pane, and click the **Properties** icon to view its details.



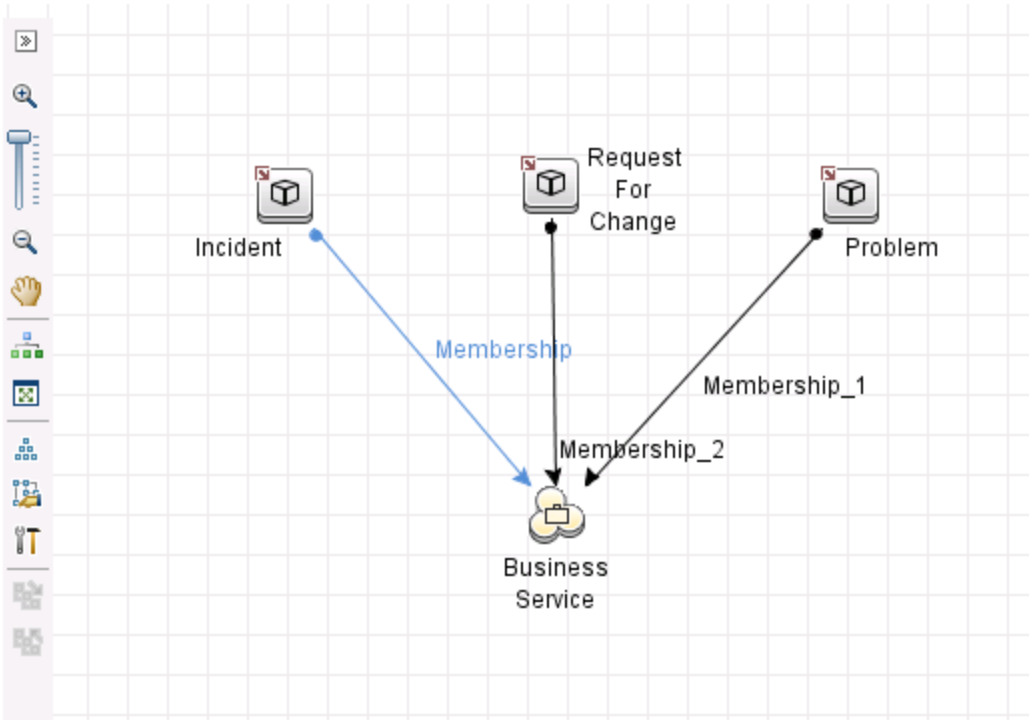
Example 3: Federate Incident, Change, and Problem Record Data from Service Manager for UCMDB CIs

The following example illustrates how to retrieve information of Incident, Change and Problem records in Service Manager that affect a UCMDB Business Service CI.

1. Log in to UCMDB as an administrator.
2. Navigate to **Modeling** > **Modeling Studio** > **Resources**.
3. For Resource Type, select **Queries** from the list.
4. Click **New** > **Query**.
5. On the CI Type tab, go to **ConfigurationItem** > **BusinessElement** > **Service** > **BusinessService**, and drag it to the query pane on the right side.

6. Go to **ItProcessRecord**, and drag **Incident**, **Problem**, and **RequestForChange** to the query pane.

7. Click the **Create Relationship** icon  to create regular relationships between the BusinessService node and the other nodes, as shown in the following figure.



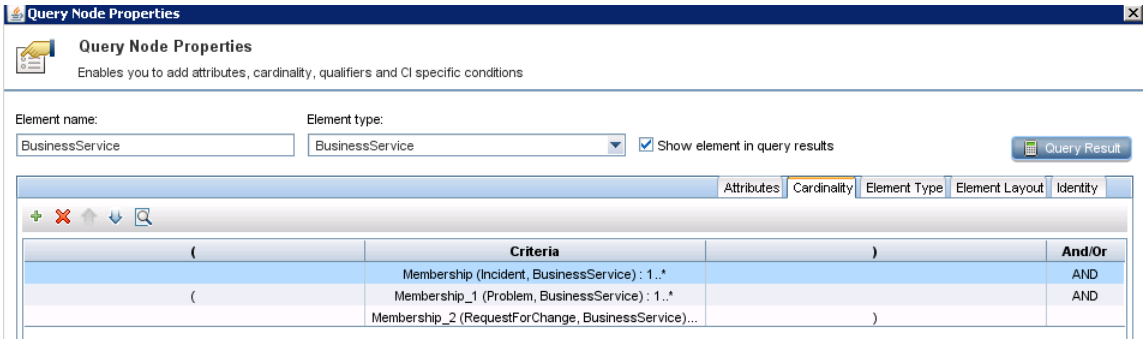
8. Select each node and click the **Data Sources** tab to specify a data source for each node.
   a. For the BusinessService node, specify **UCMDB** as the data source.

   b. For the Incident, Problem, and RequestForChange nodes, specify your integration point as the data source (**sm_integration** in this example).

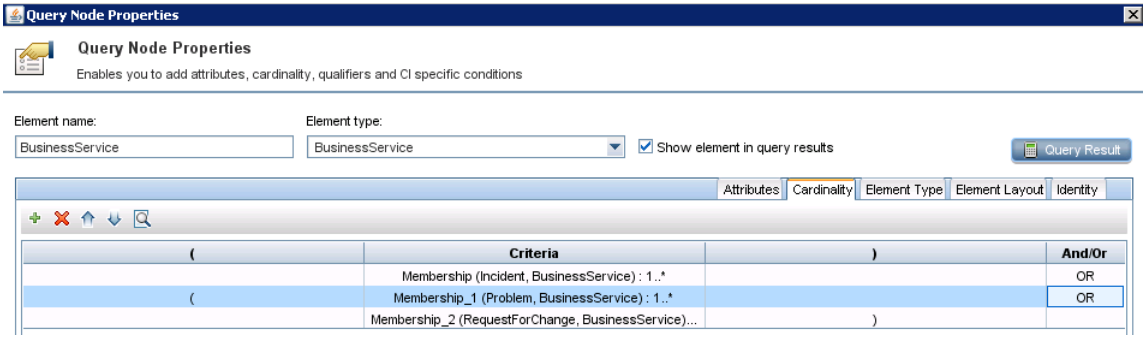
9. Save the query.

10. Optionally, edit the BusinessService node properties as needed.
    a. Select the BusinessServic node, and click **Edit** on the lower right pane.

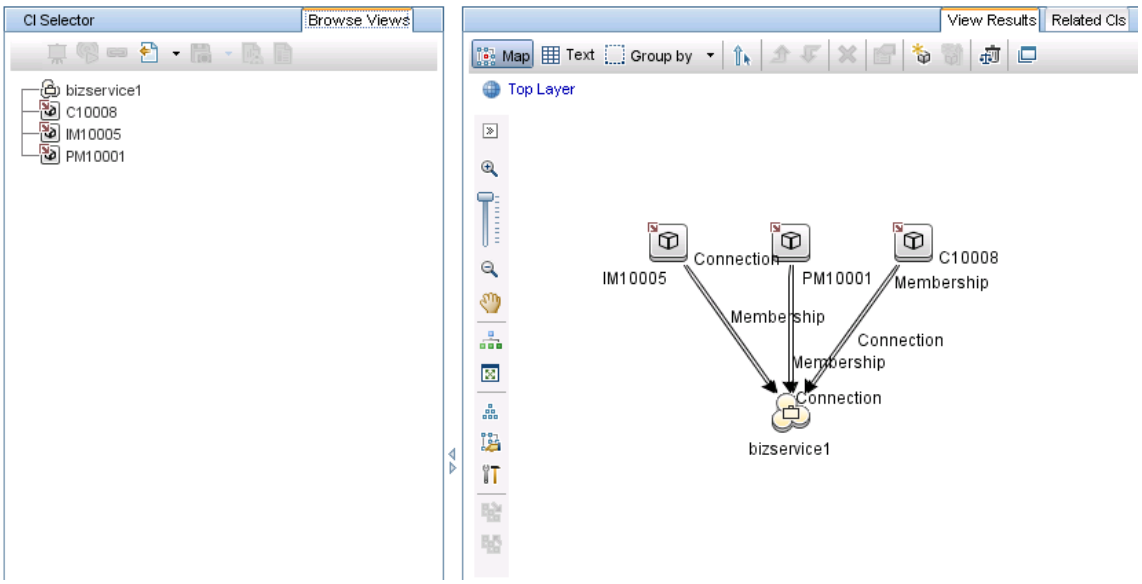
    b. Click the **Cardinality** tab. The default Cardinality setting is displayed.



c. If you wish, change either or both of the **AND** operators to **OR**. This will change the filter criteria and therefore the query results.



11. Click the **Preview** icon  to view the query results.



12. Select each SM Incident record from either the CI Selector pane or the query pane, and click the **Properties** icon  to view its details.

13. Select each UCMDB CI record from either the CI Selector pane or the query pane, and on the **Related CIs** tab click **Show Related CIs** to view its related CIs in both SM and UCMDB.

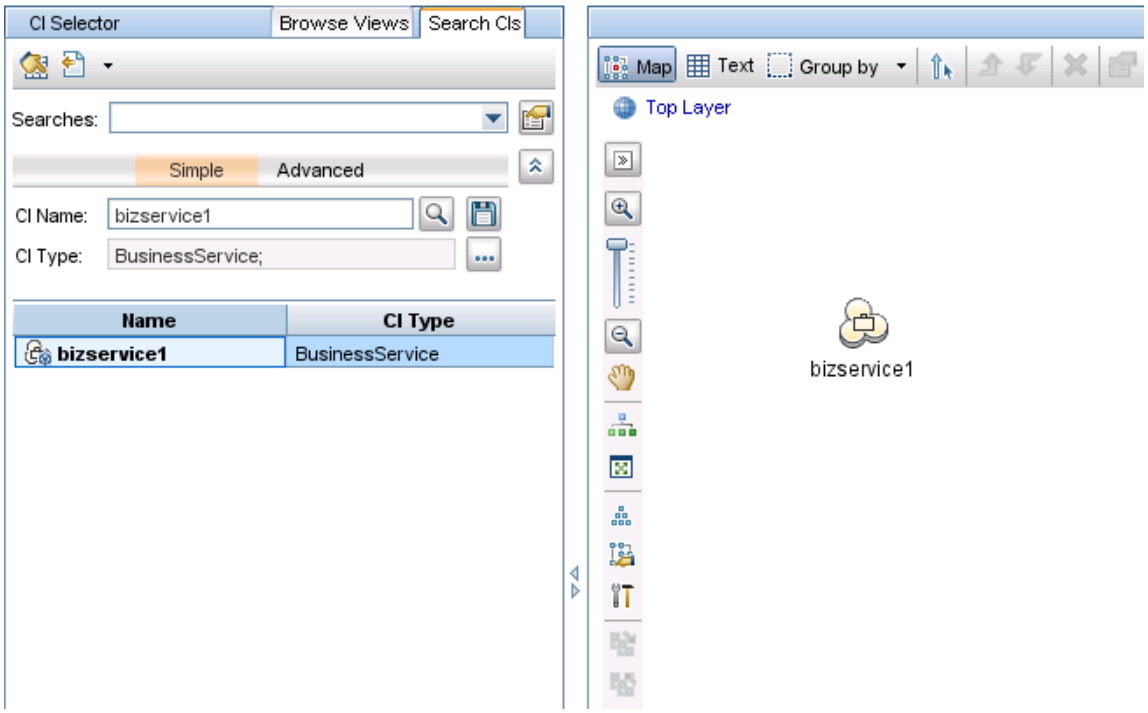
Example 4: Retrieve Service Manager Records Related to a UCMDB CI

The following example illustrates how to retrieve Service Manager records related to a UCMDB CI by using the Get Related CIs functionality.

1. Log in to UCMDB as an administrator.

2. Navigate to **Modeling** > **IT Universe Manager**.

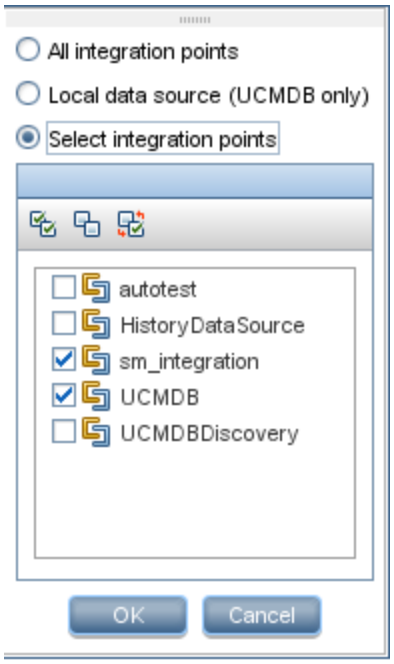
3. On the **Search CIs** tab, search for a UCMDB CI that has related records in Service Manager. For example, enter `bizservice1` in the CI Name field, click **Search**, and then double-click the CI to open it.



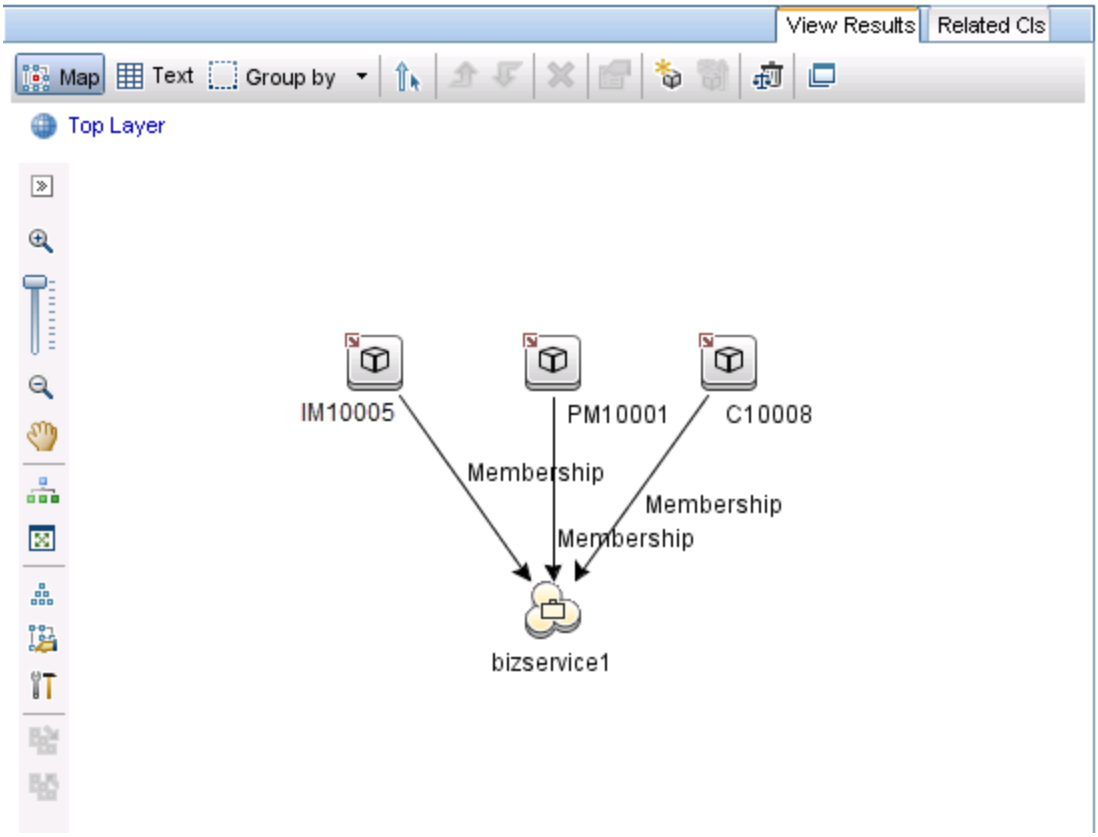
4. If the Get Related CIs pane is not displayed, click the **Show Get Related CIs pane** icon  .

5. Click the **Select target Integration Points for related CIs** icon  .

6. Select the **Select integration points** option, and then select both **UCMDB** and your integration point. Click **OK**.



7. Click **Show Related CIs**. The CI's related SM records and UCMDB CIs are displayed.



8. Select each SM record from the query pane, and click the **Properties** icon  to view its details.

Chapter 3: Multi-Tenancy (Multi-Company) Setup

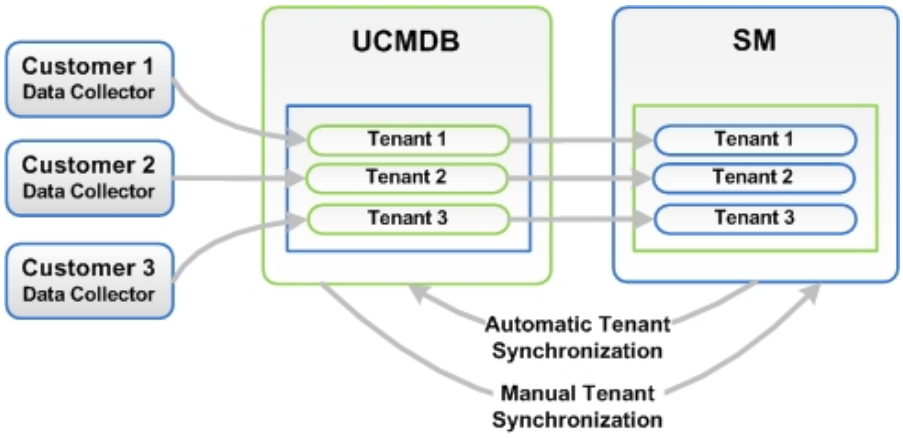
The UCMDB-SM Integration supports a multi-tenancy configuration in which both the Service Manager and UCMDB systems track Configuration Items (CIs) and Configuration Item Relationships (CIRs) by company ID. In a multi-tenancy configuration, you can tailor the integration so that each tenant only sees and works with the CIs and CIRs that match their company ID. Multi-tenancy is intended for managed service providers (MSPs) who wish to offer Configuration Management as a service to multiple tenants.

This chapter includes:

Multi-Tenancy (Multi-Company) Support

Multi-tenancy is when a single instance of software runs on a server, serving multiple client organizations (also referred to as tenants). Multi-tenancy contrasts with a multi-instance architecture where separate software instances or hardware systems are set up for different client organizations.

When implementing a multi-tenant architecture, a software application is designed to virtually partition its data and configuration so that each client organization works with a customized virtual application instance. The following figure illustrates an example multi-tenant integration deployment.



Every tenant configured in UCMDB works with the relevant tenant in SM. If UCMDB did not configure tenants, the tenant configuration must be activated in order to transfer the configuration from SM to UCMDB automatically. This function is performed once only by the system administrator.

In the event that UCMDB tenant configuration already exists and the SM configuration does not exist, SM tenants must be manually configured according to the UCMDB configuration.

Implementing Multi-Tenancy in the UCMDB-SM Integration

SM stores the company records that describe each tenant in the multi-tenant configuration. The Service Manager system is the definitive source for company records and pushes all new company IDs to the UCMDB system creating the equivalent entity in UCMDB.

SM tracks the company ID of each CI and relationship in a multi-tenant configuration. CI records inherit the company ID of the UCMDB feeder that discovered them. Relationship records inherit the company ID of the parent CI in the relationship.

Mandanten SM Security Layer

Mandanten is an SM software layer that is used to filter the customer ID from the CI information. SM uses the Mandanten to ensure that operators only see CI and relationship records where the CI's company ID matches the operator's company ID. If the view is restricted with Mandanten, then Service Manager also restricts the view to all other related records such as change requests and incidents.

What Multi-Tenant Information is Stored in UCMDB

Your UCMDB system stores a company ID attribute for each CI and CIR. The company ID determines what adapter and synchronization schedule your UCMDB system uses to update CI data. Each CI and relationship record can only have one company ID. The UCMDB system obtains a company ID from the Service Manager system.

If more than one tenant (company) shares the same CI, each tenant has their own unique CI record describing the CI. In effect, the UCMDB system creates multiple CI records to track one managed asset. Each tenant’s CI record is unique to that tenant and lists the company's unique company ID.

What Multi-Tenant Information is Stored in Service Manager

Your Service Manager stores the company records that describe each tenant in the multi-tenant configuration. The Service Manager system is the definitive source of company IDs and pushes new and updated information to your UCMDB system.

Service Manager tracks the company ID of each CI and relationship in a multi-tenant configuration. CI records inherit the company ID of the UCMDB feeder that discovered them. Relationship records inherit the company ID of the parent CI in the relationship.

In a best practices implementation, Service Manager uses Mandanten to ensure that operators only see CI and relationship records where the CI's company ID matches the operator's company ID. If you restrict the view with Mandanten, then Service Manager also restricts the view to all other related records such as change requests and incidents.

Unique Logical Names

Service Manager requires that all CIs have unique logical names. If the logical name generation process produces a duplicate logical name value, Service Manager appends an underscore and a number to the end of logical name to make it unique. For example, if two CIs would have the logical name **mytesthost**, then the second CI will instead have the name **mytesthost_1**. A second duplicate CI would have the name **mytesthost_2**.

Synchronization of Company Records

If your system meets all the conditions for multi-tenancy support, Service Manager creates a schedule record to push the company ID of the company record to your UCMDB system. Service Manager uses the following rules to determine whether to push the company ID to your UCMDB system.

Conditions where Service Manager synchronizes company ID with UCMDB

Conditions	Tenant information synchronized?	Schedule record created and action taken in UCMDB
• UCMDB-SM integration **enabled** • Multi-company mode **enabled** in Service Manager • You create a new company record in Service Manager	Yes	Synch Company with UCMDB - *<UCMDB Company ID>* • Add new company ID
• UCMDB-SM integration **enabled** • You update an existing company record that has not been synchronized with UCMDB • Multi-company mode **enabled** in Service Manager	Yes	Synch Company with UCMDB - *<UCMDB Company ID>* • Add new company ID
• UCMDB-SM integration **enabled** • Multi-company mode **enabled** in Service Manager • You disable the option to show a company in multi-company lists on a company synchronized with UCMDB	Yes	Inactivate Company with UCMDB - *<UCMDB Company ID>* • Inactivate existing company ID
• UCMDB-SM integration **enabled** • Multi-company mode **enabled** in Service Manager • You select the option to resynchronize with UCMDB on an existing company record	Yes	Synch Company with UCMDB - *<UCMDB Company ID>* • Add new company ID
• UCMDB-SM integration **enabled** • Multi-company mode **enabled** in Service Manager • You enable the option to show a company in multi-company lists for an inactivated company	Yes	Synch Company with UCMDB - *<UCMDB Company ID>* • Reactivate company ID
• UCMDB-SM integration **disabled** • Multi-company mode **enabled** in Service Manager • You update an existing company record that has already been synchronized with UCMDB	No	None

Conditions where Service Manager synchronizes company ID with UCMDB, continued

Conditions	Tenant information synchronized?	Schedule record created and action taken in UCMDB
• UCMDB-SM integration **disabled** • Multi-company mode **enabled** in Service Manager • You create a new company record in Service Manager	No	None
• UCMDB-SM integration **enabled** • Multi-company mode **enabled** in Service Manager • You disable the option to show a company in multi-company lists on a company not synchronized with UCMDB	No	None
• UCMDB-SM integration **enabled** • Multi-company mode **disabled** in Service Manager • You create a new company record in Service Manager	No	None

UCMDB Customer ID

When you enable the multi-tenancy integration, Service Manager displays a new field in each company record called UCMDB Customer ID. In order to synchronize a company record with UCMDB, you must first provide a value for this field. After you provide a UCMDB Customer ID value this field becomes read-only. You cannot change a company's UCMDB Customer ID after you set it.

This field only accepts numeric data up to ten characters long. Service Manager requires the field value to be a unique positive whole number. You cannot enter duplicate values or use decimals, negative numbers, or zero.

Your UCMDB system automatically uses the UCMDB customer ID of 1 when running in single tenant mode. You can reuse this default value in your multi-tenant implementation by assigning a Service Manager company to have this UCMDB customer ID value. Out-of-the-box, no Service Manager company has the UCMDB customer ID of 1.

UCMDB User ID and Password

When you enable the multi-tenancy integration, Service Manager displays two new fields in each company record called UCMDB UserId and UCMDB Password. These fields allow you to specify the connection information you want Service Manager to use when requesting information for the Actual State section. Any user name and password you enter in these fields must be valid for your UCMDB system.

The user name and password you provide in the Company Information record takes precedence over the user name and password you provide in the System Information record. This allows managed service providers to control access to the UCMDB system on a tenant-by-tenant basis. If you do not provide a company-specific UCMDB user name and password, Service Manager uses the credentials you provided in the System Information record.

Company Code

The multi-tenancy integration requires that each company record has a unique Company Code (company field) value. Since Company Code is a required field, your existing company records should already have Company Code values. However you should ensure that each company record has a unique Company Code value.

When using Service Manager Enhanced Generic Adapter for the UCMDB-SM integration, you must define your company codes in the ServiceManagerEnhancedAdapter9-x/mappings/scripts/SMUtils.groovy file using the following format (where a comma is used to separate mapping entries):

["<*UCMDB Company 1 Code*>":"<*SM Company 1 Code*>", "<*UCMDB Company 2 Code*>":"<*SM Company 2 Code*>"]

For example: ["1":"hp", "2":"sap"]

Caution: You should not change the Company Code value after you have enabled the multi-tenancy integration because this will cause your Service Manager data to become out of synch.

CI Reconciliation Rules

When multi-tenancy is enabled, Service Manager only reconciles the CIs whose company ID matches the company ID in the data push job. For example, when pushing CIs from company 2, the reconciliation rules only apply to the Service Manager CI records that have the company code corresponding to company number 2.

Company Information Pushed to CI and CI Relationship Records

When you enable the multi-tenancy integration, Service Manager inserts the SM Company Code value in CI and relationship records during data push. Service Manager uses the UCMDB Customer ID to look up the matching SM Company Code value.

Company Information Replicated to Incident Records

When you enable the multi-tenancy integration and select the option to create incidents when UCMDB discovers new, updated, or deleted CIs, Service Manager inserts the SM Company Code value in the incident record during replication. Service Manager uses the UCMDB Customer ID to look up the matching SM Company Code value.

Schedule Records

Service Manager uses the **problem** schedule process to manage the synchronization of company IDs to your UCMDB system. You can manually enable the **problem** schedule process from the System Status form.

When the synchronization criteria are met as described in Table *"Conditions where Service Manager synchronizes company ID with UCMDB" on page 71*, Service Manager creates a "Synch Company with UCMDB - *<UCMDB Company ID>*" schedule record (for example, "Synch Company with UCMDB - 1234567890"). If you inactivate a company, Service Manager creates an "Inactivate Company with UCMDB - *<UCMDB Company ID>*" schedule record (for example, "Inactivate Company with UCMDB - 1234567890"). The **problem** schedule process processes the new schedule record on the next background process iteration.

If your Service Manager system cannot connect to your UCMDB system for some reason, it will reschedule the company synchronization at the next scheduled interval (the out-of-the-box interval is 5 minutes). The problem schedule process updates the schedule record with the status rescheduled. If the Service Manager system receives any other error message while connecting to the UCMDB system, it updates the schedule record with the status "application failed due to error - check msglog for possible messages."

Tenant-Specific Discovery Event Manager (DEM) Rules

You can implement the condition field function in order to create SM DEM rules that are specific to a particular tenant in a multi-tenancy UCMDB-SM integration.

Tenant rules vary according to SM tenant configuration requirements, for each record information type pushed from UCMDB to SM different tenants can configure different DEM tenant rules.

Each tenant can have its own set of unique requirements and therefore may implement different processes through the integration.

One tenant may require the addition of CIs directly to SM while another tenant may require opening changes for each CI.

The following table shows a sample set of DEM rules that illustrate how to accomplish this.

Tenant-specific DEM rules

DEM rule ID	Action on new CI	Condition
ucmdbNode_advantage	Add CI	company in $L.file=”advantage”
ucmdbNode_hp	Create change	company in $L.file=”HP”

Note: DEM Rules

When creating DEM rules make sure to create separate DEM rules for each tenant.

Multi-Tenancy Use Cases

The following table describes the necessary actions to perform in various deployment situations to address multi-tenancy issues.

Multi-tenancy use cases

Deployment Integration Type	Description
UCMDB with multi-tenancy rules SM without multi-tenancy rules	When implementing a UCMDB-SM deployment that has existing multi-tenancy rules in UCMDB and does not have multi-tenancy rules configured in SM, the user creates multi-tenancy rules in SM manually and according to the rules in UCMDB.

Multi-tenancy use cases, continued

Deployment Integration Type	Description
SM with multi-tenancy rules UCMDB without multi-tenancy rules	When implementing a UCMDB-SM deployment that has existing multi-tenancy rules in configured SM and does not have multi-tenancy rules configured in UCMDB, the user creates multi-tenancy rules in UCMDB manually as well as according to the rules previously configured in SM.
UCMDB without multi-tenancy rules SM without multi-tenancy rules	When implementing a UCMDB-SM deployment that does not have multi-tenancy rules configured in UCMDB or in SM, the user configures the rules in SM. During the configuration process using the SM multi-tenancy wizard the user can create corresponding tenancy configuration in UCMDB. By creating corresponding tenancy configurations in SM the user also creates a corresponding tenant in UCMDB.

Multi-Tenancy Requirements

Your system must meet the following requirements in order for the integration to support multi-tenancy.

- HP Universal CMDB version 8.02 or later
- HP Service Manager version 9.20 or later
- Integration enabled between UCMDB and Service Manager
- Multi-company mode enabled on the Service Manager system
- Problem schedule process running on the Service Manager system

For additional information about the multi-tenancy integration, you can visit the HP Software Support Online web site or refer to the Service Manager help.

Setting up the Multi-Tenancy Integration in UCMDB

You need to perform the following tasks in UCMDB to set up the multi-tenancy integration.

1. Install a separate data flow probe for each tenant the integration will support.
   See "How to Install a Separate Data Flow Probe for Each Tenant" below.

2. Start tenant-specific data flow probes.
   See "How to Start Tenant-Specific Data Flow Probes" on page 79.

3. Configure IP address ranges for tenant-specific data flow probes.
   See "How to Configure IP Ranges for Tenant-Specific Data Flow Probes" on page 79.

Caution: Multi-tenancy is not supported for population when the integration uses the Service Manager Enhanced Generic Adapter.

How to Install a Separate Data Flow Probe for Each Tenant

If you plan to support a multi-tenant configuration, you must install a separate data probe for each tenant. Out-of-the-box, the UCMDB installer only installs one data flow probe and service.

To install additional data flow probes and start them from your operating system command prompt:

1. Log in to the host of your UCMDB system as an administrator.

2. Insert the HP Universal CMDB Setup Windows DVD into the system disc drive.

3. Start the Data Flow Probe installer (HPUCMDB_DataFlowProbe_x.xx.exe).

4. Follow the on-screen instructions to complete the wizard, but use the following values for each data flow probe you wish to install.
   a. Type a unique path for each installation folder.

   b. Use the same UCMDB application server address for each data flow probe.

   c. Type a valid data flow probe address.

   d. Type a unique name for each data flow probe identifier.

e. Create a unique customer Data Flow Probe domain for each probe (Clear the Use Default CMDB Domain option).

f. Use the same probe gateway and probe manager settings for each probe (for example, use combined or separate processes).

See the *HP Universal CMDB Deployment Guide* for complete installation instructions.

5. Repeat step 3 and step 4 for each data flow probe you wish to install.

6. Open the probe's `DiscoveryProbe.properties` file in a text editor. By default, this file is located in the following folder:
`<UCMDB installation folder>\<data flow probe installation folder>\conf`

For example, `C:\hp\UCMDB\DataFlowProbe\conf`.

Note: The <data flow probe installation folder> must be unique for each tenant.

7. Edit the following properties in the configuration file.
Discovery Probe properties set for each tenant

Property	Value
serverName	Verify the name of the UCMDB server
customerId	Type the customer ID for the tenant this data flow probe supports
appilog.collectors.probe.name	Verify the probe name is unique such as server + tenant ID
applilog.collectors.domain	Verify the domain name of the data flow probe
appilog.collectors.local.ip	Verify the data flow probe gateway name
appilog.collectors.probe.ip	Verify the data flow probe manager name
appilog.collectors.rmi.port	Type a unique port for each probe
appilog.collectors.rmi.gw.port	Type a unique port for each probe
appilog.collectors.probe.html.port	Type a unique port for each probe
appilog.collectors.local.html.port	Type a unique port for each probe

Discovery Probe properties set for each tenant , continued

Property	Value
appilog.collectors.ProbeUseSpecificRMIPortFrom	Type a unique port for each probe or type 0 to have the system automatically select it
appilog.collectors.bigBrother.port	Type a unique port for each probe

8. Save the configuration file.
9. Repeat step 6 to step 8 for the data flow probe of each tenant.

How to Start Tenant-Specific Data Flow Probes

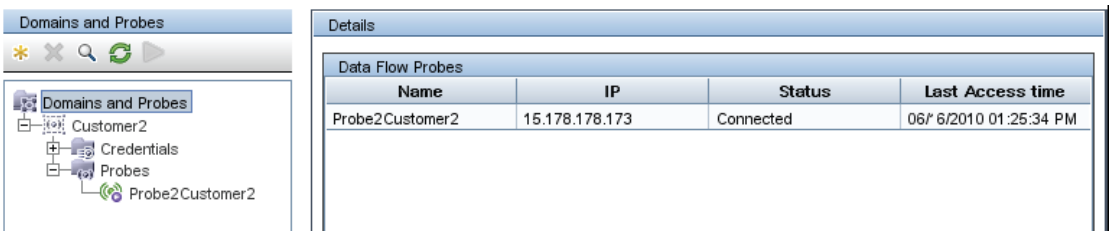
To start tenant-specific data flow probes:

1. Open the OS command prompt and navigate to the probe's bin folder. For example, `C:\hp\UCMDB\DataFlowProbe1\bin`.
2. Type `gateway console`.
3. Repeat step 1 to step 2 for each data flow probe you want to start.

How to Configure IP Ranges for Tenant-Specific Data Flow Probes

To configure IP ranges for tenant-specific data flow probes:

1. Log in to UCMDB as an administrator using the company ID of the tenant whose data flow probe you want to configure.
2. Navigate to **Data Flow Management** > **Data Flow Probe Setup**.
3. Expand the data flow probe domain containing the probe you want to start. For example, **Customer2**.
4. Expand the Probe node and select the data flow probe you want to start. For example, **Probe2Customer2**.



5. Click the **Add IP range** icon .
6. Type an IP range you want the Data Flow Probe to scan. Optionally, add any IP ranges you want to exclude.
7. Click **OK** to save the IP range.
8. Repeat step 1 to step 7 for each data flow probe you want to configure.

Setting up the Multi-Tenancy Integration in Service Manager

You need to perform the following tasks in Service Manager to set up the multi-tenancy integration.

Multi-tenancy support is an optional feature of the integration intended for Managed Service Providers (MSPs) who want to offer Configuration Management as a service to their tenants. In a multi-tenancy configuration, each CI and CIR record has a corresponding company ID. Out-of-the-box, Service Manager allows all operators to view CI data regardless of the company ID. If you wish to restrict access to CI data by company ID, you must enable Mandanten and use the company ID field as a restricting query. See the Service Manager help for more information about multi-company mode and Mandanten.

You must complete the following tasks from your Service Manager system to enable multi-tenancy support for the integration.

1. Start the process schedule.
   See "How to Start the Schedule Process " on the next page.
2. Configure the Service Manager System Information Record.
   See "How to Configure the Service Manager System Information Record" on page 82.
3. Add tenant-specific UCMDB ID and password values to company records (optional).
   See "How to Add Tenant-Specific UCMDB User ID and Password Values" on page 83.

4. Add UCMDB Customer ID values to existing company records.
   See "How to Add UCMDB Customer ID values to Existing Companies" on page 84.

5. Synchronize existing company records with UCMDB.
   See "How to Synchronize Existing Companies from Service Manager to UCMDB" on page 84.

6. Verify that Service Manager synchronized company records with UCMDB (optional).
   See "How to View Whether Company Information Is in UCMDB" on page 85.

7. Resynchronize existing company records with UCMDB (as needed).
   See "How to Resynchronize an Existing Company with UCMDB" on page 86.

8. Inactivate company records you do not want to be part of the integration (as needed).
   See "How to Inactivate a Synchronized Company" on page 87.

9. Reactivate inactive company records you want to be part of the integration (as needed).
   See "How to Reactivate an Inactive Company" on page 87

10. Add tenant-specific DEM rules.
    See "How to Add Tenant-Specific DEM Rules" on page 88.

How to Start the Schedule Process

This integration needs the **problem** schedule process in order to synchronize company records from Service Manager to UCMDB. Make sure that the process is started before you synchronize company records.

To start the **problem** schedule process:

1. Log in to Service Manager as a system administrator.

2. From the System Navigator, click **System Status**.
   A list of the currently started schedules is displayed.

3. Click the **Refresh Display** button to refresh the list.

4. If the **problem** schedule process is not in the list, perform the following steps:
   a. Click the **Start Scheduler** button.

   b. Double-click the **problem** schedule process.

   A message is displayed, indicating that the **problem** schedule process is started.

How to Configure the Service Manager System Information Record

To enable the integration to support multi-tenancy, you must provide additional information in the Service Manager System Information Record.

Caution: In order to enable multi-tenancy support, you must use HP Universal CMDB version 8.02 or greater. Earlier versions of HP Universal CMDB will produce an error message if you attempt to run them in multi-tenancy mode.

To provide additional information in the Service Manager System Information Record:

1. Log in to Service Manager as a system administrator.
2. Navigate to **System Administration** > **Base System Configuration** > **Miscellaneous** > **System Information Record**.
3. Click the **General** tab.
4. Enable the **Run in Multi-Company Mode** option.
5. Click the **Active Integrations** tab.
6. Select the **HP Universal CMDB** option.
   The form displays the UCMDB web service URL field.
7. In the UCMDB web service URL field, type the URL to the CI synchronization web service API. The URL has the following format:
   `http://<UCMDB server name>:<port>/axis2/services/ucmdbSMService`

   Replace *<UCMDB server name>* with the host name of your UCMDB server, and replace *<port>* with the communications port your UCMDB server uses.
8. In UserId and Password, type the user credentials required to manage CIs on the UCMDB system. For example, the out-of-the-box administrator credentials are **admin/admin**.
9. In the Multi-tenant web service URL field, type the URL to the company ID synchronization web service API. The URL has the following format:
   `http://<UCMDB server name>:<port>/axis2/services/UcmdbManagementService`

Replace *<UCMDB server name>* with the host name of your UCMDB server, and replace *<port>* with the communications port your UCMDB server uses.

10. Type the user name and password required to synchronize company IDs on the UCMDB system. For example, the out-of-the-box system administrator credentials for UCMDB are **sysadmin/sysadmin**.
11. Click **Save**. Service Manager displays the message: Information record updated.
12. Log out of the Service Manager system, and log in again with an administrator account.
13. Click **System Status** > **Display Options** > **All Tasks**.
14. Type `k` in the Command field next to the **problem** schedule process and click **Execute Commands**. Wait a few minutes for the **problem** schedule process to close.
15. Click **Start Scheduler**.
16. Double-click the **problem** schedule process. The system now supports multi-tenancy for UCMDB.

How to Add Tenant-Specific UCMDB User ID and Password Values

You can provide a tenant-specific UCMDB user name and password for Service Manager to use when requesting information for the Actual State section. If you provide no credentials, Service Manager uses the credentials in the System Information Record for all tenants.

Note: Any credentials you provide in the company record take precedence over credentials you provide in the System Information Record. The UCMDB UserId and UCMDB Password fields are available only when you have enabled the multi-tenancy integration.

To add tenant-specific UCMDB user name and password:

1. Log in to Service Manager as a system administrator.
2. Navigate to **System Administration** > **Base System Configuration** > **Companies**.
3. Type the search criteria you want to use to find company records. For example, leave the search form blank to search all company records.
4. Click **Search**.

5. Type the user name you want this company to use to connect to UCMDB in the UCMDB UserId field.
6. Type the password for the UCMDB user name in the UCMDB Password field.
7. Click **Save**.
8. Repeat step 3 through step 7 for each company for which you want to provide credentials.

How to Add UCMDB Customer ID values to Existing Companies

You can use the following steps to add a UCMDB Customer ID value to your existing Service Manager company records.

1. Log in to Service Manager as a system administrator.
2. Navigate to **System Administration** > **Base System Configuration** > **Companies**.
3. Type the search criteria you want to use to find company records. For example, leave the search form blank to search all company records.
4. Click **Search**.
5. Type a numeric value in the UCMDB Customer ID field for this company.
6. Click **Save**.
7. Service Manager prompts to confirm that you want to synchronize the record with UCMDB. Click **Yes** if you want to synchronize the company now, or click **No** if you want synchronize the company later.
8. Click **Next** to go to the next company in the record list.
9. Repeat step 5 through step 8 for each company in the record list.

How to Synchronize Existing Companies from Service Manager to UCMDB

Your Service Manager system may already contain company records that you want to use with the multi-tenancy integration.

If you update any field in a company record that has not yet been synchronized to UCMDB, Service Manager prompts whether you want to synchronize the company to UCMDB.

Note: Service Manager will not prompt you to synchronize the company record if you have disabled the option to show the company in multi-company lists, or if there is a pending schedule record associated with the company. See "How to Inactivate a Synchronized Company" on page 87 for more information.

This task includes the following steps:

1. Log in to Service Manager as a system administrator.
2. Navigate to **System Administration** > **Base System Configuration** > **Companies**.
3. Type the search criteria you want to use to find company records. For example, leave the search form blank to search all company records.
4. Click **Search**.
5. Select a company record to update.
6. Update the company record.
7. Click **Save**. Service Manager prompts you to confirm that you want to synchronize the record with UCMDB.

   Note: Service Manager saves the company record regardless of your synchronization choice.

How to View Whether Company Information Is in UCMDB

When you enable the multi-tenancy integration, Service Manager displays a read-only field in each company record that indicates whether or not the UCMDB Customer ID has been synchronized with your UCMDB system.

Note: The UCMDB Customer ID field is visible only when you enable the multi-tenant UCMDB integration.

To view whether company information is in UCMDB:

1. Log in to Service Manager as a system administrator.
2. Navigate to **System Administration** > **Base System Configuration** > **Companies**.
3. Type the search criteria you want to use to find company records. For example, leave the search form blank to search all company records.
4. Click **Search**.
5. Review the status of the **Synched with UCMDB** field.
   If the check box is checked, then Service Manager has already synchronized the company ID with your UCMDB system. If the check box is unchecked, then Service Manager has yet to add this company to your UCMDB system.

   Note: For more information about synchronization failures, see "Schedule Records" on page 74.

How to Resynchronize an Existing Company with UCMDB

Service Manager provides you a means to resynchronize company records with your UCMDB system in case you lose UCMDB data for some reason. For example, you might intentionally remove UCMDB data during integration testing, or you might need to recover data after a disaster. You can force Service Manager to synchronize companies with your UCMDB system with the Re-synch with UCMDB option.

To resynchronize an existing company with UCMDB:

1. Log in to Service Manager as a system administrator.
2. Navigate to **System Administration** > **Base System Configuration** > **Companies**.
3. Type the search criteria you want to use to find company records. For example, leave the search form blank to search all company records.
4. Click **Search**.
5. Select a company record to synchronize.
6. Click the **Re-synch** button next to the **Synched with UCMDB** check box.

Note: The **Re-synch** button is available only from company records that have already been synchronized with UCMDB and have the **Synched with UCMDB** check box checked. If your UCMDB

system already has a company with this ID value, it will ignore the resynchronization request. Service Manager will also ignore a resynchronization request if there is an existing schedule record to resynchronize the company with UCMDB. In this case, it displays the message "`A schedule record has already been added to re-synch this company with UCMDB.`"

How to Inactivate a Synchronized Company

After you have synchronized a company record with UCMDB you can no longer delete the record. Instead, you can inactivate a company record, which causes the UCMDB system to cease all further CI updates for the company. Any existing CI data for the company remains in the UCMDB system associated with the inactive UCMDB Customer ID, but both the company and any associated CIs will no longer be visible from the UCMDB system.

To inactivate a synchronized company:

1. Log in to Service Manager as a system administrator.
2. Navigate to **System Administration** > **Base System Configuration** > **Companies**.
3. Type the search criteria you want to use to find company records. For example, leave the search form blank to search all company records.
4. Click **Search**.
5. Select a company record to inactivate.
6. Select **No** from **Show Company in Multi-Company Lists**.
7. Click **Save**.
8. If this company was previously synchronized with UCMDB, Service Manager prompts you to confirm the inactivation.
9. Click **Yes** to confirm the inactivation or **No** to cancel your changes.

How to Reactivate an Inactive Company

You can reactivate any inactive companies on your Service Manager system to include them in the multi-tenancy integration. You must also synchronize the company with UCMDB for UCMDB to process any CI updates for this company.

To reactivate an inactive company:

1. Log in to Service Manager as a system administrator.
2. Navigate to **System Administration** > **Base System Configuration** > **Companies**.
3. Type the search criteria you want to use to find company records. For example, leave the search form blank to search all company records.
4. Click **Search**.
5. Select a company record to reactivate.
6. Select **Yes** from **Show Company in Multi-Company Lists**.
7. Click **Save**. Service Manager prompts you to reactivate the company with UCMDB.
8. Click **Yes**. Service Manager creates a schedule record to reactivate the company.

How to Add Tenant-Specific DEM Rules

You can use the condition field to create DEM rules that are specific to a particular tenant in a multi-tenancy UCMDB-SM integration. For example, one tenant may want to add CIs directly to Service Manager while another tenant may want to open changes for each CI. The following sample DEM rules illustrate how to accomplish this.

Tenant-specific DEM rules

DEM rule Id	Action on new CI	Condition
ucmdbNode_advantage	Add CI	`company in $L.file="advantage"`
ucmdbNode_hp	Create change	`company in $L.file="HP"`

Tip: It is a best practice to create a separate DEM rule for each tenant.

Chapter 4: Standards and Best Practices

This chapter includes:

UCMDB-SM Configuration Best Practices

This section provides best practices and recommendations for successfully implementing this integration in various environments. This section provides you with valuable understandings and techniques that will enhance the UCMDB-SM integration as well as solve common problems by providing solutions and workarounds to these issues. These practices and recommendations may vary slightly according to each implementation, as the specific system requirements and settings alter per system environment.

This section includes:

CI Name Mapping Considerations

UCMDB allows duplicate CI names while Service Manager requires unique logical names. Before pushing UCMDB CIs, you need to define a correct CI name mapping for them. For example, many UCMDB CIs (such

as CIs of the Running Software, CRG, Switch, or Router type) have the same display label.

To prevent duplicate CI names from occurring in Service Manager when pushing UCMDB CIs, the following mappings are provided out-of-the-box:

CRG mapping

Out-of-the-box, UCMDB CRG records are mapped to Service Manager as follows:

- If a Cluster exists for a CRG, the CRG is mapped to this CI logical name: <Cluster display label>_<CRG display label>;
- If the CRG does not have a Cluster, but has several IP addresses, the CRG is mapped to the following (where the IP addresses are sorted alphabetically):
  <IpAddress1>_..._<IpAddressN>.<authoritativeDnsName>_<CRG display label> (when IpAddress.authoritativeDnsName exists)

  <IpAddress1>_..._<IpAddressN>_<CRG display label> (when IpAddress.authoritativeDnsName does not exist)
- If neither a Cluster nor an IP address exists for the CRG, it is mapped directly to <CRG display label>.

Running Software mapping

Running Software CIs are prefixed with their root container node display label when mapped to a Service Manager CI: <Node display label>_<Running Software display label>.

Switch and Router mapping

Switch or Router type CI records in UCMDB are prefixed with their MAC addresses when mapped to a Service Manager CI: <MACAddress1>_..._<MACAddressN>_<Switch or Router display label>, where the MAC addresses are sorted alphabetically.

Bi-Directional Data Synchronization Recommendations

The UCMDB-SM integration supports bi-directional data synchronization between UCMDB and Service Manager (SM). HP recommends that you follow the following best practices to avoid unnecessary problems due to improper use of the data push and population features:

- For CIs/CI Relationships that UCMDB can automatically discover, use UCMDB as the data source. Do not make changes to them in Service Manager; instead, always let UCMDB discover their changes and push the changes to SM.

- For CIs/CI Relationships that UCMDB cannot automatically discover, use SM as the data source. Do not make changes to them in UCMDB; instead, always make changes to them in SM and populate the changes to UCMDB.
- For CIs/CI Relationships that are already created in SM and UCMDB can automatically discover, run a one-time population to synchronize them to UCMDB, and then use UCMDB as their data source.

Caution: Problems like the following may occur if you do not follow these best practices:

Problem 1:

[Population Adapter] After CIs/CI Relationships are pushed from UCMDB, if you directly make changes in SM to these records without ever populating them back to UCMDB first, the changes cannot be populated to UCMDB.

Workaround:

Changing these UCMDB records in SM is not recommended; however if you need to do so you can do the following to solve this issue: After the records are pushed to SM, populate them back to UCMDB first before making any changes to them in SM. This way the changes can then be populated to UCMDB.

Problem 2:

[Population Adapter] After a Composition relationship between a Node CI (node 1) and Running Software CI is pushed to SM, if you change the upstream CI of the relationship from node 1 to node 2 and then run a change population to populate this change, the Running Software CI will be removed in UCMDB.

Workaround:

It is recommended that you remove the running software in UCMDB and create a new one instead of directly replacing the container of the running software in SM. If you cannot avoid doing so, do the following:

After you change the upstream CI of the relationship from node 1 to node 2, do not directly run the change population. Follow these steps to avoid this issue:

1. Update the Running Software CI in SM (or simply save it to mark it as updated).
2. Run a Running Software CI change population. This will create node 2 (if it does not already exist in UCMDB) and a new Composition relationship between node 2 and this Running Software CI.

3. Run a delta population to synchronize the relationship change to UCMDB. The relationship between node 1 and the Running Software CI will be removed, and the new relationship created in step 2 will remain.

If you have run a delta population after changing the upstream CI of the relationship from node 1 to node 2, and as a result the Running Software CI has been removed in UCMDB, follow these steps to solve this issue:

1. Update the Running Software CI in SM (or simply save it to mark it as updated).
2. Run a Running Software CI change population. This will create the Running Software CI, node 2 (if it does not already exist in UCMDB) and a new Composition relationship between node 2 and this Running Software CI.

Push Scheduling Recommendations

Push jobs are run using two main methods, the first of which is manually executing the push job and the second is scheduling the push job.

All push jobs can potentially produce a strain on the UCMDB and SM systems; HP recommends that you adhere to the following guidelines.

Scheduler time frames

It is important for you to understand the function of the Scheduler “time frame” concept. Running push jobs creates an increase in system activity and may affect application responsiveness. In order to enable users to effectively interact with applications HP recommends the following guidelines:

In order to reduce system strain, schedule the UCMDB to SM push to run at non-peak usage hours, preferably when system usage is at a minimum.

Scheduler frequency

It is important to be aware of the business requirements when configuring the schedule frequency. The scheduler frequency depends on infrastructure environment changes that must be synchronized between UCMDB and SM.

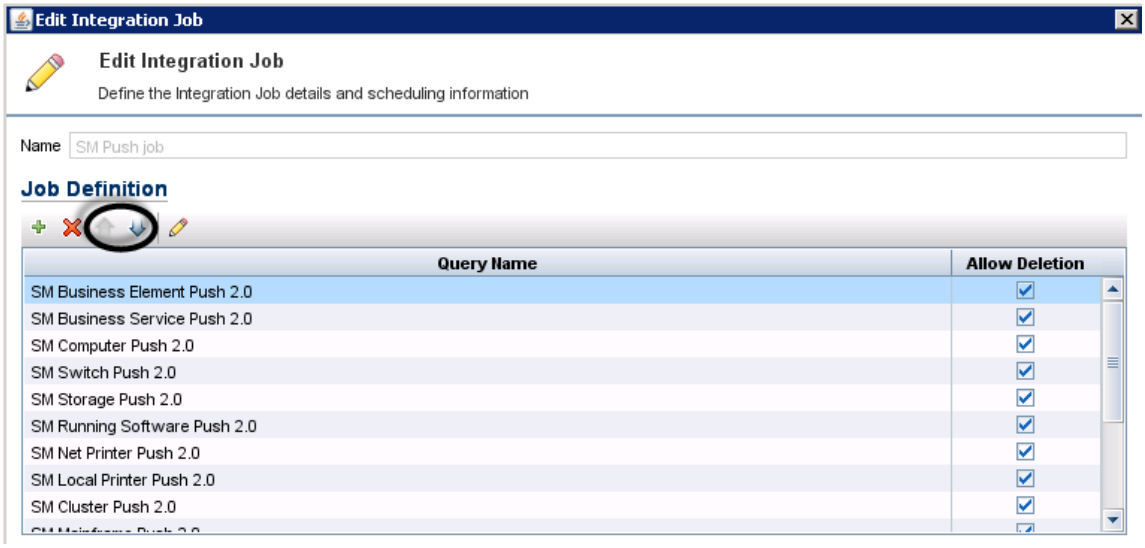
Define the scheduling frequency based on the business requirements for consuming up-to-date CI information. Most implementations require a daily update. When scheduling small IT systems that are prone to frequent changes, the scheduling frequency may need to be increased.

Push Job dependencies

UCMDB Push Jobs do not support dependencies between each other. Each “Push Job” is considered a

separate task and users cannot define job dependencies. For example, that one job is dependent on another or upon completion before the next job is run.

It is important that both CI queries and their dependent Relationship queries exist in the same Job in order to avoid relationships not being pushed to Service Manager. You can change the position of each query in the list to define an appropriate execution sequence. See the following figure for an example.



Push in Clustered Environments

A clustered SM environment is comprised of multiple servlets running in parallel with a load balancer that dispatches user requests to any available servlet. You must configure the UCMDB-SM integration to point to a specific servlet and not to the SM loadBalancer. In order to perform this, you must first create a dedicated web service listener.

This section includes:

- "Dedicated Web Services" below
- "Step-by-Step Cluster Configuration Process" on the next page
- "Connecting to Multiple SM Processes" on page 95

Dedicated Web Services

A Service Manager system configured for vertical or horizontal scaling uses a load balancer to redirect client connection requests to an available SM process. However, most Web Services clients cannot handle a redirect request and will fail if they use the SM load balancer as the endpoint URL.

HP recommends creating one or more SM processes dedicated to Web Services requests. The user must configure the relevant external web service clients to connect directly to the dedicated Service Manager processes.

Step-by-Step Cluster Configuration Process

This section describes the steps to configure a cluster environment for the integration.

How to Configure Web Clients

To configure the relevant external web clients:

1. Stop the Service Manager service.
2. Open the `sm.cfg` file, and create a dedicated SM process to listen for Web Services requests using the `-debugnode` parameter.
   The following entries create a dedicated process listening on ports 13085 and 13445.

```
01 sm -httpPort:13080 -loadbalancer
02 sm -httpPort:13081 -httpsPort:13443
03 sm -httpPort:13083 -httpsPort:13444
04 sm -httpPort:13085 -httpsPort:13445 -debugnode
```

   Explanation

   The code excerpt illustrates the various settings for each of the SM process listeners (web services) that enable SM clients to connect to the SM service.

   Line 01 defines the load balancer port (13080).

   Lines 02 and 03 define the SM ports to which non-dedicated SM clients are redirected by the SM load balancer.

   Line 04 defines the debugnode port that is utilized by the dedicated SM clients.

Note: Debugnode parameter

The debugnode parameter tells the SM load balancer not to forward any client connection requests to this Service Manager process. Only clients that directly connect to the process can access this port.

How to Configure the Debugnode

To configure the debugnode:

1. Start the SM service.
2. Configure any external web service clients to connect directly to the SM processes running in debugnode. When performing an integration using UCMDB, the Service Manager Adapter for SM should be configured to connect to the debugnode port.
   For example, for normal connections set the endpoint URL to:

   `http://<fully qualified host name>:13085/SM/9/rest/<Service Name>`

   and for SSL-encrypted connections set the URL to:

   `https://<fully qualified host name>:13445/SM/9/rest/<Service Name>`

   These clients may include UCMDB (for push purposes), Connect-It, and additional applications.

Connecting to Multiple SM Processes

If you want to improve performance, you can connect to multiple Service Manager processes. The integration supports both Service Manager vertical and horizontal load balancer environments.

You can create more than one SM process that is dedicated to Web Services requests, and configure the **URL Override** field of the integration point with the dedicated SM processes. This field value (if any) overrides the Hostname/IP and Port settings.

The following is an example value of this field, which connects two SM processes:

`http://<fully qualified host name1>:13080/SM/9/rest;http://<fully qualified host name2>:13082/SM/9/rest`

Initial Load Configurations

Before the configuration process can begin, you must first assess the amount of CI and relationship data to be transferred from UCMDB to SM and ascertain the iteration process that is required based on the volume.

You must first assess whether or not all of the data can be pushed in a single iteration. This is ascertained by the amount of data that is included in the push queries and the amount of time you have to push this data.

This section includes:

Note: The performance data presented in this document is based on tests that were performed at HP and is provided for reference only. The integration performance may significantly differ in your environment depending on your hardware configuration.

Push Performance in a Single-Threaded Environment

The Push of 22,500 UCMDB root CIs (roots in queries) and/or Relationships in a single-threaded environment takes about an hour and is performed in a linear fashion. See the following table.

Performance data in a single-threaded environment

Number of root CIs/CI Relationships Pushed per Hour	Multi-Threading Settings in sm.properties
22,500	number.of.concurent.sending.threads=1 min.objects.for.concurent.sending=50 number.of.chunks.per.thread=3 recommended.min.cis.per.chunk=50

To view or edit the sm.properties file in UCMDB, navigate to **Data Flow Management** > **Adapter Management** > **ServiceManagerEnhancedAdapter9-x** > **Configuration Files** > **sm.properties**.

Number of Root CIs and Relationships/22,500

The push time (in hours) in any given environment is calculated as follows:

If the push of a single planned query has the potential of breaching the permitted time frame, the data must be divided into several queries. Each query must be pushed individually.

This query division is performed by creating several queries, each with different node conditions that enable data filtering. Once all queries are pushed for the first time, the Initial Load process is complete.

Note: Applying node conditions

When applying node conditions to the various SM Sync Queries, you must make sure that all of the information is included in the queries, so that all relevant data is copied to SM.

Implementing Multi-Threading

In order to improve performance, the Service Manager adapter utilizes multiple threads for the push of CI and Relationship data to SM. The following section explains these settings and how to configure them for maximum performance.

The multi-threading configuration is defined in the sm.properties file on the UCMDB server. To view or edit the file in UCMDB, navigate to **Data Flow Management** > **Adapter Management** > **ServiceManagerEnhancedAdapter9-x** > **Configuration Files** > **sm.properties**.

The following are example multi-threading definitions in the sm.properties file:

```
01 number.of.concurrent.sending.threads=6

02 min.objects.for.concurrent.sending=50

03 number.of.chunks.per.thread=3

04 recommended.min.cis.per.chunk=50
```

Explanation

The code excerpt illustrates the relevant multi threading settings on the UCMDB server.

- Line 01 defines the number of parallel threads UCMDB will open to SM for CI push. Setting this parameter to 1 disables multi-threading, while a values of 2 or higher enables multi-threading.
- Line 02 defines the minimum number of SM objects needed to use multiple threads as opposed to a single thread.
- Line 03 defines the number of chunks per thread. This number multiplied by the number of threads gives you the total number of CI data chunks.
- Line 04 defines the recommended minimum number of CIs per CI data chunk.

The total number of chunks = number.of.chunks.per.thread * number.of.concurrent.sending.threads

The integration implements a queue mechanism as follows:

The data passed from UCMDB to SM is divided into equal chunks, and these chunks are placed in a queue.

Each available thread pulls the next chunk from the queue until all threads are available. Once this process has completed, the push is complete.

The mechanism is designed to minimize idle time of each thread. As each thread processes its chunk in parallel, some threads may finish before others and it is inefficient for them to wait for each other.

Caution: Defining too many threads

It is ineffective to over-increase the number of threads as this causes the SM server to overload. In enterprise environments where the SM server processing the push data is very robust the number of threads can be increased to 10 and in some cases even 20; however, you must take into account that increasing the number of threads raises CPU usage on the SM server during push, which may reduce application performance.

Push Performance in Multi-Threaded Environments

The push of 60,000 UCMDB root CIs (roots in queries) and/or Relationships in an out-of-the-box multi-threaded environment takes about an hour and is performed in a linear fashion. See the following table.

Performance Data in an Out-Of-The-Box Multi-Threaded Environment

Number of Root CIs/Relationships Pushed per Hour	Multi-Threading Settings in sm.properties (Default)
60,000	number.of.concurrent.sending.threads=6 min.objects.for.concurrent.sending=50 number.of.chunks.per.thread=3 recommended.min.cis.per.chunk=50

The push time (in hours) in any given environment is calculated as follows:

Number of Root CIs and Relationships/60,000

Push Performance in Multiple SM Processes Environments

The Push of 190,000 UCMDB root CIs (roots in queries) and/or Relationships in a multi-threaded environment with multiple SM processes takes about an hour and is performed in a linear fashion. See the following table:

Performance data in a multiple SM processes environment

Number of root CIs/Relationships pushed per hour	SM processes	Multi-threading settings in sm.properties (default)

160,000	2 server hosts, with each host running 4 processes	`number.of.concurrent.sending.threads=60`
		`min.objects.for.concurrent.sending=50`
		`number.of.chunks.per.thread=3`
		`recommended.min.cis.per.chunk=50`
		Data Push Chunk Size = 4000 (in Integration Settings in UCMDB)

For more information about defining multiple SM processes for the integration, see "How to Create an Integration Point in UCMDB" on page 33.

The push time (in hours) in any given environment is calculated as follows:

Number of Root CIs and Relationships/160,000

How to Set up SM DEM Rules for Initial Loads

SM Discovered Event Manager Rules (DEM Rules) enable the user to define the appropriate action to take for each event type that is reported to SM.

Each CI and relationship record pushed from UCMDB to SM is analyzed against the existing SM records and open Change requests. SM rules define the appropriate action to be taken for each type of CI data update that is sent to SM.

To view or update the SM Discovered Event Manager Rules:

1. Log in to Service Manager as a system administrator.
2. Navigate to **Tailoring** > **Web Services** > **Discovered Event Manager Rules**.
3. Press **ENTER** or click the **Search** button.
   A list of all the Discovered Event Manager Rules is displayed. Each rule is usually linked to a CI Type or a subset of CIs of the same type.
4. Click the individual CI Discovered Event Manager Rule to view its details.

To set up DEM Rules for initial loads:

Tip: When performing an Initial Load, HP recommends setting the SM Discovered Event Manager

Rules to add newly reported CIs as described below. This minimizes the “noise” of an Initial Load, which could potentially create tens of thousands of Changes or Incidents.

For each of the Discovered Event Manager Rules, perform the following steps:

1. Select the relevant Discovered Event Manager Rule.
2. Go to the **Action if matching record does not exist** section, select the **Add the record** option.
3. In the **Action if record does not exist but unexpected data discovered** section, select the **Log Results and Update Record** option.
4. In the **Action if record is to be deleted** section, select the **Delete Record** option.
5. Save the Discovered Event Manager Rule record.

How to Configure Differential or Delta Load DEM Rules

Tip: Once the “Initial Load” or “Data Load” of the CI data is completed, HP recommends applying Differential or Delta Load settings. These settings apply to all data loaded from UCMDB to SM.

These loads send only updates regarding modifications discovered in the IT infrastructure from UCMDB to SM.

To set up the SM DEM Rules for Differential or Delta Loads:

1. Log in to Service Manager as a system administrator.
2. Navigate to **Tailoring** > **Web Services** > **Discovered Event Manager Rules**.
3. Press Enter or click the **Search** button.
   A list of all the Discovered Event Manager Rules in SM is displayed.
4. For each of the Discovered Event Manager Rules, perform the following steps:
   a. Select the relevant Discovered Event Manager Rule.
   b. In the **Action if matching record does not exist** section, select the appropriate action required for each newly detected CI. If uncertain, select the **Add the record** option.

c. In the **Action if record does exist but unexpected data discovered** section, select the appropriate action for each CI that was modified, resulting in an unexpected or incorrect result. The recommended best practice is to select the **Open a Change** option.

d. In the **Action if record is to be deleted** section, select the appropriate action required for each CI that was removed. The recommended best practice is to select the **Delete Record** option for CI Relationships, and select the **Update record to the selected status** option for CIs.

e. Save the Discovered Event Manager Rule record.

Fault Detection and Recovery for Push

Universal CMDB has provided a fault detection and recovery mechanism since version 9.05: individual CI failures no longer cause the entire push to fail, and you can review all failed CIs in the Universal CMDB studio and then re-push them.

Duplicate logical.name issue

A typical fault you may encounter is the duplicate logical name issue, which is caused by the use of different unique key fields in Universal CMDB and Service Manager: CI logical.name in Service Manager is unique, and it usually maps to CI display label in Universal CMDB (which is not unique). HP recommends that you follow the following guidelines (listed from the highest to lowest priority) to resolve this issue:

- Make sure that each display label field value in UCMDB is unique;
- If uncertain of the above, in the adapter mapping configuration file avoid direct mapping between Universal CMDB display label and SM logical name;
- Map SM logical name to another Universal CMDB field that is unique;
- Add a prefix or suffix to UCMDB display label values;

Note: Out-of-the-box, SM logical name of Running Software is mapped with a prefix of DNS name:

```
<target_mapping datatype="STRING" name="CIIdentifier"
    value="SMPushFunctions.getCIIdentifier(Root['display_label'],Root.Node*.
    getAt('display_label'))"/>

    public static String getCIIdentifier(String name, def arr){
            if( fIsEmpty(arr) ){ return name;}else{
                return covertArray2String(arr)+name;
            }
```

```
        return name;
}
```

- If you cannot do any of the above, you can use the UCMDB Fault Detection and Recovery mechanism together with the “Duplication Rule” setting of DEM rules as described in the following.

To set up DEM Rules for duplicate logical names:

1. Log in to Service Manager as a system administrator.
2. Navigate to **Tailoring** > **Web Services** > **Discovered Event Manager Rules** > **Duplication Rule** tab.
3. For each of the Discovered Event Manager Rules, perform the following steps:
   a. Go to the **Action if logical name is duplicated** section, and select the **Return Error** option.
   b. Save the Discovered Event Manager Rule record.

Note: After you run a push job, CIs with a duplicate logical name are reported as failed CIs with a duplicate name exception. You can review the failed CIs in the Universal CMDB studio, fix the errors by either changing the data in Universal CMDB or in the adapter mapping configuration file (XML and Groovy), and then re-push the failed CIs.

How to Enable Lightweight Single Sign-On (LW-SSO) Configuration

You can enable LW-SSO for the integration so that users can directly view UCMDB CI records from the Service Manager web client by clicking the **View in UCMDB** button, without providing a UCMDB username and password.

Note: LW-SSO is not supported for the Service Manager Windows client.

To enable LW-SSO for the integration:

1. For each Service Manager user account that needs LW-SSO, create a user account in UCMDB with the same username. The passwords in the two systems can be different.
2. Enable LW-SSO in the Service Manager Web tier. For details, see the *Configure LW-SSO in the*

Service Manager Web tier topic in the Service Manager help.

3. Enable LW-SSO in UCMDB. For details, see the *HP Universal CMDB Deployment Guide*.

Frequently Asked Questions

This section provides answers to frequently asked questions about the UCMDB-SM integration.

This section includes:

When Is a New CI Created in Service Manager?

CIs are created in SM under the following circumstances:

- A CI is manually added to SM through the Configuration Management module.
- UCMDB reports a newly discovered CI according to the following:
  - When a new CI is reported and the Discovered Event Manager Rules are set to **Add the Record**.
  - When a new CI is reported, the Discovered Event Manager Rules are set to **Open an Incident** and the Incident has been closed.
  - When a new CI is reported, the Discovered Event Manager Rules are set to **Open a Change** and the Change has been verified.

Can I Analyze the Reason for a CI Deletion in SM?

No.

SM opens a change request on the deleted CI and includes the following information:

“Delete event for CI “CI Name” triggered by discovery”.

Workaround

An SM change request does not contain a description of the reason for deletion, however it is possible to extract specific information about CI deletions from the UCMDB History Database. UCMDB data provides information about the user or the discovery pattern that initiated the CI deletion.

How Do I Monitor Relationship Changes Between UCMDB and SM?

To understand the relationship change in SM, a distinction must be made between the various types of Relationship Changes:

- The second endpoint of the relationship has Changed, so instead of CI X being linked to CI Y through a relationship, now CI X is related to CI Z.
- An attribute of the relationship has changed.

The first type of Relationship change is supported by the UCMDB-SM integration, therefore, such "Relationship Changes" can either invoke CI relationship updates, or perform the creation of Incidents or Changes, which are then reviewed and monitored.

The second is also supported, but it is not covered out-of-the-box; you can configure the Universal CMDB query to expose such attributes of the relationship, and configure the Service Manager WSDL to expose the mapped field, and then configure the adapter mapping configuration in the XML and Groovy. However such a Relationship Attribute Change cannot perform the creation of Incidents or Changes, and only supports invoking CI relationship updates directly.

What Kinds of Relationships are Pushed from UCMDB to SM?

Any kinds of relationships are pushed from UCMDB to SM under the following conditions:

- The relationship appears in a Push Query located in the **Service Manager** > **Push** folder in the UCMDB Query Manager.
- The relationship is named **Root** in the Push Query.
- The relationship is mapped to an appropriate target in SM in the UCMDB configuration files (XML and Groovy files).

The out-of-the-box relationships that are pushed from UCMDB to SM are relationships between two CIs such as:

- Between Business Services and Applications;
- Between Business Service and Host;
- Between an Application and a Network Component; or
- Between Host, Network Components and Printers.

What is a Root CI Node?

A Root node is a TQL query node that represents the CI type that is created through push to SM from the TQL query structure. The rest of the TQL query structure contains information that can be incorporated within the Root CI type and is used to enrich the record in SM with additional information and or attributes.

What Is a Root Relationship?

A Root relationship is a relationship within a query and created in SM through push. It represents a relationship between two Root CIs. Only the relationships marked with Root are pushed to SM.

What is the "Virtual-Compound" Relationship Type Used in a UCMDB-SM Integration Query?

When more than two UCMDB CI entities are connected in series, the "Virtual-Compound" represents the relationship between the first and last entities. This is a virtual relationship, as no physical representation exists.

The "Virtual-Compound" relationship type is a relationship that links two CI type entities that have a logical relationship. See the following figure.

Explanation

The illustration shows an example of a Virtual-Compound relationship. The relationship in SM is created directly between the Host and the Business Service.

When Do I Need the Population Feature?

You need the population feature under any of the following circumstances:

- You have done modeling in SM, especially when you are in the planning and design phases, and you want your models to be reflected in UCMDB;
- You want to implement the UCMDB-SM integration, however you have already invested in your SM CMDB and do not want to lose that investment;
- You want to continue to maintain some parts of the SM CMDB while maturing your UCMDB/Discovery implementation.

Can I Populate Physically Deleted CIs from SM to UCMDB?

No.

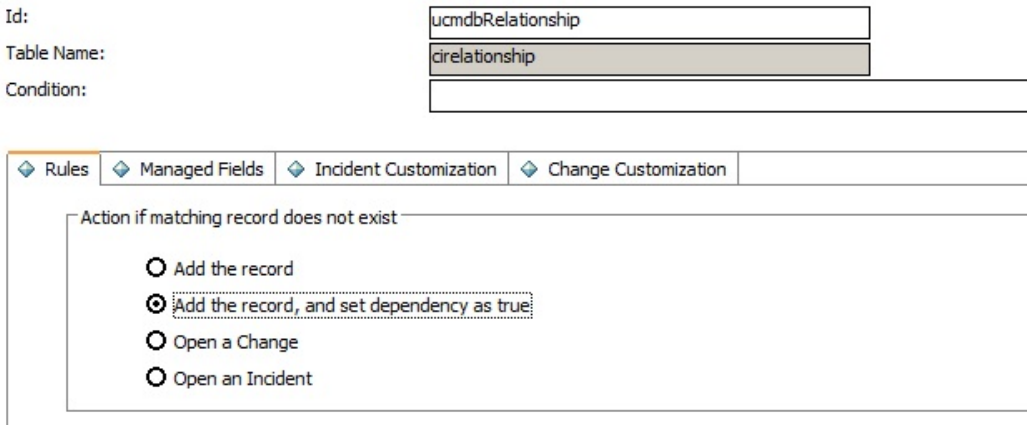
Physical deletions of CIs are allowed in SM, but SM cannot get such “deletion changes” and the population feature will not synchronize such changes to UCMDB.

Physical deletions of CIs can be considered as exceptions, which occur only after you create CIs by mistake. Normally, you delete a CI by setting its status to something like **Retired/Consumed**. In case such CIs have been populated to UCMDB, it is your responsibility to remove them manually from UCMDB.

How Do I Keep the Outage Dependency Setting of a CI Relationship in SM?

Out-of-the-box, CI relationships that are pushed from UCMDB to SM do not have outage dependency information by default. If you need such information, you can set the DEM rule of the CI Relationship WSDL as follows:

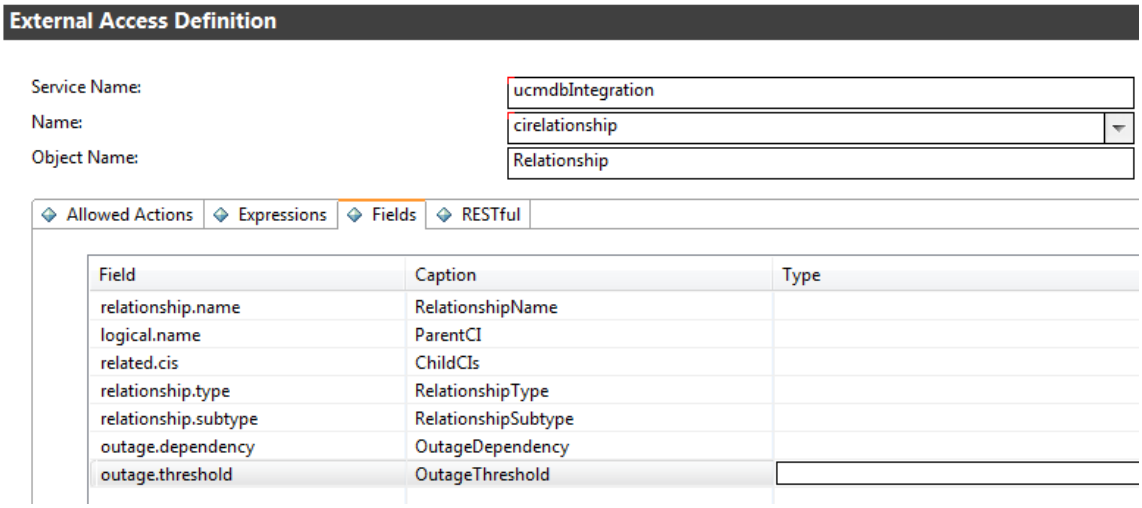
1. Log in to Service Manager as a system administrator.
2. Navigate to **Tailoring** > **Web Services** > **Discovered Event Manager Rules**.
3. Open the ucmdbRelationship record.
4. On the Rules tab, select **Add the record, and set dependency as true**.



This will set the Outage Dependency of each CI Relationship to true, and set the number of dependent downstream CIs to 1 (because UCMDB supports only one-to-one relationships).

If you want to set outage dependency only for some relationships, for example, if you want to configure outage dependency for relationships that start from Business Service, you can configure the adapter configuration file (XML) and WSDL definition; you can also configure outage dependency per relationship type (UCMDB query).

1. In the WSDL definition, expose fields **outage.dependency** and **outage.threshold**.



2. In the XML file, set the exposed outage fields. For example, if you want to set the outage dependency to `true` and threshold to `1` for Business Service relationships, you simply need to change the XML mapping file **SM Business Service Relations Push 2.0.xml**. In the XML mapping file, use the following `OutageDependency` and `OutageThreadhold` settings:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
    <integration>
        <info>
            <source name="UCMDB" vendor="HP" version="10.20"/>
            <target name="SM" vendor="HP" version="9.40"/>
        </info>
        <import>
            <scriptFile path="mappings.scripts.SMPushFunctions"/>
        </import>
        <target_entities>
            <source_instance root-element-name="Root_directly" query-name="EA_SM
    Business Service Relations Push">
                <target_entity name="Relationship">
                    <target_mapping datatype="STRING" name="RelationshipType"
    value="SMPushFunctions.getDisplayName(Root_directly['element_
    type'],ClassModel)"/>
                    <target_mapping datatype="STRING" name="ParentCI"
    value="SMPushFunctions.getEndId(OutputCI.getExternalId().getEnd1Id())"/>
                    <target_mapping datatype="STRING_LIST" name="ChildCIs"
```

```
value="[SMPushFunctions.getEndId(OutputCI.getExternalId().getEnd2Id())]"/>

                <target_mapping datatype="BOOLEAN" name="OutageDependency"
value="true"/>
                <target_mapping datatype="NUMBER" name="OutageThreadhold"
value="1"/>

            </target_entity>
        </source_instance>
    </target_entities>
</integration>
```

How Do I Create an XML Configuration File?

You create an XML configuration file in Adapter Management. You can copy the content of an existing XML configuration file to the new file and then make necessary edits.

To create an XML configuration file:

1. Log in to UCMDB as an administrator.
2. Navigate to **Data Flow Management** > **Adapter Management**. **ServiceManagerEnhancedAdapter9-x** > **Configuration Files**.
3. Click the **Create new resource** icon  .
4. Select **New Configuration File**.
5. Enter a name for the file. The file name should use this format: <*AdapterID*>/mappings/<*Synch Type*>/<*filename*>. For example: `ServiceManagerEnhancedAdapter9-x/mappings/push/SM Computer Push.xml`.
6. Click **OK**. A message is displayed, asking if you want to open the file in the Visual Mapping tool editor.
7. Click **Yes** or **No** to continue.
   UCMDB creates the new XML configuration file in the Configuration Files folder of the adapter.
8. Copy the content of an existing XML configuration file to the new file.
9. Make necessary edits to the new file.

Caution: Invalid XML

When removing XML elements from an XML file, keep in mind that the remaining elements must constitute a valid XML file, which will be used to translate the UCMDB Query Definition.

How Do I Use the Load Fields Button to Add Multiple Managed Fields?

Service Manager stores a list of managed fields in the **ucmdbIntegration** web service, which consists of a number of web services objects. You can add more managed fields to DEM Rules so that Service Manager can monitor changes in more CI attributes in UCMDB and trigger the actions defined in relevant DEM Rules.

You can manually add managed fields that are exposed in associated WSDL definitions to DEM Rules; however, you can use the **Load Fields** button to automatically (and therefore correctly) add managed fields to DEM Rules.

1. Click the **Managed Fields** tab of the DEM Rule.
2. Click the **Load Fields** button.
3. If the table (in the Table Name field) of the DEM rule record has only one WSDL definition associated to it, all fields exposed in the WSDL definition are immediately added to the Managed Fields list.
   A message is displayed: `<XX> new fields loaded.`
4. If the table has more than one WSDL definition associated to it, the Managed Fields Importing wizard opens, and a list of WSDL definitions (ucmdbIntegration web service objects) is displayed.
   a. Select one or more objects, and click **Next**. All new fields that can be added from the selected web service objects are displayed.
   b. If you want to add all of the fields, click **Finish**; if you want to ignore some of them, change their Action value from **Add** to **Ignore**, and then click **Finish**.
      A message is displayed: `<XX> new fields loaded.`
5. Save the DEM Rule record.

What Is the Purpose of the <container> Element in the Population Configuration File (smPopConf.xml)?

Out-of-the-box, the smPopConf.xml file has a container element.

```
<tql name="SM RunningSoftware Population 2.0" citype="running_software">
      <request  type="Retrieve" dataType="ci"
        resourceCollectionName="ucmdbRunningSoftwares"
        resourceName="ucmdbRunningSoftware"
        basicQueryCondition="type=&quot;runningsoftware&quot;"
        fullQueryCondition="istatus~=&quot;Retired/Consumed&quot;"
        changedCreationQueryCondition="created.by.date&gt;'{fromDate}' and
    istatus~=&quot;Retired/Consumed&quot;"
        changedUpdateQueryCondition="created.by.date&lt;='{fromDate}' and
    devicemodtime&gt;'{fromDate}' and istatus~=&quot;Retired/Consumed&quot;"
        changedDeletionQueryCondition="devicemodtime&gt;'{fromDate}' and
    istatus=&quot;Retired/Consumed&quot;"/>
        <container tql="SM Computer Population 2.0"
          keyFields="CIIdentifier"
          linkTql="SM Computer Composition Software 2.0"
          linkRetrieveCondition="downstreamci.logical.name=&quot;$$$&quot; and
    upstreamci.type=&quot;computer&quot; and
    downstreamci.type=&quot;runningsoftware&quot; and
    relationship.subtype=&quot;Composition&quot;"
          linkRetrieveConditionKey="CIIdentifier"
          linkValueFields="upstreamci.logical.name"/>
    </tql>
```

- In UCMDB, RunningSoftware CIs must exist together with a Root Container (Node); however, Service Manager allows RunningSoftware CIs without a Node.
- The integration adapter synchronizes CIs and Relationships separately; when populating a RunningSoftware CI, the integration has no chance to check if a relationship exists between the CI and a Node.

With the <container> element, the integration populates RunningSoftware CIs together with a container.

Can I Populate Sub-Item Deletions?

Yes.

Service Manager and UCMDB store CI information in different data structures, and therefore one SM CI may be synchronized to UCMDB as several CIs. For example, during population, an SM computer CI record is synchronized to a Node CI in UCMDB, and the computer CI's attributes to CIs such as IP, Interface, Location, etc (which are referred to as sub-items of the Node CI.). In this case, the Node CI is the root CI.

The integration allows you to populate sub-item deletions to UCMDB. For example, if you delete the IP Address attribute value of a computer, the corresponding IP CI record in UCMDB will be deleted too.

What Happens if a Population Job Failed or Completed?

When a population job failed

The failure prevents the remaining population tasks from running. The next job run will start from the last Success time. If pagination occurs (that is, the tasks are divided into multiple pages), the tasks will run again and again within the first page from the last “Success” time (once the end of the first page is reached, no new tasks will be executed).

When a population job completed

When the status of a job is "completed" (but not "completed successfully"), warnings occurred. A warning does not prevent the remaining population tasks from running. The next job run will run all tasks again starting from the last Success time. If pagination occurs (the tasks are divided into multiple pages), the tasks on all pages will be re-run (including those that successfully completed last time).

Chapter 5: Tailoring the Integration

You can tailor the UCMDB-SM integration to meet your business needs by adding or removing managed CI types, attributes, and relationship types. This chapter describes the integration architecture and tailoring options for data push, population, and federation.

This section includes:

Integration Architecture

Before you tailor the integration, you should understand how the following components of the out-of-the-box integration work.

Integration Class Model

UCMDB 9.x or later no longer uses a private class model of CI types to manage integration CIs, as was required in prior versions. Instead, the integration uses the standard UCMDB managed objects and maps them to Service Manager CI types and attributes with queries and transformation files.

Integration Queries

This section describes out-of-the-box queries used for data push, Actual State, and population.

Queries for Push

For the push feature, the integration uses a collection of queries to gather CI attribute information from Universal CMDB and send it to the Service Manager system.

To access the out-of-the-box data push queries, navigate to **Modeling** > **Modeling Studio**, select **Queries** for Resource Type, and then navigate to the **Root** > **Integration** > **Service Manager** > **Push** folder.

If you want to change what CI types, Relationship types or attributes are part of the integration, you must also edit the integration queries to support your updated CI types, CI Relationship types and attributes.

Query name	Description
SM Local Printer Push 2.0	This query gathers CI attributes from printer CIs. It also gathers related CI attributes from the Node CI type.
SM Net Printer Push 2.0	This query gathers CI attributes from Node CI types whose NodeRole contains “printer.” It also gathers related CI attributes from the following CI types through containers and links: IPAddress, Interface, CPU, FileSystem, DiskDevice, and Location.
SM Mainframe Push 2.0	This query gathers CI attributes from the following Node CI types: Mainframe Logical Partition, and Mainframe CPC. It also gathers related CI attributes from the following CI types through containers and links: IPAddress, Interface, CPU, FileSystem, DiskDevice, and Location.
SM Mobile Device Push 2.0	This query gathers CI attributes from Node CI types whose NodeRole contains “pda_handheld.” It also gathers related CI attributes from the following CI types through containers and links: IPAddress, Interface, CPU, FileSystem, DiskDevice, and Location.

Query name	Description
SM Network Component Push 2.0	This query gathers CI attributes from Node CI types whose NodeRole contains router, adsl_modem, appletalk_gateway, bandwith_manager, cable_model, csu_dsu, ethernet, fddi, firewall, hub, kvm_switch, load_balancer, multicast_enabled_router, nat_router, token_ring, undefined_network_ component, voice_gateway, voice_switch, or vpn_ gateway. It also gathers related CI attributes from the following CI types through containers and links: IPAddress, Interface, CPU, FileSystem, DiskDevice, and Location.
SM Cluster Push 2.0	This query gathers CI attributes from the following Node CI type: ClusterResourceGroup. It also gathers related CI attributes from the following CI types through containers and links: IPAddress, Interface, CPU, FileSystem, DiskDevice, Location, and Cluster.
SM Computer Push 2.0	This query gathers CI attributes from the node CI type with NodeRole containing “desktop”, “server”, “virtualized_system” or not set. It also gathers related CI attributes from the following CI types through containers and links: IPAddress, Interface, CPU, FileSystem, DiskDevice, and Location.
SM Storage Push 2.0	This query gathers CI attributes from the Node CI type whose NodeRole contains san_ switch, san_gateway, san_router, or whose Display Name is Storage Array. It also gathers related CI attributes from the following CI types through containers and links: IPAddress, Interface, CPU, FileSystem, DiskDevice, and Location.
SM Running Software Push 2.0	This query gathers CI attributes from Running Software CIs.
SM Switch Push 2.0	This query gathers CI attributes from the Node CI type whose NodeRole contains atm_ switch, frame_relay_switch, or lan_switch. It also gathers related CI attributes from the following CI types through containers and links: IPAddress, Interface, CPU, FileSystem, DiskDevice, and Location.
SM Service Element Push 2.0	This query gathers CI attributes from the Business Element CI type.
SM Business Service Push 2.0	This query gathers CI attributes from the Business Service CI type.

Query name	Description
SM Layer2 Topology Relations Push 2.0	This query gathers relationships between the following components: Two or more nodes. The query includes compound relationships because the relationships can extend through a group.
SM Business Service Relations Push 2.0	This query gathers relationships between the following components: • Business Service and Running Software CIs • Business Service and Node CIs • Two or more Business Services The query includes compound relationships because the relationships can extend through a group.
SM CRG Relations Push 2.0	This query gathers relationships between the following components: Node and Cluster Resource Group CIs. The query includes compound relationships because the relationships can extend through a group.
SM Node Relations Push 2.0	This query gathers relationships between the following components: • Node and Printer CIs • Node and RunningSoftware CIs The root class of the relationship is composition.

Note: The Service Manager Enhanced Generic Adapter has been introduced since UCMDB version 10.20. This adapter can coexist with an old XSLT adapter. In order to distinguish its queries from those of an XSLT adapter, all queries of the enhanced adapter are suffixed with **2.0**.

Queries for Actual State

Out-of-the-box, the queries in the following table are used for retrieving CI information from UCMDB to the Actual State section of the Service Manager Configuration Item (CI) form. Service Manager retrieves CI Actual State information by calling a UCMDB web service that retrieves CI data according to these queries.

The queries are located in the **Integration** > **SM Query** folder in the UCMDB Modeling Studio.

Query name	Description
localPrinterExtendedData	This query gathers real-time extended information from Printer CIs in UCMDB.
applicationExtendedData	This query gathers real-time extended information from RunningSoftware CIs in UCMDB.
businessServiceExtendedData	This query gathers real-time extended information from business service CIs in UCMDB.
hostExtendedData	This query gathers real-time extended information (such as Asset, Party, Location, LogicalVolume, WindowsService, Printer, InstalledSoftware, FileSystem, IPAddress, Interface, DiskDevice, and Cpu) from the node CI type in UCMDB.

Queries for Federation

Due to a technical limitation of the Generic Adapter framework, queries that describe the relationships between UCMDB CI types and the federated CI types (Incident, Problem and RFC for Service Manager) are required. There are approximately 300 queries that are defined for federation in the **Integration** > **Service Manager** > **Federation** folder in the Modeling Studio, so that you do not have to develop them by yourself for the out-of-the-box CI Types in UCMDB. Usually, you do not need to modify these queries; instead, you only need to define new queries for your own custom CI types in UCMDB.

For information on how to define queries for federation, see the *Achieving Data Federation Using the Generic Adapter* section in the *HP Universal CMDB Developer Reference Guide*.

Note: This technical limitation was introduced in UCMDB 10.20, and might be fixed in a future release.

Queries for Population

For CI/CI Relationship population, the integration uses the following queries to save CI/CI Relationship attribute information to UCMDB.

Query Name	Description
SM Business Service Population 2.0	Defines the CI store structure of business service CIs.

Query Name	Description
SM Business Application Population 2.0	Defines the CI store structure of Application Service CIs.
SM Infrastructure Service Population 2.0	Defines the CI store structure of Infrastructure Service CIs.
SM Running Software Population 2.0	Defines the CI store structure of running software CIs.
SM Computer Population 2.0	Defines the CI store structure of computer CIs.
SM CLIP Down Time Population 2.0	Defines the CI store structure of ScheduledDowntime CIs.
SM Biz Containment Biz 2.0	This query defines the CI store structure of CI relationships in which a bizservice CI contains another.
SM Biz Usage Biz 2.0	Defines the CI store structure of CI relationships in which a bizservice CI uses another.
SM Biz Containment Computer 2.0	This query defines the CI store structure of CI relationships in which a bizservice CI contains a computer CI.
SM Biz Containment Software 2.0	Defines the CI store structure of CI relationships in which a bizservice CI contains a RunningSoftware CI.
SM Biz Usage Computer 2.0	Defines the CI store structure of CI relationships in which a bizservice CI uses a computer CI.
SM Biz Usage Software 2.0	Defines the CI store structure of CI relationships in which a bizservice CI uses a RunningSoftware CI.
SM Computer Connects Computer 2.0	Defines the CI store structure of CI relationships in which a computer CI connects to another.
SM Computer Composition Software 2.0	Defines the CI store structure of CI relationships in which a RunningSoftware CI is contained within a computer CI and the RunningSoftware CI cannot exist without the container.
SM CI Connection Down Time CI 2.0	Defines the CI store structure of CI relationships in which a ScheduledDowntime CI connects to an affected CI.

Query Requirements

The integration requires that any custom queries you create meet certain formatting conditions. Any queries that you want to include in the integration must meet these conditions:

- To query CIs, a query must contain one CI type labeled Root. The Root node is the main CI that UCMDB synchronizes. All other CIs are contained CIs of the Root CI.
- To query relationships, a query must contain one or more relationships labeled Root.
- A query must contain only the Root CI and CIs that are directly connected to it. The Root CI is always the top node of the query hierarchy.
- A query layout cannot have cycles.
- If a query synchronizing relationships has cardinality, it must be cardinality 1...*. Additional cardinality entries must have an OR condition between them.
- If you want the integration to only synchronize specific CIs, you must configure the condition on the query to filter such CIs.

Service Manager Web Services

Service Manager uses web services messages to get and receive CI information from your UCMDB system. Out-of-the-box, UCMDB sends more CI attribute information than the Service Manager system actually manages. Service Manager users can view all of the CI attribute information the UCMDB system sends from the Actual State section of the CI record.

Service Manager publishes several web services for use by the UCMDB-SM integration. The UCMDB system uses the web services to map UCMDB CI types and CI attributes to web services objects that the Service Manager system recognizes. You can add UCMDB CI types or CI attributes that you want Service Manager to manage by using the Visual Mapping tool, and the tool will automatically update one or more of these web services to define web service objects.

Managed Fields

Note: Managed fields are used only for the data push feature.

A Service Manager managed field is a field where the system compares the CI attribute value in the incoming UCMDB web services message to the value in a Service Manager CI record. If the values in the

web services message do not match those in the CI record, Service Manager runs a Discovery Event Manager (DEM) rule to determine what action to take. The DEM rule determines which of the fields that are published as web services objects are fields managed by the integration. Only value changes in managed fields trigger the DEM rule.

The **ucmdbIntegration** web service consists of a set of web services objects, each of which defines a list of web service fields. Out-of-the-box, the integration uses only part of them (see the Mappings between Service Manager web service objects, tables, and DEM rules table), some of them (along with their relevant DEM Rules) have been deprecated (see the Deprecated ucmdbIntegration web service objects for data push table), and some are used for population or federation (see the ucmdbIntegration web service objects used for population or federation table).

Mappings between Service Manager web service objects, tables, and DEM rules

This web service object	Publishes fields from this table	And uses this DEM rule ID
Relationship	cirelationship	ucmdbRelationship
ucmdbRunningSoftware	device	ucmdbRunningSoftware
ucmdbBusinessService	joinbizservice	ucmdbBusinessService
ucmdbNode	joinnode	ucmdbNode

Deprecated ucmdbIntegration web service objects for data push

This web service object	Publishes fields from this table	Recommended replacement (object)
ucmdbApplication	device	ucmdbRunningSoftware
ucmdbComputer	ucmdbComputer	ucmdbNode
UcmdbDevice	device	ucmdbRunningSoftware
ucmdbNetwork	joinnetworkcomponents	ucmdbNode
ucmdbPrinter	joinofficeelectronics	ucmdbNode

ucmdbIntegration web service objects used for population or federation

This web service object	Publishes fields from this table	And is used for	Requires a DEM Rule?
cirelationship1to1	cirelationship1to1	Population	No
ucmdbIDPushBack	device	Population	No
UcmdbChange	cm3r	Federation	No
UcmdbChangeTask	cm3t	Federation	No

ucmdbIntegration web service objects used for population or federation, continued

This web service object	Publishes fields from this table	And is used for	Requires a DEM Rule?
UcmdbIncident	probsummary	Federation	No
UcmdbProblem	rootcause	Federation	No

The following sections list the fields published as web services objects used for data push (see the Mappings between Service Manager web service objects, tables, and DEM rules table) and indicate whether or not they are managed fields in an out-of-the-box Service Manager system. You can use this reference to determine if you need to publish a field as a web service object, and also if you need to create a DEM rule for the object.

Object Name: Relationship

Service Manager publishes the following fields from the cirelationship table:

Web service and managed fields of the Relationship object

Field published as web service object	Caption used in web service messages	Is the field a managed field?
relationship.name	RelationshipName	
logical.name	ParentCI	
related.cis	ChildCIs	Yes
relationship.subtype	RelationshipSubtype	

Object Name: ucmdbRunningSoftware

Service Manager publishes the following fields from the device table:

Web service and managed fields of the ucmdbRunningSoftware object

Field published as web service object	Caption used in web service messages	Is the field a managed field?
ucmdb.id	UCMDBId	
ci.name	ApplicationName	Yes
type	Type	
subtype	Subtype	
company	CompanyId	

Web service and managed fields of the ucmdbRunningSoftware object, continued

Field published as web service object	Caption used in web service messages	Is the field a managed field?
logical.name	CIIdentifier	Yes
product.version	ProductVersion	
vendor	Vendor	
version	Version	
id[1]	CIName	

Object Name: ucmdbBusinessService

Service Manager publishes the following fields from the joinbizservice table:

Web service and managed fields of the ucmdbBusinessService object

Field published as web service object	Caption used in web service messages	Is the field a managed field?
ucmdb.id	UCMDBId	
ci.name	ServiceName	Yes
type	Type	
subtype	Subtype	
company	CustomerId	
logical.name	CIIdentifier	Yes
vendor	ServiceProvider	
id[2]	CIName	

Object Name: ucmdbNode

Service Manager publishes the following fields from the joinnode table.

[1]This attribute is used only for the population feature.
[2]This attribute is used only for the population feature.

Web service and managed fields of the ucmdbNode object

Field published as web service object	Caption used in web service messages	Is the field a managed field?
ucmdb.id	UCMDBId	
type	Type	
subtype	Subtype	
company	CustomerId	
logical.name	CIIdentifier	Yes
default.gateway	DefaultGateway	Yes
network.name	DNSName	Yes
building	Building	Yes
room	Room	Yes
floor	Floor	Yes
location	Location	
addlIPAddr[addlIPAddress]	AddlIPAddress	Yes
addlIPAddr[addlSubnet]	AddlSubnet	Yes
addlMacAddress	AddlMacAddress	Yes
bios.id	BIOSId	Yes
operating.system	OS	Yes
os.version	OSVersion	Yes
physical.mem.total	PhysicalMemory	Yes
serial.no.	SerialNo	
vendor	Vendor	
cpu[cpu.id]	CpuID	
cpu[cpu.name]	CpuName	
cpu[cpu.clock.speed]	CpuClockSpeed	
file.system[mount.point]	MountPoint	

Web service and managed fields of the ucmdbNode object, continued

Field published as web service object	Caption used in web service messages	Is the field a managed field?
file.system[disk.type]	DiskType	
file.system[file.system.type]	FilesystemType	
file.system[disk.size]	DiskSize	
asset.tag	AssetTag	
machine.name	HostName	Yes
disk.device[model.name]	ModelName	
disk.device[disk.vendor]	DiskVendor	
disk.device[disk.name]	DiskName	
id[1]	CIName	
isVisualization	IsVisualization	
istatus	AssetStatus	

Service Manager Reconciliation Rules

Service Manager reconciliation rules allow the integration to identify CI records in your Service Manager system that match CIs in your UCMDB system. Service Manager attempts to reconcile CI records with every push of CI attributes from your UCMDB system. The integration uses the following workflow to match UCMDB CIs with Service Manager CIs.

1. The UCMDB system sends a web service message containing the latest CI attribute data to Service Manager.
2. Service Manager scans the web service message for the CI ucmdb.id value.

   Note: Out-of-the-box, Service Manager does not display the ucmdb.id field on CI record forms to prevent users from changing the value. If you want to add this value to your forms, you can find the ucmdb.id field defined in the **device** table. HP recommends that you make this a read-only field.

[1]This attribute is used only for the population feature.

3. Service Manager searches for an existing CI record that has the same ucmdb.id value.
4. If Service Manager finds a CI with a matching ucmdb.id value, no reconciliation is needed. Service Manager compares the UCMDB CI attributes to the Service Manager managed fields and runs the appropriate Discovery Event Manager (DEM) rules as needed.
5. If Service Manager cannot find a CI with a matching ucmdb.id value, it runs the reconciliation rules.
6. Service Manager searches for an existing CI record with the same reconciliation field values.
7. If Service Manager finds a CI with a matching reconciliation field value, it updates the CI record with the ucmdb.id value of the matching UCMDB CI. Service Manager compares the UCMDB CI attributes to the Service Manager managed fields and runs the appropriate DEM rule as needed.
8. If Service Manager cannot find a CI with a matching reconciliation field value, it runs the DEM rule for “Action if matching record does not exist.” Out-of-the-box, the DEM rule has Service Manager create a new CI record. Service Manager creates the CI record using the ucmdb.id value of the incoming UCMDB CI.

Performance Implications

Because Service Manager attempts to reconcile CIs with every push, the number of reconciliation fields you have will affect the integration’s performance. The more reconciliation rules you have, the more searches Service Manager must perform to match CIs. To improve the performance of reconciliation searches, you should choose reconciliation fields that are unique keys of the underlying Service Manager table. For example, if you want to reconcile CI records in the **device** table, use the logical.name field as a reconciliation field because it is a unique key. For information about how to create a reconciliation rule, see "How to Add DEM Reconciliation Rules" on page 134.

Dependence on DEM Rules

Service Manager uses the **Action if matching record does not exist** DEM rule whenever it cannot reconcile CIs. You must review the DEM settings and decide if they meet your business standards prior to the initial push of CIs from UCMDB to Service Manager. For example, you can have Service Manager create a change request for every CI in the initial CI push by selecting the **Open a Change** option.

Service Manager Discovery Event Manager Rules

You only need to create Discovery Event Manager (DEM) rules if you want to accomplish any of the following custom actions:

Change the Conditions Under Which a DEM Rule Runs

Service Manager will run a DEM rule only if the condition field evaluates to true. Out-of-the-box, no DEM rule has a condition statement that restricts when the rule runs, and all the integration DEM rules will always run by default.

You can update a DEM rule's condition statements if you want to restrict when Service Manager runs your DEM rules. For example, adding the following condition to the ucmdbNode DEM rule restricts the rule to desktop CIs.

```
subtype in $L.file=”Desktop”
```

You can also use the condition field to create multiple DEM rules that apply to the same table name. For example, the following DEM rules both apply to the joinnode table.

DEM rules using different conditions to affect the same table

DEM rule Id	Table Name	Condition
ucmdbNode	joinnode	`subtype in $L.file!=”Desktop”`
ucmdbDesktop	joinnode	`subtype in $L.file=”Desktop”`

Typically, you will only need to add conditions if your business processes require the integration to take different actions with certain CI types or SLAs.

Change the Action the DEM Rule Takes

Out-of-the-box, the integration DEM rules take the following actions:

- Add a CI record when the UCMDB data does not match an existing Service Manager CI record
- Open a Change or log results and update a CI record when the UCMDB CI attribute data does not match the CI attribute data in the Service Manager CI record
- Delete a CI record when the UCMDB data specifies that the CI has been deleted

You can change the integration DEM rules to meet your business processes. For example, you could use the ucmdbNode DEM rule to open a change when the integration finds a non-desktop CI with

unexpected data, and use the ucmdbDesktop DEM rule to log results and update the record when the integration finds a desktop CI with unexpected data.

Caution: If you want to use the Change Management verification and Change Management validation features of the integration, your DEM rules must use the **Open a Change** option for the "Action if record exists but unexpected data discovered" event.

Create Custom JavaScript to Open Change or Incident Records

Service Manager uses the **discoveryEvent** JavaScript to create CI names and to set the values of required fields when opening change or incident records. Out-of-the-box, the script uses the following default values.

Default values to create a new CI

You can update the createCIName and populateNewCI functions to set the following CI values.

Default values used to create a new CI

CI attribute	Default value defined in discoveryEvent
record.logical_name	System generated ID number
record.assignment	AUTO
record.istatus	Installed
record.os_name	Value in record.operating_system

Default values to create a new change

You can update the populateChange function to set the following change values.

Default values used to create a new change

CI attribute	Default value defined in discoveryEvent
change.category	Unplanned Change
change.reason	Value in reason
change.initial_impact	3
change.severity	3
change.coordinator	Change.Coordinator

Default values used to create a new change, continued

CI attribute	Default value defined in discoveryEvent
change.requested_by	discovery
change.status	initial

Default values to create a new incident

You can update the populateIncident function to set the following incident values.

Default values used to create a new incident

CI attribute	Default value defined in discoveryEvent
incident.category	incident
incident.subcategory	hardware
incident.product_type	missing or stolen
incident.assignment	Hardware
incident.initial_impact	3
incident.severity	3
incident.logical.name	Value of id
incident.site_cateogry	C
incident.contact_name	ANALYST, INCIDENT
incident.affected_item	MyDevices

Integration Tailoring Options

The integration offers the following tailoring options:

- "How to Update the Integration Adapter Configuration File (sm.properties)" on the next page
- "How to Add DEM Reconciliation Rules" on page 134
- "How to Add Discovery Event Manager Rules" on page 136
- "How to Add a CI Attribute to the Integration for Data Push" on page 141

How to Update the Integration Adapter Configuration File (sm.properties)

The integration uses a properties file (sm.properties) as a configuration file of the adapter. Out-of-the-box, this file has been set up based on best practices, so usually you can keep the default parameter values. Optionally, you can update the parameter values to better suit your needs.

To update the `sm.properties` file:

1. Log in to UCMDB as an administrator.
2. Navigate to **Data Flow Management** > **Adapter Management** > **ServiceManagerEnhancedAdapter9-x** > **Configuration Files**.
3. Click the properties configuration file: `sm.properties`.
4. Update the parameter values as needed. For a list of the parameters, see the following table.

   #if true, will use globalId instead of ucmdbId in the SM integration.

   use.global.id=true

   # Type 0, the feature will be disabled

   # Type 1, the enum type will expand to "{value}"

   # Type 2, the enum type will expand to "{index}-{value}"

   type.of.expand.enum=2

   #if false, will use the type directly instead of label of the type.

use.type.label=true

#parameter for populate data from service manager

pop.pagination.switch=on

pop.pagination.recordcount=1000

pop.createci.key=sm_id

the property used as ucmdb id key for population in xslt files, it is treated as "global_id" by adapter if is.global.id is ture,otherwise it is treated as CMDB ID.

pop.ucmdb.id.key=ucmdb_id

#SM web service for push back uCMDB ID

ucmdbid.pushback.request=UpdateucmdbIDPushBackRequest

ucmdbid.pushback.xslt=ucmdbid_pushback.xslt

#whehter checking the soap connections to SM instances before running the job

check.sm.connections=false

#the external web service interface type. restful or soap

connector.type=restful

Parameters in the sm.properties file

Parameter	Default value	Comment
timeout.minutes	10	The integration connection timeout value (in minutes)

Parameters in the sm.properties file, continued

Parameter	Default value	Comment
number.of.concurrent.sending.threads	6	The number of concurrent threads used for the data push feature ■ 1: Disabled ■ 2 or higher: Enabled **Note:** If you are connecting to multiple Service Manager instances to improve the CI data push performance (see the URL Override configuration in "How to Create an Integration Point in UCMDB" on page 33), you are recommended to increase this value for optimized performance. For example, set it to 12 if you are connecting to two Service Manager instances.
min.objects.for.concurrent.sending	50	The minimum number of Service Manager objects that is required to use concurrent sending instead of single thread sending **Note:** It is used for the push feature.
number.of.chunks.per.thread	3	The number of chunks per thread used for the push feature Total of number of chunks = number.of.chunks.per.thread * number.of.concurrent.sending.threads

Parameters in the sm.properties file, continued

Parameter	Default value	Comment
recommended.min.cis.per.chunk	50	The minimum allowed number of CIs per CI data chunk. This value is used to mitigate the risk of making your data chunks too small. For example, suppose your system uses the following settings: ■ number.of.chunks.per.thread=3 ■ number.of.concurrent.sending.threads=3 ■ recommended.min.cis.per.chunk=50 When pushing 1000 CIs, the number of CIs in each chunk is calculated as 1000/3/3. The calculated value is greater than the recommended.min.cis.per.chunk value, so the calculated value is used. However, when pushing 270 CIs, the calculated value is less than the recommended.min.cis.per.chunk value, so the calculated value is used.
max.running.hours.for.multi.threaded	20	The maximum number of hours before a push request times out in a multi-threaded environment.
number.of.cis.per.request	1000	The maximum number of objects retrieved from Service Manager by ID **Caution:** It is used for the population and federation features. Do not set it to a number greater than 1000 in case the request has a 64K limit.

Parameters in the sm.properties file, continued

Parameter	Default value	Comment
federation.request.pagination.switch	on	
federation.request.max.size	2000	size of query conditions for federation
federation.isin.max.count	500	size of isin{} size
type.of.expand.enum	2	It configures the value mapping rule for the UCMDB enum type ■ 0: The feature will be disabled ■ 1: The enum type will expand to "{value}" ■ 2: The enum type will expand to "{index}-{value}" **Note:** It is used for the push feature.
op.pagination.switch	on	It indicates if pagination (client driven) is enabled ■ on: Enabled. ■ off: Disabled. **Note:** It is used for the population feature.
pop.pagination.recordcount	1000	The maximum number of records displayed on each page when pagination is enabled. **Note:** It is used for the population feature.

Parameters in the sm.properties file, continued

Parameter	Default value	Comment
pop.createci.key	sm_id	The UCMDB field of a CI record that stores the Service Manager CI ID. **Note:** It is used for UCMDB ID Push Back by the population feature.
ucmdbid.pushback.request	UpdateucmdbIDPushBackRequest	The web service request for pushing the UCMDB ID back to Service Manager. **Note:** It is used for the population feature.
check.sm.connections	false	It indicates whehter to check the SOAP connections to Service Manager instances before running a job. You can enable it under any of the following circumstances: ■ Your Service Manager is running in High Availability mode (with load balancing), and you want to connect UCMDB to multiple Service Manager instances. ■ You want UCMDB to not run a job when no integration connections are available, rather than run the job and then report a failure.

How to Add DEM Reconciliation Rules

It is possible that your Service Manager system already contains CI records that match CIs in your UCMDB system. Rather than add duplicate CI records to your Service Manager system, you can configure Service Manager to reconcile CI records between the two systems based on the values of specific fields.

Service Manager always attempts to reconcile CI records based on the unique key field of the Service Manager table and the **ucmdb.id** field. You can specify additional fields to reconcile on from the DEM

Reconciliation Rules form. If Service Manager finds a matching value in any one of these fields, it updates the Service Manager CI record with the attributes from the incoming UCMDB record.

When multi-tenancy is enabled, Service Manager only reconciles the CIs whose company ID matches the company ID in the data push job. For example, when pushing CIs from company 2, the reconciliation rules only apply to the Service Manager CI records that have the company code corresponding to company number 2.

In order to specify reconciliation fields, you will need to be familiar with the table and field names in both your Service Manager and UCMDB systems. If you want to reconcile on a particular attribute from the UCMDB system, you should verify that there is a corresponding Service Manager managed field for the attribute. Without such a mapping, Service Manager will not know to search for matching values in the CI record.

Note: Not all UCMDB attributes have a corresponding field in Service Manager. You may need to tailor your Service Manager system to add a matching field if one does not already exist.

Using join tables for reconciliation

When setting reconciliation rules, if the device type you are reconciling has a joindef definition (as defined in the **devtype** table), use the join table name instead of the **device** table. For example, if you want to reconcile computer CIs, use the **joincomputer** table instead of the **device** table.

Sequence of reconciliation

A reconciliation rule specifies what Service Manager table and field you want to search for matching CI values. It also specifies the sequence in which you want Service Manager to process reconciliation rules. By default, Service Manager processes rules in alphabetical order by field name. For example, Service Manager will reconcile CIs against the **asset.tag** field before reconciling CIs on the **ci.name** field.

To change the order in which Service Manager reconciles CIs, you can add a numeric value to the sequence field. For example, the following reconciliation rules ensure that Service Manager processes CIs by the **ci.name** field prior to reconciling them against the **asset.tag** field.

Sample reconciliation rules ordered by sequence

Table Name	Field Name	Sequence
joincomputer	ci.name	1
joincomputer	asset.tag	2

A Discovery Event Manager (DEM) reconciliation rule allows you to specify which Service Manager fields you want to use to determine if an existing CI record matches a CI in a UCMDB system. An administrator typically specifies reconciliation rules prior to starting UCMDB data push jobs so that Service Manager will not create duplicate CI records.

To create a DEM reconciliation rule:

1. Log in to Service Manager as a system administrator.
2. Navigate to **Tailoring > Web Services > DEM Reconciliation Rules**. Service Manager displays the DEM Reconcile Record form.
3. In Table Name, type the name of the Service Manager table containing the field you want to reconcile on.
4. In Field Name, type the name of the Service Manager field containing the values you want to reconcile on.
5. In Sequence, type a number to specify what order you want Service Manager to run this rule.

   Note: If you do not specify a sequence value, Service Manager will process field names alphabetically.

6. Click **New**. Service Manager creates the reconciliation rule.

How to Add Discovery Event Manager Rules

Service Manager uses Discovery Event Manager (DEM) to define the actions the system should perform when the actual state of an incoming configuration item (CI) record differs from the managed state of a CI record in HP Service Manager. The DEM rules allow you to define whether the Service Manager system adds, updates, or deletes CI records based on incoming UCMDB data.

For CI records only, the DEM rules also allow you to define how Service Manager should handle duplicate logical names.

To access DEM rules in Service Manager, navigate to **Tailoring** > **Web Services** > **Discovered Event Manager Rules**, and then click **Search** to view existing rules or click **New** to create new rules.

This section includes:

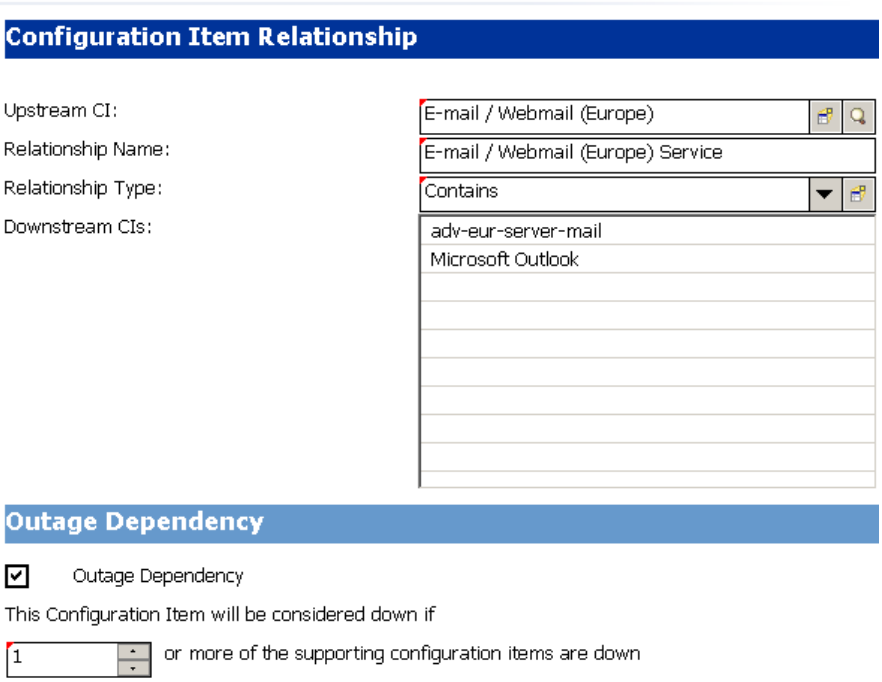
DEM Rules

Service Manager offers the following rules options:

Action if matching record does not exist

This is the action you want Service Manager to take if it cannot find a matching CI record.

- **Add the record:**(Default) Service Manager will add a CI record when it cannot find a matching record. See "How to Add DEM Reconciliation Rules" on page 134 to define what fields Service Manager uses to match CI records.
- **Add the record, and set dependency as true**: This option is available only for synchronization of CI relationship data. Service Manager adds the CI relationship record and enables outage dependency for the record by doing the following:
  - Checks the Outage Dependency check box;
  - Sets the number of dependent downstream CIs to 1. This is because UCMDB only supports one-to-one CI relationships.



- **Open an Incident:**Service Manager opens an incident for someone to review the new CI record. The incident enables someone to investigate whether the new CI record is compliant with your business

practices.

- **Open a Change:**Service Manager opens an unplanned change for someone to review the new CI record. The change allows you to investigate whether the new CI record is compliant with your business practices. If the CI record is compliant, the change can be approved. If the CI record is not compliant, then the change can be denied and the CI record removed. The change record lists both the current and proposed attribute values.

Action if record exists but unexpected data discovered

This is the action you want Service Manager to perform if it does not find a matching CI attribute value.

- **Open a Change:** (Default) Service Manager opens an unplanned change to review the actual state of the CI record. The change allows someone to investigate whether the new attribute value is compliant with your business practices. If the value is compliant, the change can be approved. If the value is not compliant, then the change can be denied and the CI attribute value reverted to its managed state.
- **Log Results and update record:**Service Manager logs the results of the actual state of the CI record, and then updates the CI record.
- **Open an Incident:**Service Manager opens an Incident to investigate the actual state of a CI record and determines what actions must be performed or initiated to bring the record into compliance with Service Manager.

Action if record is to be deleted

This is the action you want Service Manager to perform if an external event specifies that the record needs to be deleted.

- **Delete record:** (Default for CI Relationship records) This option is available for synchronization of both CI and CI Relationship records. Service Manager automatically deletes the CI/CI Relationship record.
- **Open an Incident:** This option is available only for synchronization of CI Relationship records. Service Manager opens an incident to investigate the deleted record and determines which actions must be performed or initiated to bring the record into compliance with Service Manager.
- **Open a Change:** This option is available only for synchronization of CI Relationship records. Service Manager opens an unplanned change to review the deleted record. The change allows someone to

investigate whether the deleted record is compliant with your business practices. If the record is compliant, the change can be approved. If the record is not compliant, then the change can be denied and the record added back to the system.

- **Update record to the selected status:** (Default for CI records) This option is available only for synchronization of CI records. Service Manager updates the status of the CI record to a value selected from the drop-down list (for example, **Retired/Consumed**), rather than delete the record permanently.

  Note: Values available from the drop-down list are defined in the **ICM Status** global list.

- **Open a Change to update record to the selected status:** This option is available only for synchronization of CI records. Service Manager opens an unplanned change to update the CI record's status to a value selected from the drop-down list (for example, **Retired/Consumed**). The change allows someone to investigate whether the requested status change is compliant with your business practices. Once the change has been approved and closed, Service Manager automatically changes the CI record to the selected status. If the change has been denied, Service Manager makes no changes to the CI record.

- **Open an Incident to update record to the selected status:** This option is available only for synchronization of CI records. Service Manager opens an incident to update the record's status to a value selected from the drop-down list (for example, **Retired/Consumed**). Once the incident has been closed, Service Manager automatically updates the CI record to the selected status.

Duplication Rules

UCMDB may create two completely separate yet legit CI records that happen to have the same "name." The UCMDB name field is mapped to the logical.name field (which must be unique) in Service Manager. Pushing the two CI records to Service Manager would cause a duplicate logical name problem. You have several ways to avoid this problem. See the following table.

Solutions to the duplicate logical name problem

Product side	Solution
UCMDB	Change the names directly in UCMDB or change the UCMDB reconciliation rule to make sure the names are not the same. This is highly recommended.

	In the integration adapter mapping configuration (xslt) file, avoid mapping the UCMDB name field to the SM logical name field directly in either of these ways:
	• Map another UCMDB unique attribute to the SM logical.name field, and map the UCMDB name field to another SM field;
	• Add a prefix to the name. The following are examples.
	  ■ UCMDB switches or routers are simply named as “Router” or “Switch” and identified by their underlying MACs. You can configure their “SM logical name” to be “<MAC> + <name>”.
	  ■ UCMDB databases often have the same name (due to the implementation of clusters and Oracle RACs). You can configure their “SM logic name” to be “<full DNS name> + <name>”.

Service Manager	Use the duplication rule options in DEM Rules in Service Manager.

Service Manager offers the following duplication rule options on the Duplication Rule tab in each DEM rule with a Table Name other than “cirelationship”:

- **Action if logical name is duplicated (CI with different uCMDB ID)**: This is the action you want Service Manager to perform if the logical name is already used by another CI record when a CI record is added or updated.
  - **Rename to <name>_[RENAMED]_1/2/3**: (Default) Service Manager changes the logical name by adding a suffix.
  - **Return Error**: Service Manager returns a duplicate key error to UCMDB.

CI Attributes Displayed in Change and Incident Records

Service Manager displays a Change Details section on the corresponding change or a CMDB Changes section on the corresponding incident when you configure DEM to open either change records or incident records when it discovers CI attribute changes through the UCMDB-SM integration. Service Manager only displays a tab for CI attributes when the UCMDB-SM integration is enabled and you have

defined a rule in the Discovery Event Manager to create a change or incident record when a CI is added, updated, or deleted.

Both the Change Details and CMDB Changes sections display the current CI attribute values alongside the actual attribute values discovered by UCMDB. You can use this information to approve or deny a change or escalate an incident to the proper assignment group.

Searching for Change and Incident Records Opened by the Integration

You can use the following search criteria to find change and incident records opened by the UCMDB-SM integration.

Search options available for change and incident records

Record type	Search option available
Change	Search for records with the category **unplanned change**.
Incident	Search for records using the **Generated by UCMDB Integration** option.

How to Add a CI Attribute to the Integration for Data Push

You can use the following steps to add a CI attribute to the integration.

1. Does the CI attribute already exist in the UCMDB class model?
   Yes. Go to Step 3.

   No. Go to Step 2.

2. Add the CI attribute to the UCMDB class model.
   See "How to Add the CI Attribute to the UCMDB Class Model" on the next page.

3. Add the CI attribute to the query layout.
   See "How to Add a CI Attribute to the Query Layout" on page 143.

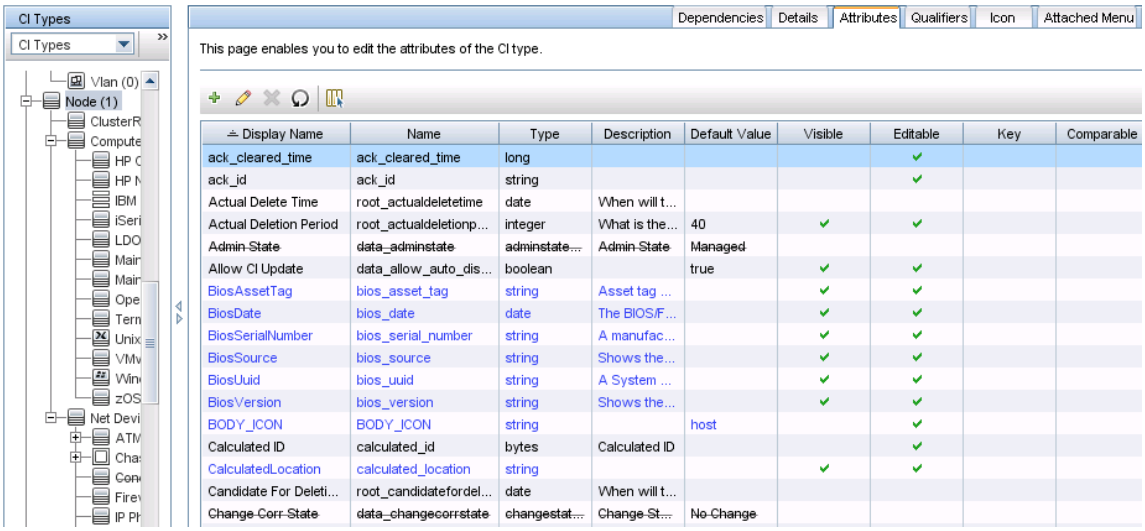
4. Add a web service field for the Service Manager CI type.
   See "How to Add a Web Service Field for the Service Manager CI Type" on page 145.

5. Map the CI attribute to the Service Manager web service field.

   See "How to Map the CI Attribute to a Service Manager Web Service Field" on page 153.

How to Add the CI Attribute to the UCMDB Class Model

The integration only uses a subset of the CI attributes available from your UCMDB system. Out-of-the-box, the integration consists of CI attributes that are typically managed from a Service Manager system such as host name and host DNS name. Before creating a new UCMDB CI attribute, you should determine if there are any existing CI attributes in your UCMDB system that provide the data you want. In most cases, there is an existing attribute tracking the data that you want to add to the integration. For example, the Node CI type contains many attributes that you can add to the integration.



The following steps illustrate how to add a new CI attribute to an existing CI type. This is not the expected typical scenario. Typically, you would add an existing CI attribute to the integration, which means you do not need to perform these steps.

Note: The integration does not require any special steps to add a CI attribute to the UCMDB class model. You can use the standard CI attribute creation procedures to add a CI attribute. For more information on CI attribute creation, see the UCMDB Help Center.

As an example, the following steps will add an attribute named **comments** to the Business Service CI type.

To add a CI attribute to the UCMDB class model:

1. Log in to UCMDB as an administrator.
2. Navigate to **Modeling** > **CI Type Manager**.

3. Select the CI type to which you want to add a new CI attribute from the CI Types navigation tree. For example, **ConfigurationItem** > **BusinessElement** > **BusinessService**.
4. Click the **Attributes** tab.
5. Click the **Add** icon. 
The Add Attribute window opens.
6. In Attribute Name, type the unique name you want to use for the new CI attribute. For example, `comments`.

   Caution: The name cannot include any of the following characters: ` ' / \ [ ] : | < > + = ; , ? *`.
7. In Display Name, type the name that you want UCMDB to display in the interface. For example, `Comments`.
8. (Optional) In Description, type a description of the new CI attribute.
9. In Attribute Type, select **Primitive** or **Enumeration/List** depending on the data type of the attribute. For example, select **Primitive** and select **string**.
10. In Value Size, type the maximum character length that the attribute can have. For example, `300`.
11. Select the **Enable default value** option, and enter a default value. Or, do not select this option to leave the default value blank.
12. Click **OK** to save the attribute.
13. Click the **Save** icon  to save attribute changes to the CI type.

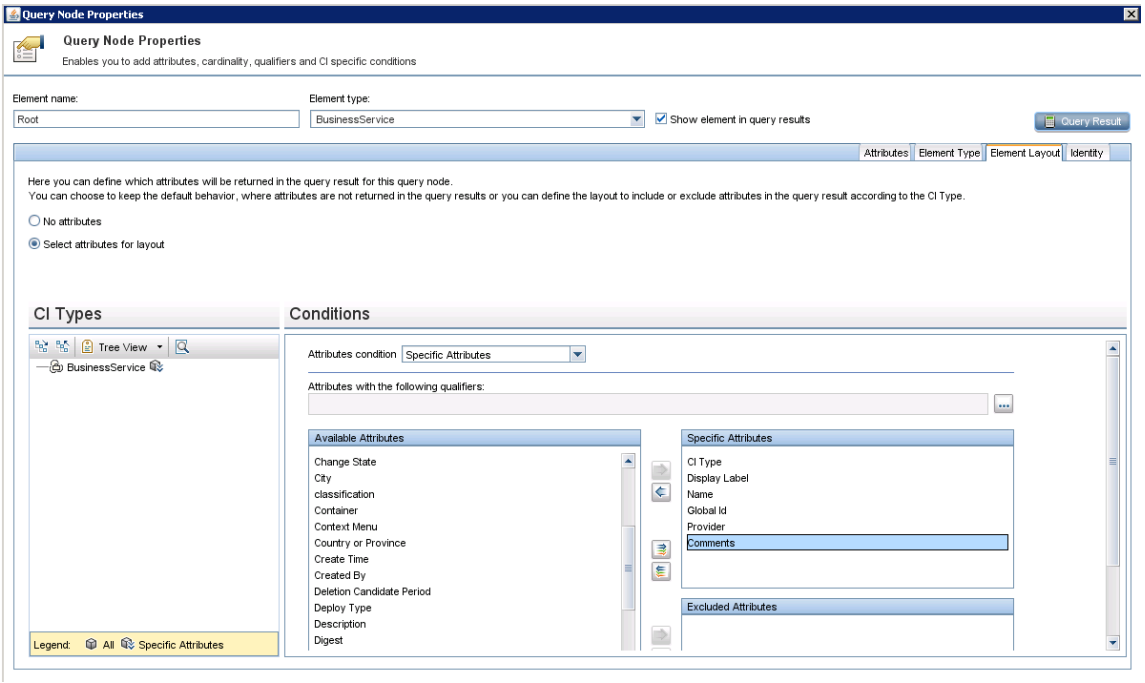
Now, the attribute has been added to the CI type. Next, you need to add the attribute to a query that synchronizes the CI type. See "How to Add a CI Attribute to the Query Layout" below.

How to Add a CI Attribute to the Query Layout

To add an existing CI attribute to the integration, you must add this attribute to the layout setting from the query that synchronizes the CI type. You must know which CI type contains the CI attributes that you want to add to the integration.

To add a CI attribute to the query layout:

1. Log in to UCMDB as an administrator.
2. Navigate to **Modeling** > **Modeling Studio**.
3. For Resource Type, select **Queries**.
4. From the Queries navigation tree, click **Integration** > **Service Manager**.
5. Select the query that manages the CI type whose attributes you want to add to the integration. For example, **Push > SM Computer Push 2.0**. UCMDB displays the TQL layout of the query.
6. Select the node from the query layout that contains the CI attribute that you want to add to the integration. For example, **Root**.
7. Right-click the node and select **Query Node Properties**. The Query Node Properties window opens.
8. Select the CI type (**Database** in this example), and click the **Element Layout** tab.
9. Select the CI attribute that you that want to include in the integration from the **Available Attributes** list, and click the **Add** icon to add it to **Specific Attributes** list. For example, **Comments**.



10. Click **OK** to save the node properties.
11. Click the **Save** icon  to save the query.

Now, the CI attribute has been added to the relevant query layout. Next, you need to add a Service Manager web service field that will be mapped to this UCMDB CI attribute.

How to Add a Web Service Field for the Service Manager CI Type

The integration uses only a subset of the CI attributes available from your Service Manager (SM) system. CI attributes must be mapped between UCMDB and SM. Before creating a new SM CI attribute that will be mapped to the UCMDB CI attribute you added previously, you must determine if there are any existing CI attributes in your Service Manager system that provide the data you want. In most cases, there is an existing attribute tracking the data that you want to add to the integration. For example, the Computer CI type contains many attributes that you can add to the integration.

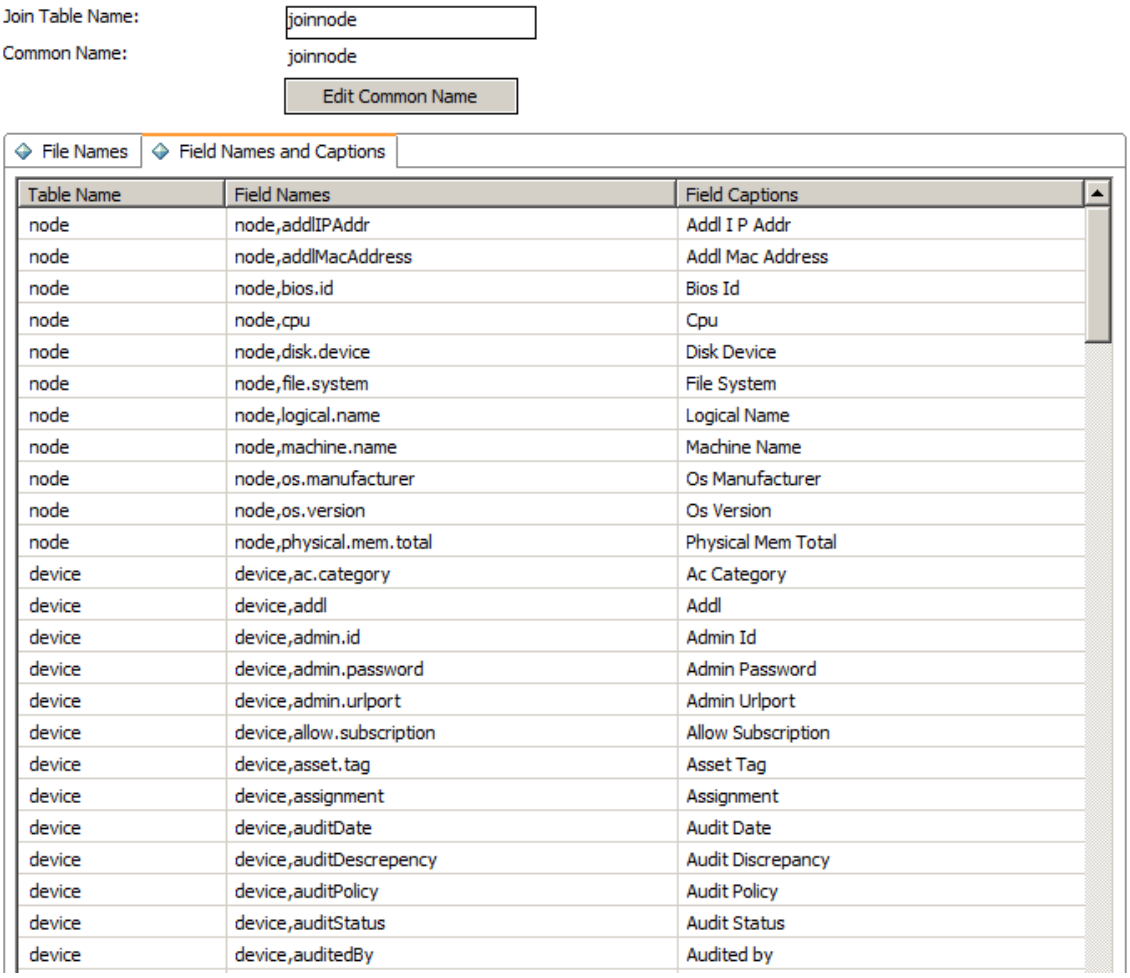


Table Name	Field Names	Field Captions
node	node,addlIPAddr	Addl I P Addr
node	node,addlMacAddress	Addl Mac Address
node	node,bios.id	Bios Id
node	node,cpu	Cpu
node	node,disk.device	Disk Device
node	node,file.system	File System
node	node,logical.name	Logical Name
node	node,machine.name	Machine Name
node	node,os.manufacturer	Os Manufacturer
node	node,os.version	Os Version
node	node,physical.mem.total	Physical Mem Total
device	device,ac.category	Ac Category
device	device,addl	Addl
device	device,admin.id	Admin Id
device	device,admin.password	Admin Password
device	device,admin.urlport	Admin Urlport
device	device,allow.subscription	Allow Subscription
device	device,asset.tag	Asset Tag
device	device,assignment	Assignment
device	device,auditDate	Audit Date
device	device,auditDescrepency	Audit Discrepancy
device	device,auditPolicy	Audit Policy
device	device,auditStatus	Audit Status
device	device,auditedBy	Audited by

If you decide to use an existing attribute instead of creating a new one, you can directly modify the web service definition in Service Manager to expose that attribute, and then map this SM attribute to the UCMDB attribute by using the UCMDB Visual Mapping tool (see "How to Map the CI Attribute to a Service Manager Web Service Field" on page 153).

If you decide to add a new attribute, follow the steps below. The steps vary depending on the data type of the attribute: simple (string, for example), or complex (array of structure or structure).

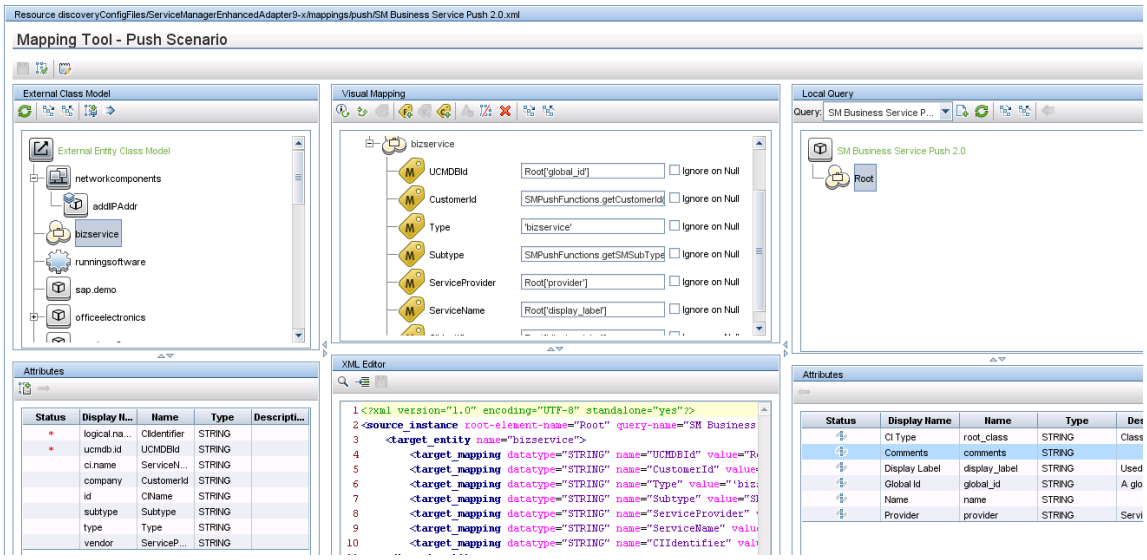
Add a Simple Attribute to the SM CI Type

As an example, the following steps illustrate how to add a simple attribute (**comments**) to an existing CI type (Business Service).

1. Log in to UCMDB as a system administrator.

2. Click **Data Flow Management** > **Adapter Management**, and then select **ServiceManagerEnhancedAdapter 9-x** to open the corresponding XML mapping file (**SM Business Service 2.0.xml** in this example) with the Visual Mapping tool editor.

3. On the External Class Model pane, select the CI type (**bizservice**).

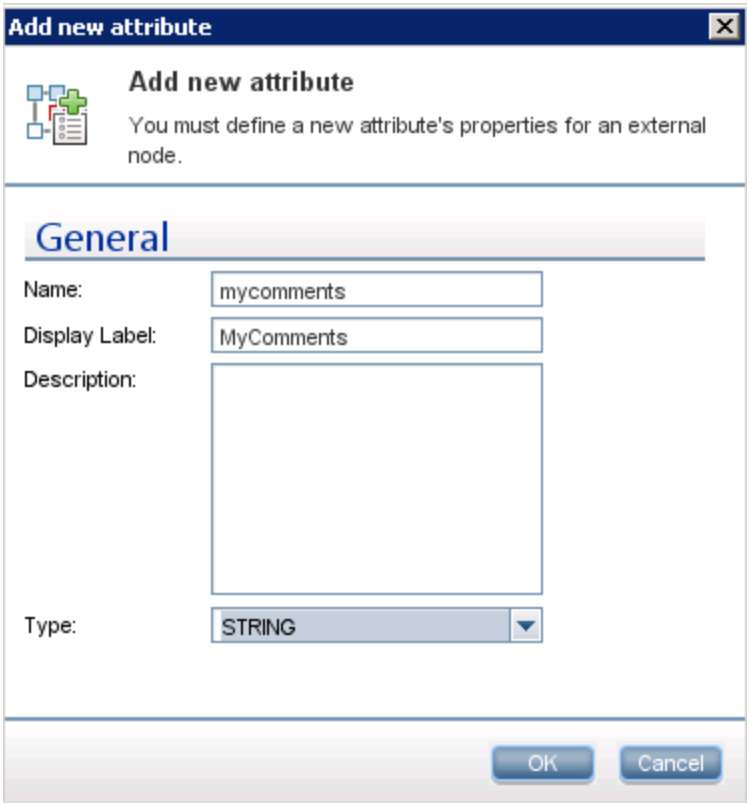


4. On the left-side Attributes pane (which displays the fields that are retrieved from the Service Manager web service object for the Business Service CI type), click the **Add New Attribute to Selected External Node** icon.

5. Type a name and display label for the new attribute, and click **OK**.

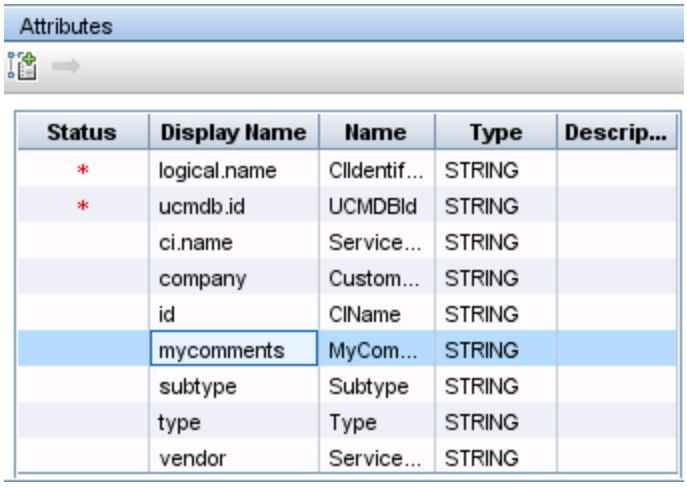
> **Caution:** If the you enter a name that duplicates or is part of an existing field name in the attribute table of the Service Manager CI type or in the **device** table, an error will occur

indicating the attribute could not be created.



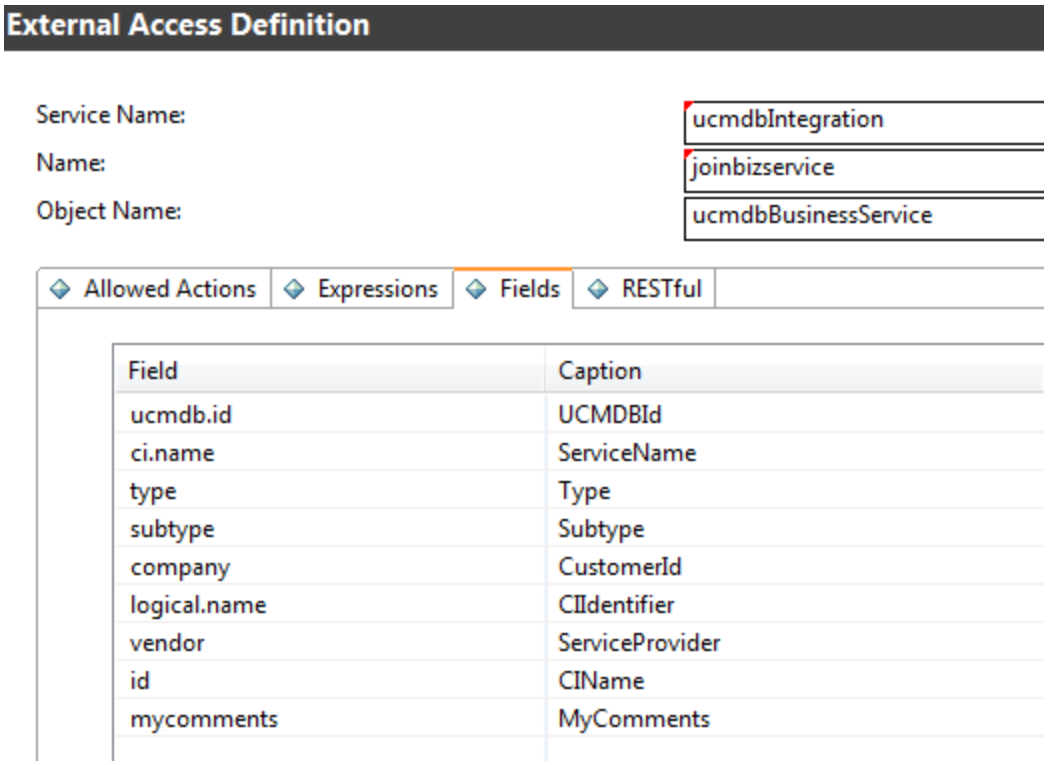
6. Click OK again to confirm the attribute creation.

   The new attribute (**mycomments**) appears in the Service Manager attribute list.



Status	Display Name	Name	Type	Descrip...
*	logical.name	CIIdentif...	STRING	
*	ucmdb.id	UCMDBId	STRING	
	ci.name	Service...	STRING	
	company	Custom...	STRING	
	id	CIName	STRING	
	mycomments	MyCom...	STRING	
	subtype	Subtype	STRING	
	type	Type	STRING	
	vendor	Service...	STRING	

At the same time, UCMDB invokes the Service Manager Web Service API to add the new attribute, expose it in the Web Service API and set it as a managed field. You can open the web service object in Service Manager to verify the new attribute (**Tailoring** > **Web Services** > **Web Service Configuration**).



7. Save the configuration file.

Add an Array of Structure or Structure to the CI Type

Service Manager uses an Array of Structure or Structure to store complex attributes, for example, the ports of a Computer CI. You can also create such kind of complex attributes through the Visual Mapping tool in UCMDB.

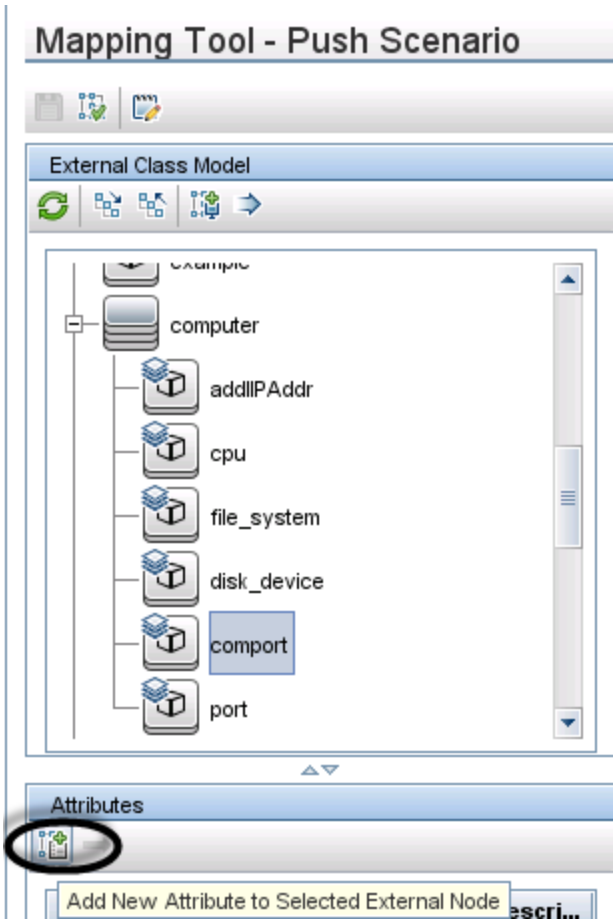
The following steps illustrate how to add an attribute named **comport** to the Computer CI type.

1. Log in to UCMDB as a system administrator.
2. Open the corresponding XML mapping file (**SM Computer Push 2.0.xml** in this example), with the Visual Mapping tool editor.
3. Select the CI type (**computer**), and click the **Add New CI Type to External Class Model** icon.

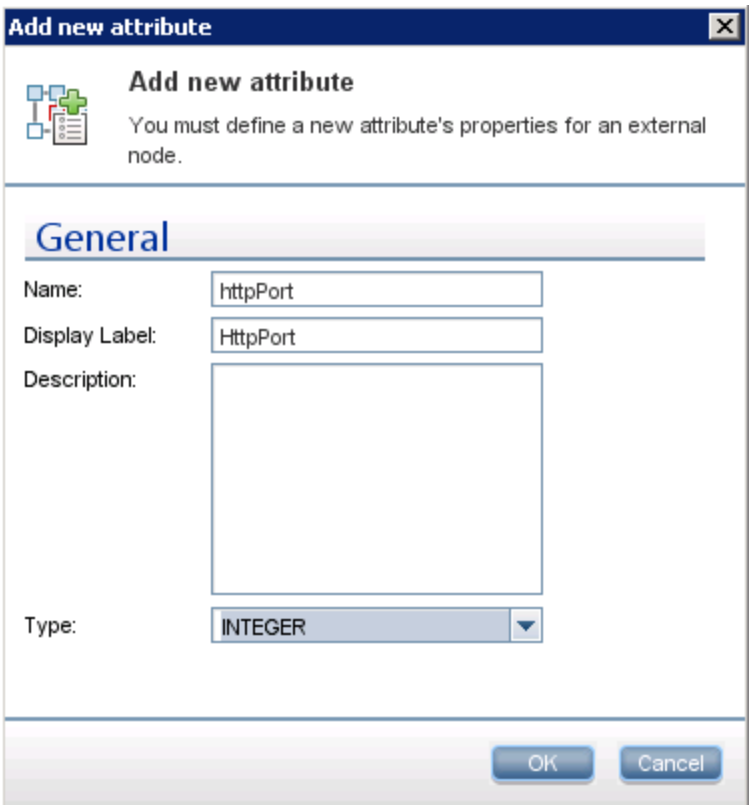
4. In the child CI type creation dialog that appears, enter a name and description for the attribute, select **Many to One** for **Relation with Parent** if the attribute data type is array of structure or select **One to One** if the data type is structure.

5. Click **OK**. The attribute is created on the Service Manager side.

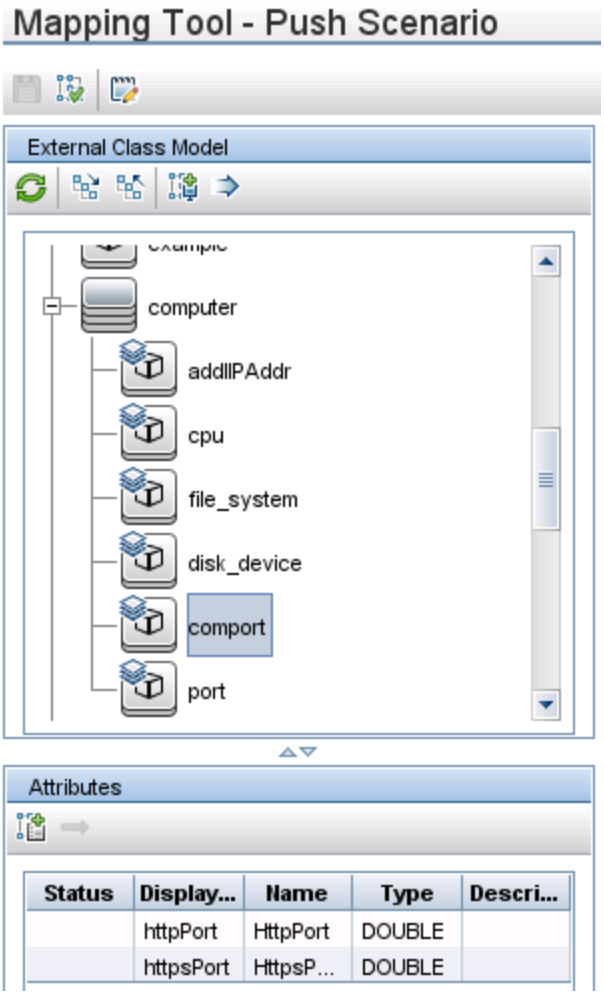
6. Select the new attribute in the External Class Model pane, and click the **Add New Attribute to Selected External Node** icon.



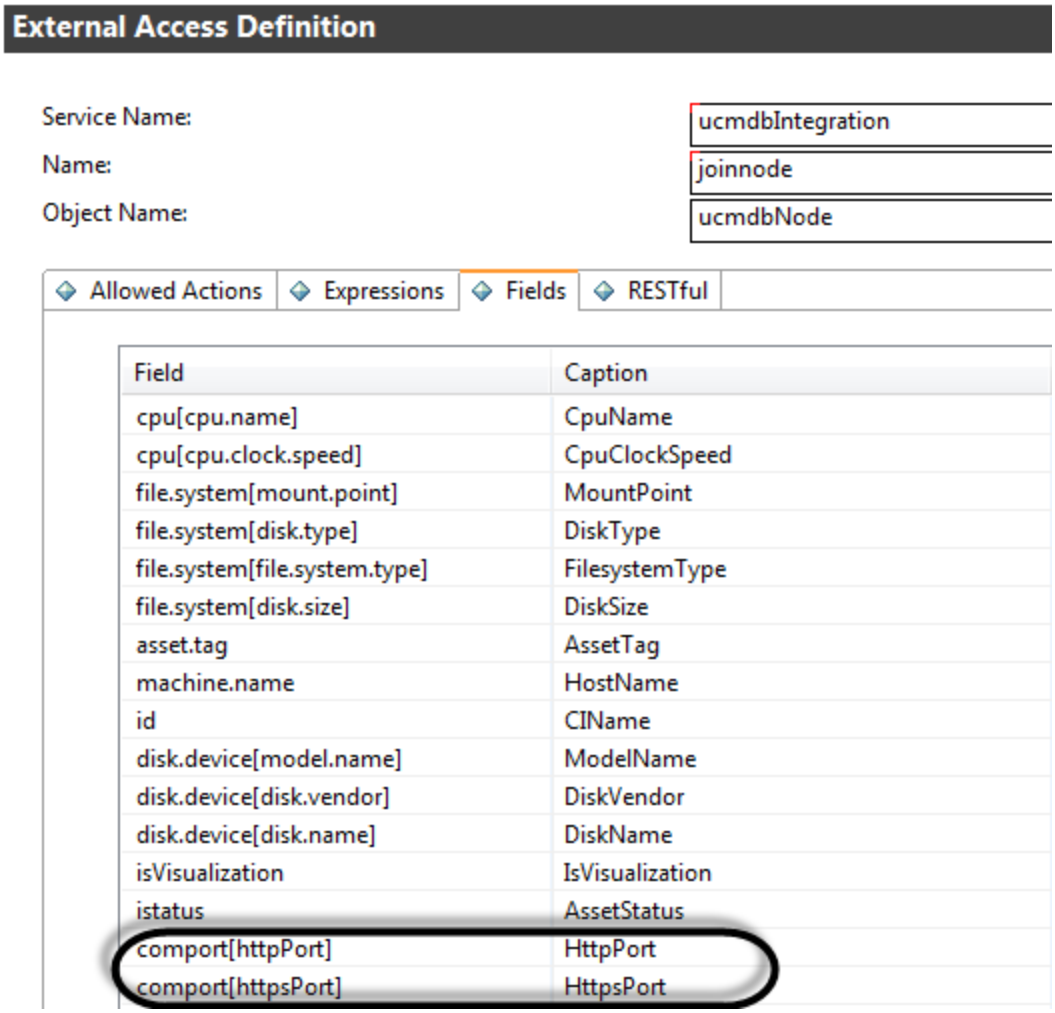
7. Add a field (**httpPort**) to the array of structure, and click **OK**. Click **OK** again to confirm the attribute creation.



8. Repeat the previous two steps to add more fields to the array of structure as needed.



The attribute is also automatically added to the corresponding web service object in Service Manager.



9. Save the configuration file.

Next, you need to map this Service Manager CI type attribute to the one you added previously in UCMDB.

How to Map the CI Attribute to a Service Manager Web Service Field

The integration uses an adapter to transform UCMDB CI attributes to web services objects recognized by Service Manager. The adapter in turn specifies what XML transformation files the integration should use to convert UCMDB queries into properly formatted Service Manager web services messages.

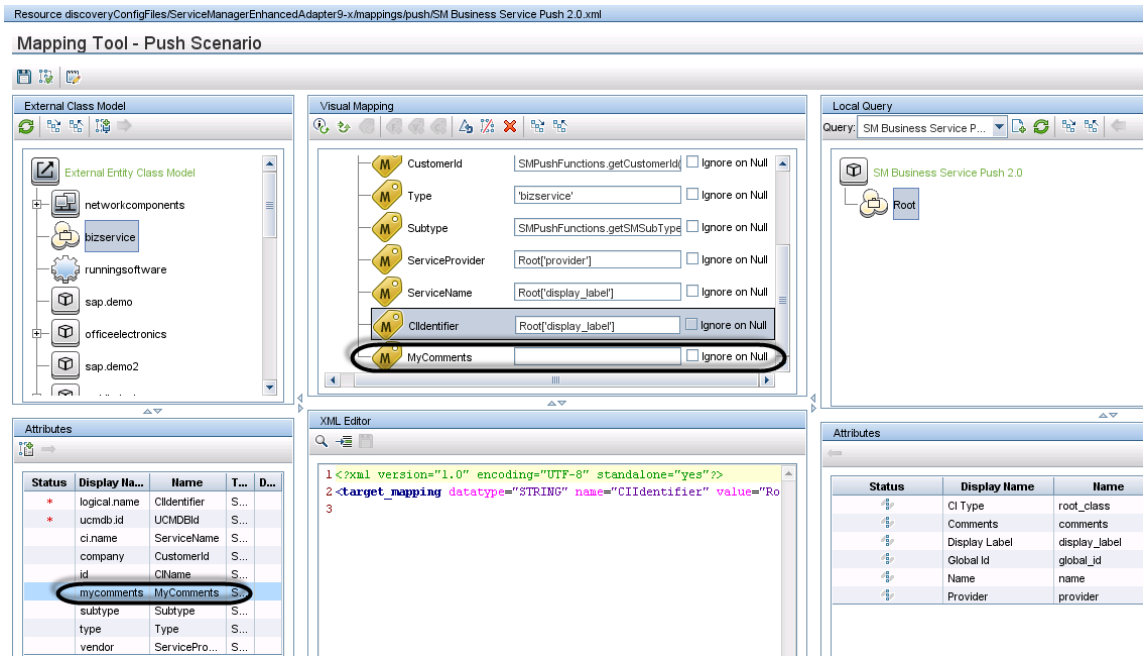
Out-of-the-box, each integration query has a corresponding XML configuration file that maps to a particular CI type in UCMDB. In addition, each attribute for which you enabled calculation requires its own entry in the XML configuration file. Without an XML transformation entry, Service Manager cannot receive any CI attribute updates from your UCMDB system.

If you want to add a new attribute to the integration, you must edit the XML configuration file for the parent CI type and add an entry for the CI attribute. For information about which CI types each query manages, see "Queries for Push" on page 114.

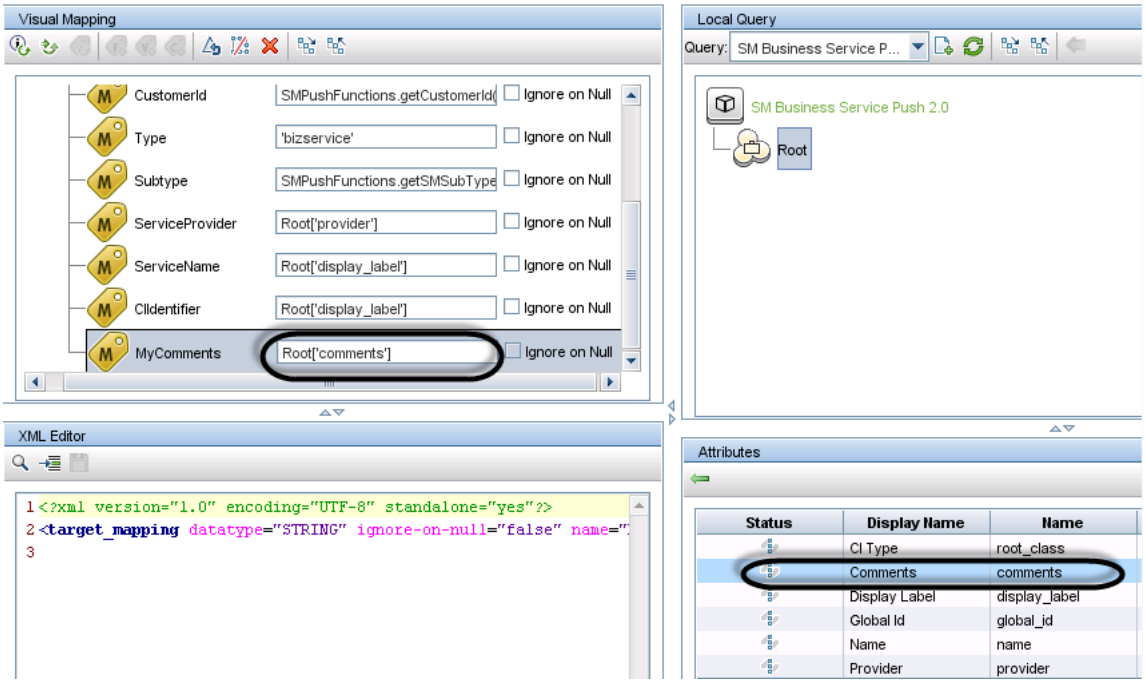
The following steps illustrate how to map UCMDB CI attribute **Comments** (which you added previously) to the **MyComments** Service Manager attribute (which you created in the previous step).

To map a UCMDB CI attribute to an SM CI attribute:

1. Log in to UCMDB as an administrator.
2. Navigate to **Data Flow Management** > **Adapter Management** > **ServiceManagerEnhancedAdapter9-x** > **Configuration Files**.
3. Double-click the XML configuration file that manages the parent CI type of your CI attribute. For example, open **SM Business Service Push 2.0.xml** to add an attribute to the **SM Business Service Push 2.0** query.
4. Select the relevant CI type in the External Class Model pane (**bizservice** in this example), and drag and drop the attribute (**mycomments**) from the left-side Attributes pane into the Visual Mapping pane.



5. Select the relevant Node in the Local Query panel (**Root** in this example), and drag and drop the attribute (**comments** in this case) into the mapping entry that you created in the previous step.



6. Save the XML configuration file. The **Comments** attribute in UCMDB is now mapped to the **MyComments** attribute in Service Manager.

> **Note:** When you create or edit and then save a configuration file in Adapter Management, UCMDB automatically restarts the adapter with the new configuration file.

How to Add a CI Type to the Integration for Data Push

You can use the following steps to add a CI type to the integration.

1. Does the CI type already exist in the UCMDB class model?
   Yes. Go to Step 3.

   No. Go to Step 2.

2. Add the CI type to the UCMDB class model.
   See "How to Add the CI Type to the UCMDB Class Model" on the next page.

3. Create a query to synchronize the CI type.
   See "How to Create a Query to Synchronize the CI Type" on page 159.

4. Add the CI type's attributes to the query layout.
   See "How to Add the CI Type's Attributes to the Query Layout" on page 163.

5. Add the CI type to Service ManagerService Manager.
   See "How to Add the CI Type to Service Manager" on page 165.

6. Map the CI type's attributes to web service fields.
   See "How to Map the CI Type's Attributes to Web Service Fields" on page 168.

7. Add custom queries to integration data push jobs.
   See "How to Add a Custom Query to an Integration Job" on page 179.

How to Add the CI Type to the UCMDB Class Model

Before creating a new UCMDB CI type, you should determine if there are any existing CI types in your UCMDB system that provide the CI attributes you want. In most cases, you can create links to one or more existing CI types to create a new logical CI type for use by the integration.

The following steps illustrate how to create a new CI type called SM RDBMS based on an existing CI type called database.

Note: The integration does not require any special steps to add a CI type to the UCMDB class model. You can use the standard CI type creation procedures to add a CI type. For more information on CI type creation, see the UCMDB Help Center.

To add a CI type to the UCMDB class model:

1. Log in to UCMDB as an administrator.

2. Navigate to **Modeling** > **CI Type Manager**.

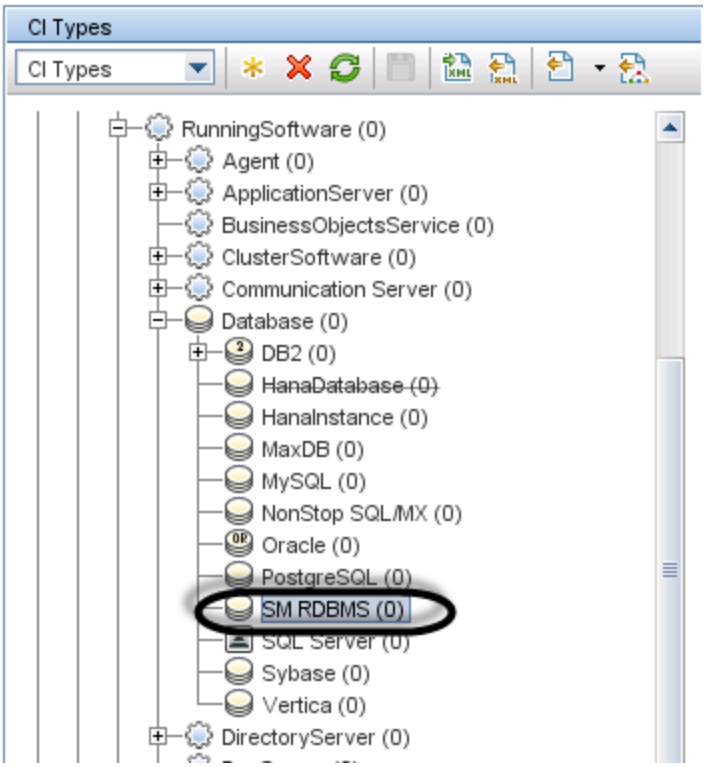
3. Select the base CI type you want to use for your new CI type from the CI Types navigation tree: **Managed Object** > **ConfigurationItem** > **Infrastructure Element** > **Running Software** > **Database**.

4. Click the **New** icon  .
   The Create Configuration Item Type window opens.

5. In Name, type the unique name you want to use for the new CI type. For example, `sm_rdbms`.

   Caution: The name cannot include any of the following characters: ` ' / \ [ ] : | < > + = ; , ? *.

6. In Display Name, type the name you want UCMDB to display in the interface. For example, `SM RDBMS`.
7. In Description, type a description of the new CI type. This is an optional field. For example, `Hosts running relational databases`.
8. In Base CI Type, verify that the proper base CI type is selected. Your new CI type will inherit the attributes of the base CI type you select here. For example, **Database**.
9. Click **Next**. The wizard displays a list of CI attributes from the base CI type.
10. Add, edit, or remove CI attributes as needed for the new CI type. For example, accept the default attributes inherited from **Database**.
11. Click **Next**. The wizard displays a list of qualifiers from the base CI type.
12. Add or remove qualifiers as needed for the new CI type. For example, accept the default qualifiers.
13. Click **Next**. The wizard displays a list of icons associated with the CI type.
14. Select the icons associated with this CI type. For example, accept the default abstract class icon.
15. Click **Next** to add any menu item properties or label definitions as needed. For example, accept the default settings from the base CI type.
16. Click **Finish** to create the CI type.



17. Select your new CI type from the tree. For example, **SM RDBMS**.

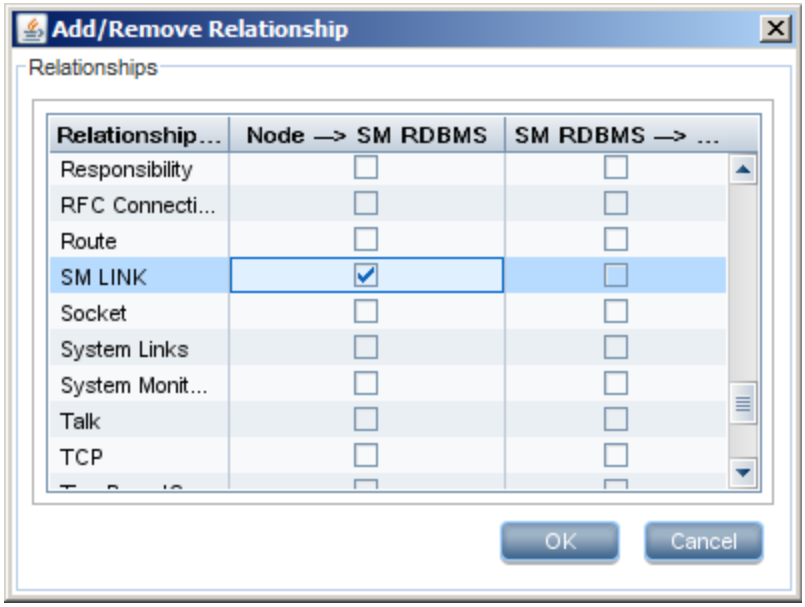
18. Browse to an existing CI type you want to link to, and control-click it to add it to your selection. For example, **Node**.

    Note: Choose an existing CI type that has the attributes that you want to be part of your new logical CI type.

19. Right-click one of the selected CI types, and click **Add/Remove Relationship**. The Relationships window opens.

20. Create an SM Link relationship from the existing CI type to the new CI type. For example, from **Node** to **SM RDBMS**.

    Note: You need to create a new SM Link relationship if it does not exist.



21. Click **OK** to create the relationship.

22. Click the **Save** icon to save the CI type.

How to Create a Query to Synchronize the CI Type

The integration uses queries to gather CI attribute values and pass them to your Service Manager system. You must create a query for any CI type you add to the integration. Any query you create must conform to the "Query Requirements" on page 119.

The following steps illustrate how to create a new query named **rdbmsData** for the **SM RDBMS** CI type described in previous sections.

1. Log in to UCMDB as an administrator.

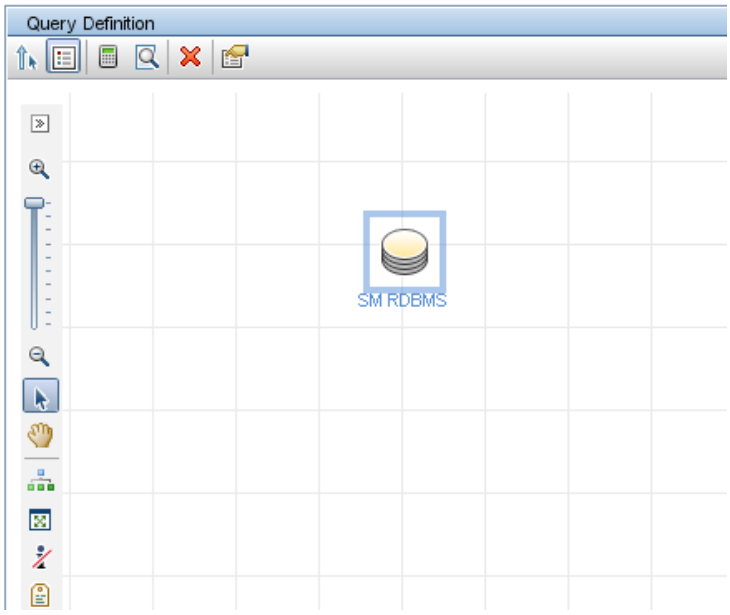
2. Navigate to **Modeling** > **Modeling Studio**.

3. From the Queries navigation tree, click **Integration** > **Service Manager**.

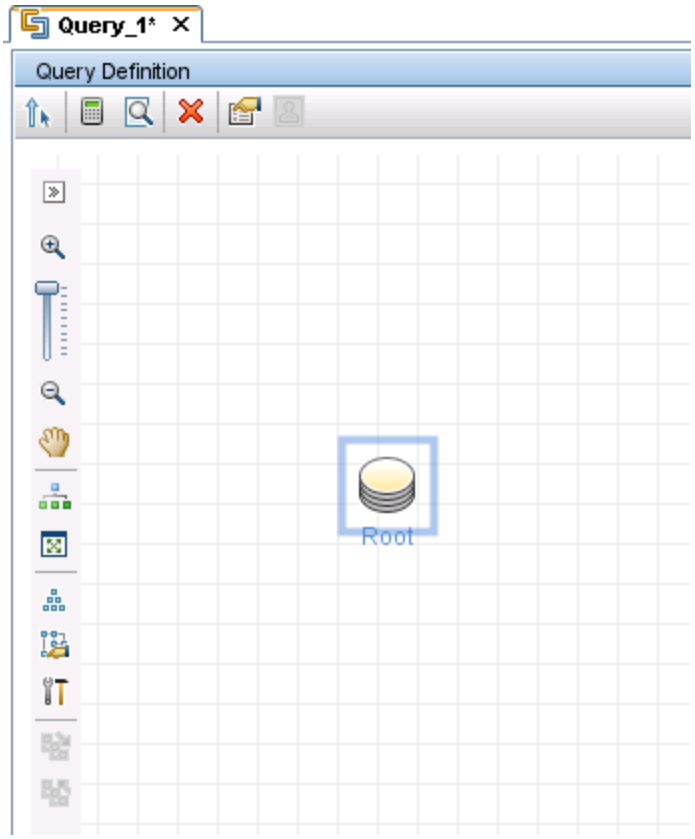
4. Right-click **Service Manager**, and select the **New** > **Query**.
   The Query Definition window opens.

5. Find the CI type that will be the root node of your query from the CI Type Selector. This CI type is typically the one that provides the most attributes for the CI. For example, **Managed Object** > **ConfigurationItem** > **InfrastructureElement** > **RunningSoftware** > **Database** > **SM RDBMS.**

6. Drag the root CI type from the CI Type Selector and drop it into the empty Editing pane. UCMDB displays the icon of the CI type.

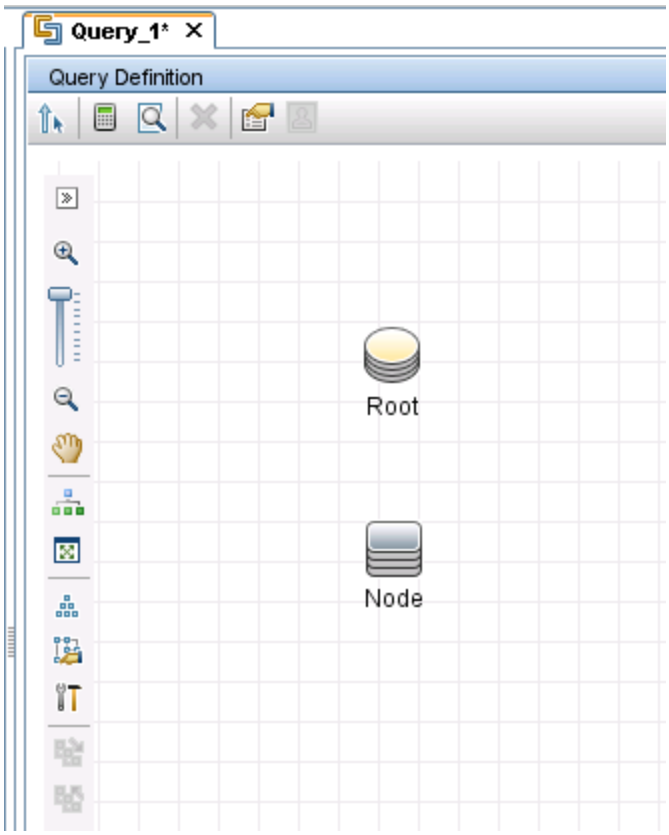


7. Right-click the CI type icon, and select **Query Node Properties**. The Node Properties window opens.
8. Change the Element Name to **Root**.
9. Click **OK** to save the node properties.



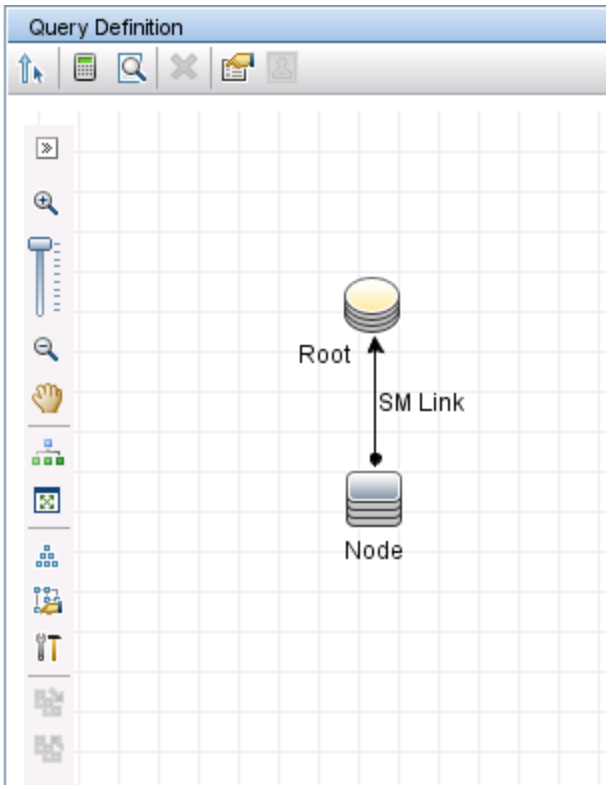
10. Find any additional CI types you want to add to the query from the CI Type Selector. These CI types typically provide additional CI attributes. For example, **Managed Object** > **ConfigurationItem** > **Infrastructure Element** > **Node**.

11. Drag the additional CI type from the CI Type Selector and drop it into the Query Definition pane. UCMDB displays the icon of the additional CI type.



12. Create relationships between the Root CI type and the additional CI types as needed. For example, create an SM Link between **Root** and **Node**.
    a. Select **Root** and control-click the additional CI type. For example, **Node**.
    b. Right-click one of the selected items, and click **Add Relationship**. The Add Relationship window opens.
    c. Type a Relationship Name. For example `SM Link`.
    d. Select the relationship direction.

e. Click **OK** to add the relationship.



13. Repeat step 10 to step 12 for each additional CI type you want to add to the query. For example, SM RDBMS does not need any additional CI types.

14. Click the **Save** icon to save the query.
    a. In Query Name, type the unique name you want to use for the new query. For example, `rdbmsData`.

    b. In the folder tree, select the folder in which you want to save the query. For example, **Root** > **Integration** > **Service Manager** > **Push**.

    c. Click **OK**. UCMDB adds your new query to the query list.

How to Add the CI Type's Attributes to the Query Layout

To add a CI attribute to the integration, you must enable the calculation layout setting from the query that synchronizes the CI type. Because you must enable calculation for each attribute you want to add to the integration, you should be familiar with the integration CI types and their attributes.

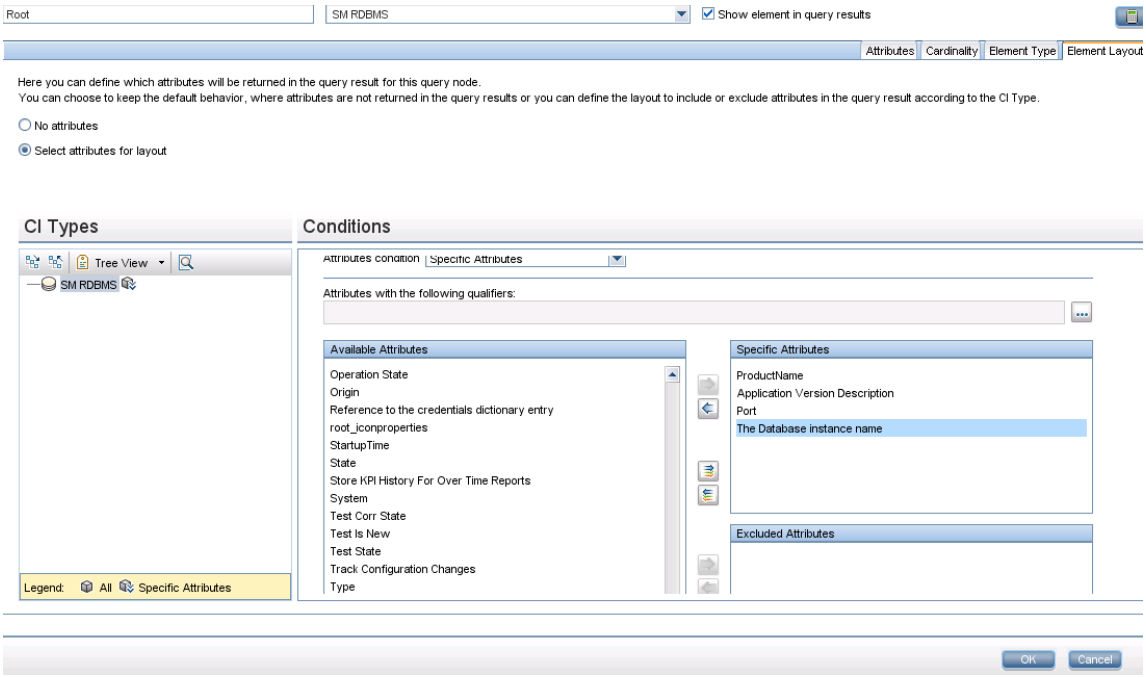
The following steps illustrate how to enable calculation for attributes of the **SM RDBMS** CI type described in previous sections.

To add a CI type's attributes to the query layout:

1. Log in to UCMDB as an administrator.
2. Navigate to **Modeling** > **Modeling Studio**.
3. From the Queries navigation tree, click **Integration > Service Manager**.
4. Double-click the query that manages the CI type whose attributes you want to add to the integration. For example, **rdbmsData**. UCMDB displays the TQL layout of the query.
5. Right-click the **Root** node from the query layout, and then select **Query Node Properties**. The Query Node Properties window opens.

   Caution: Your integration query must contain a node called **Root**. See "Query Requirements" on page 119 for more information.

6. Click the **Element Layout** tab, and select the **Select attributes for layout** option.
7. Select **Specific Attributes** from the **Attributes condition** list, and from the Available Attributes list select the attributes you want and move them to the Specific Attributes list. For example, add the **Product Name**, **Application Version Description**, **The Database Instance Name**, and **Port** attributes.



8. Click **OK** to save the query node properties.
9. Select any additional node that contains the attributes you want to add to the integration. For example, **Node**.
10. Repeat step 4 through step8 for each additional node.
11. Click **OK** to save the query node properties.
12. Repeat step 9 to step 13 for each additional node that contains the attributes you want to add to the integration.
13. Click the **Save** icon to save the query  .

How to Add the CI Type to Service Manager

Before creating a new Service Manager CI type, you should determine if there are any existing CI types in your Service Manager system that provide the CI attributes you want. In most cases, you can reuse the existing CI types for the integration.

You can add a new CI type to Service Manager by using the Visual Mapping tool in UCMDB, which means you do not have to log out of UCMDB and then log in to Service Manager.

The following steps illustrate how to create a new CI type called **RDBMS**.

Note: This example is provided only as an illustration of the steps. The best practice is to reuse the existing Service Manager CI type **RunningSoftware** to map with UCMDB CI type **SM RDBMS**.

1. Log in to UCMDB as an administrator.
2. Open an existing XML mapping file (**SM Computer Push 2.0.xml**, for example) with the Visual Mapping tool editor.
3. Select the Root node in the External Class Model panel, and click the **Add New CI Type to External Class Model** icon.
4. Enter parameter values as follows.

   Name: rdbms

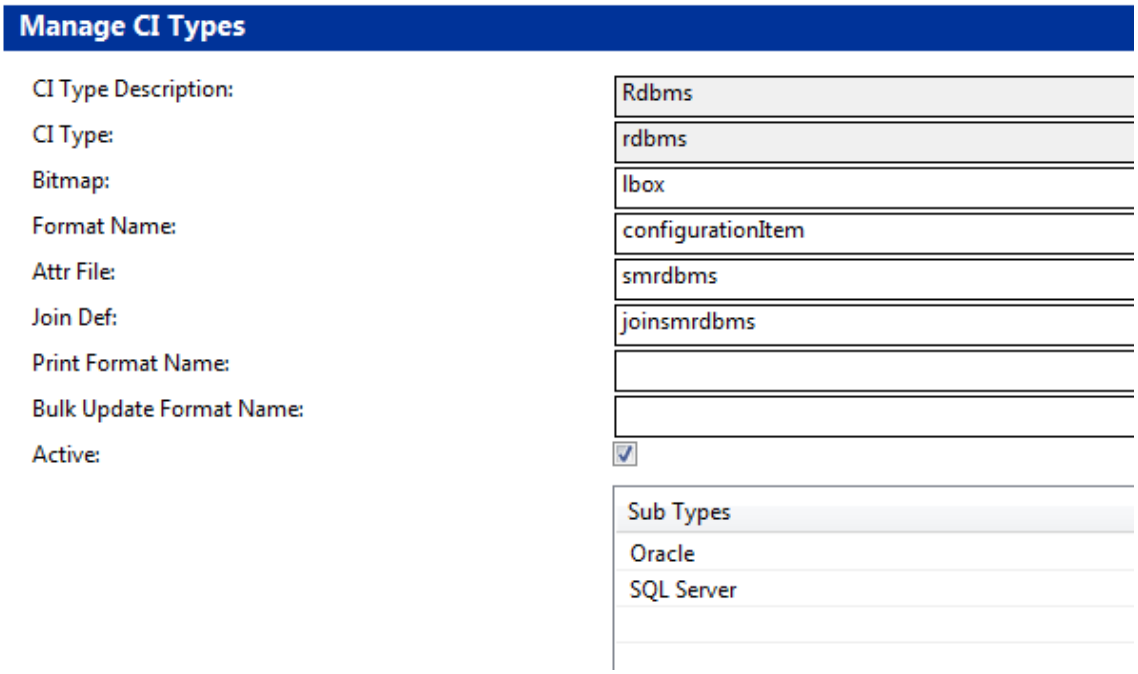
   Description: CI type for RDBMS

   table: smrdbms

   subtype: Oracle, SQL Server

5. Click **OK**.

6. Click **OK** again to confirm the CI type creation. The CI type is automatically created in Service Manager, as shown in the following figure.



What happens on the Service Manager (SM) side?

All relevant SM objects (DBDICT, JoinDef, Web Service API, DEM Rule, and so on) for the new CI type are automatically created on the SM side, except the format. The new CI type will use **configurationItem** as the default format.

If you want to use custom formats in SM for the new CI type, you have to create them manually. You can create the formats in Forms Designer based on existing view forms and bulk update forms. To access Forms Designer in Service Manager, type `fd` in the command line or navigate to **Tailoring** > **Forms Designer**. For more information about creating forms in Service Manager, see the Service Manager online help and the *Tailoring Best Practices Guide*.

How to Map the CI Type's Attributes to Web Service Fields

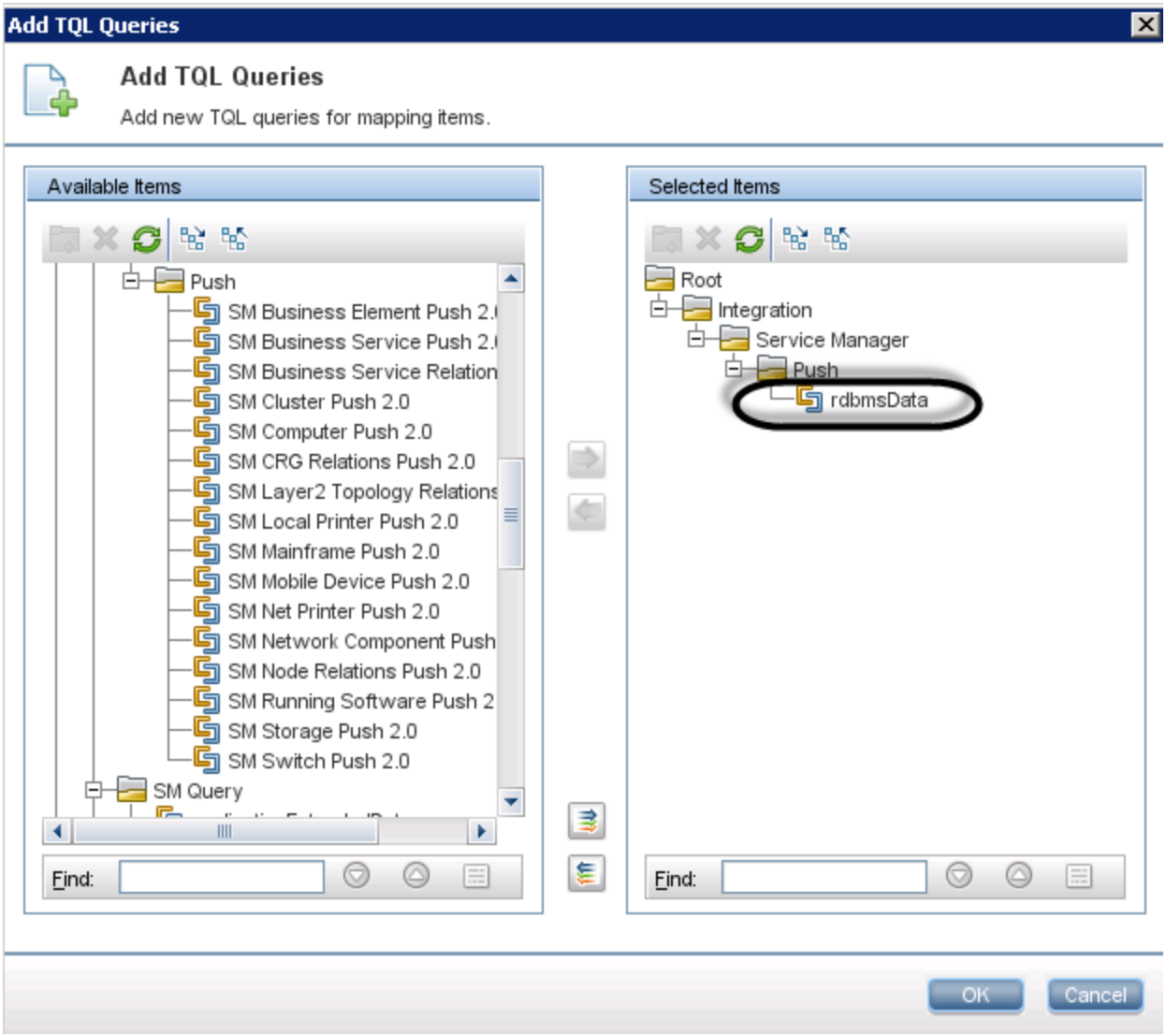
The integration uses the Service Manager Adapter to transform UCMDB CI attributes to web services objects recognized by Service Manager. The Service Manager Adapter uses XML configuration files to convert UCMDB queries into a properly formatted Service Manager web services messages. Out-of-the-box, each integration query has a corresponding XML configuration file. In addition, each attribute you enable for synchronization from advanced layout settings requires its own entry in the XML configuration file.

If you want to add a CI type to the integration, you must create a matching XML configuration file that defines how the Service Manager Adapter transforms each CI type into a Service Manager web service object. See "Integration Queries" on page 113 for information about the CI types each query manages.

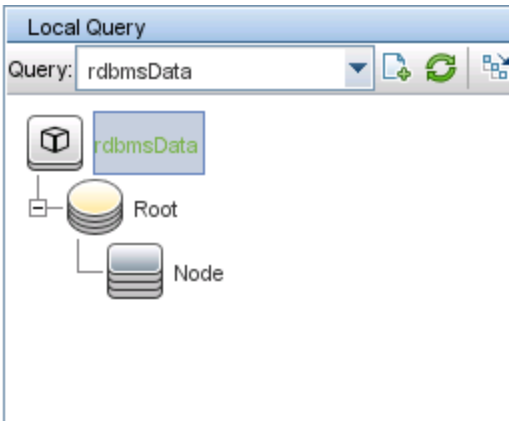
The following steps illustrate how to create an XML configuration file for the rdbmsData query described in previous sections.

To map a CI type's attributes to web service fields:

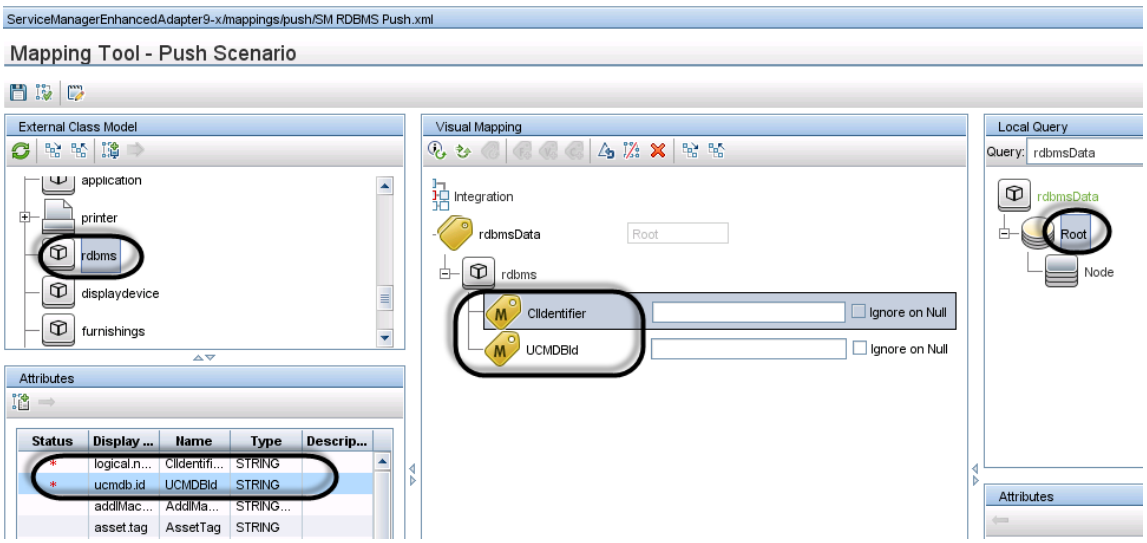
1. Log in to UCMDB with an administrator account.
2. Navigate to **Data Flow Management** > **Adapter Management** > **ServiceManagerEnhancedAdapter9-x**, and select the adapter.
3. Click the **Create New Resource** icon  .
4. Select **New Configuration File**.
5. Enter the full file name: <AdapterID>/mappings/push/<filename>. For example, `ServiceManagerEnhancedAdapter9-x/mappings/push/SM RDBMS Push.xml`.
6. Click **OK**. The mapping file is created.
7. Double-click the new mapping file to open it with the Visual Mapping tool editor.
8. In the Local Query pane, click the **Add TQL Queries** icon to add the **rdbmsData** query.


Add TQL Queries
Add TQL Queries
Add new TQL queries for mapping items.
Available Items
Push
SM Business Element Push 2.
SM Business Service Push 2.
SM Business Service Relation
SM Cluster Push 2.0
SM Computer Push 2.0
SM CRG Relations Push 2.0
SM Layer2 Topology Relations
SM Local Printer Push 2.0
SM Mainframe Push 2.0
SM Mobile Device Push 2.0
SM Net Printer Push 2.0
SM Network Component Push
SM Node Relations Push 2.0
SM Running Software Push 2
SM Storage Push 2.0
SM Switch Push 2.0
SM Query
Find:
Selected Items
Root
Integration
Service Manager
Push
rdbmsData
Find:
OK
Cancel

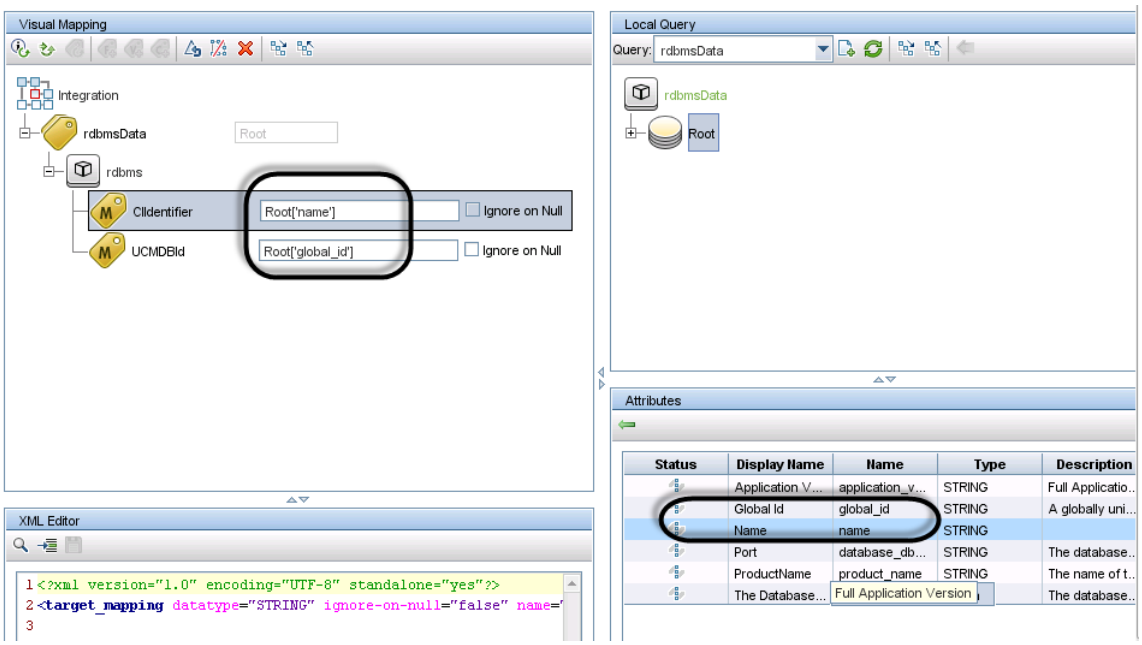
9. Click **OK**.



10. Drag and drop the node “Root” from the Local Query pane into the mapping area.
11. Drag and drop the node “RDBMS” from the External Class Model pane into the mapping area.
12. Drag and drop SM attributes you want into the mapping area.



13. Drag and drop the relevant UCMDB attributes into the mapping area.



14. Save the new mapping file.

> **Note:** When you create or edit and then save a configuration file in Adapter Management, UCMDB automatically restarts the adapter with the new configuration file.

Using Groovy scripts in XML Configuration Files

The Service Manager Enhanced Generic Adapter uses XML and Groovy mapping scripts for four field mapping scenarios: One to One, One to Many, Many to Many, and Value Conversion. Groovy scripts have to be used for complex scenarios. For more information, see "Update the Configurations for Custom CI Types in UCMDB" on page 23.

How to Add a CI Relationship Type to the Integration for Data Push

Once you have added a new CI type to the integration and have created relationships between it and other CI types in UCMDB, for each of these relationship types you need to perform the following tasks so that UCMDB can push each type of relationships to Service Manager.

As an example, the following steps illustrate how you add a relationship type named **Ownership** (between the **Cost** and **CostCategory** CI types) to the integration for data push.

Note: These steps assume that you have already added the **Cost** and **CostCategory** CI types to the integration.

1. Create a query to push the relationship type.
   See "How to Create a Query to Push a Relationship Type" below.

2. Map the relationship query to a Service Manager web service object.
   See "How to Map a Relationship Type Query to the Service Manager Web Service Object" on page 176.

3. Create an XML configuration file for the new relationship type.

   See "How to Create an XML Configuration File for a Relationship Type" on page 177.

How to Create a Query to Push a Relationship Type

Once you have created a relationship between two CI types, you must create a query to push the relationship to Service Manager.

Note: Any query you create must conform to the "Query Requirements" on page 119.

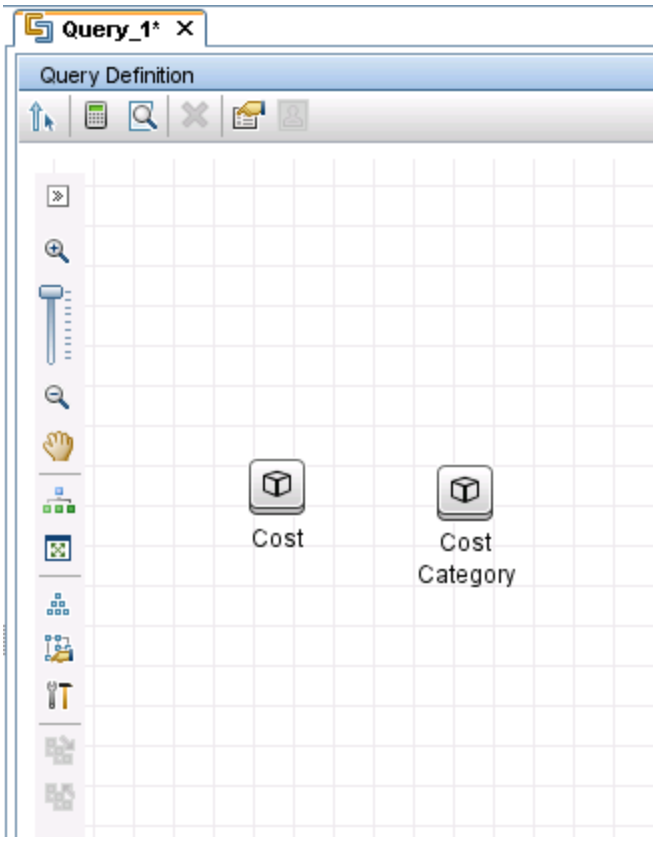
To create a new query called **Cost CostCategory Ownership Relations** for Ownership relationships between the Cost and CostCategory CI types:

1. Log in to UCMDB as an administrator.

2. Navigate to **Modeling** > **Modeling Studio**.

3. Click **New** > **Query**. The Query Definition pane is displayed.

4. From the CI Type Selector, drag the Cost and CostCategory CI types to the query pane.



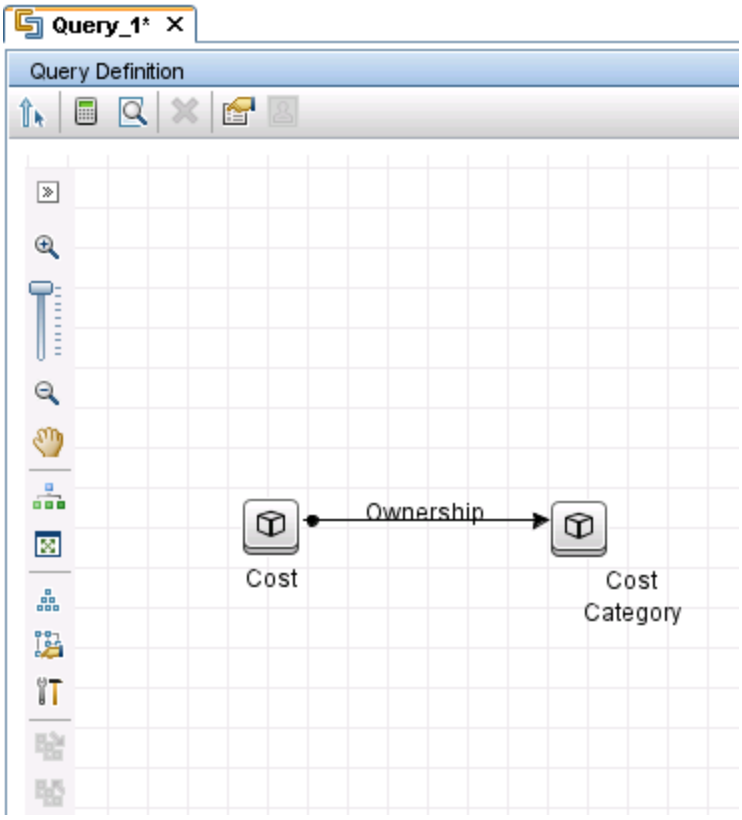
5. Create an Ownership relationship from Cost to CostCategory.
   a. Click the **Create Relationship** icon.

   b. Select the **Cost** node, and drag the arrow from it to the CostCategory node.

   c. Select **Regular Relationship**, and click **OK**.

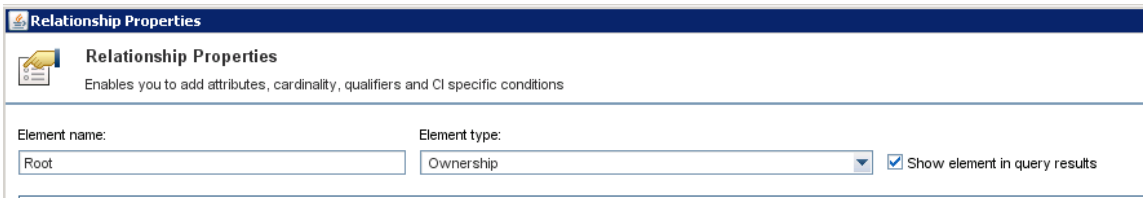
   d. Select **Connection** > **Ownership**, enter `Ownership` for Relationship Name, and click **OK**. An

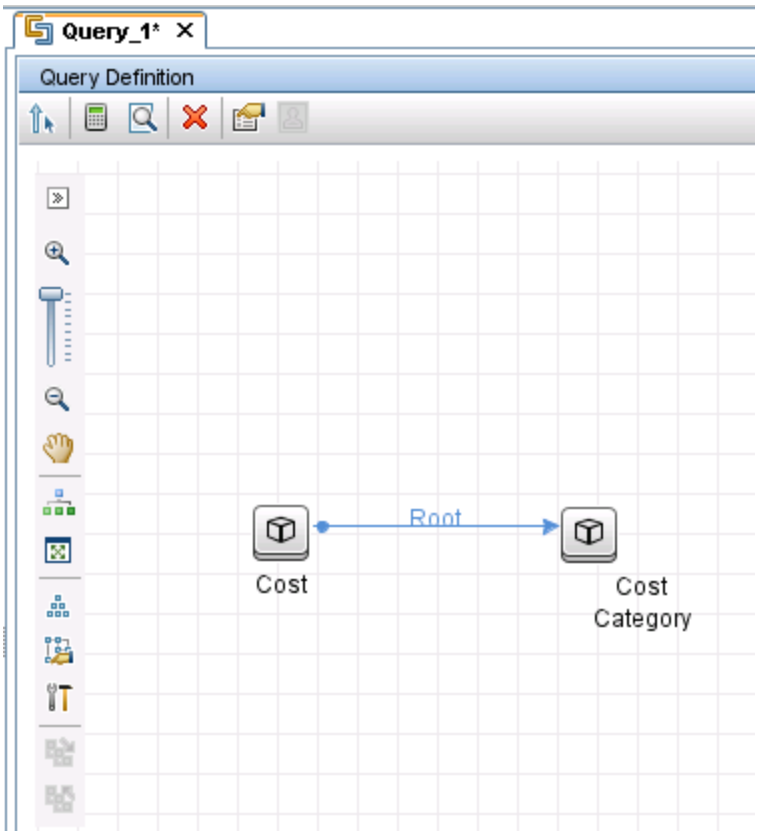
Ownership relationship is created between the CI types.



6. Right-click the relationship arrow, and select **Relationship Properties**.

7. Change the element name from **Ownership** to **Root** (or a name starting with “Root_”), and then click **OK**.





8. Click the **Save** icon, and save the query as described in the following.
   a. Enter a query name. For example, `Cost CostCategory Ownership Relations Push`.
   b. Select the **Integration** > **Service Manager** > **Push** folder.
   c. Click **OK**.

The query is now created. You are ready to add this query to the push configuration file (smPushConf.xml).

How to Map a Relationship Type Query to the Service Manager Web Service Object

Once you have created a query for a relationship type, you need to map the query to the Relationship web service object in Service Manager as described in the following steps.

1. Navigate to **Data Flow Management** > **Adapter Management** > **ServiceManagerEnhancedAdapter9-x** > **Configuration Files**.
2. Click the **smPushConf.xml** file.

3. Add a mapping entry by copying an existing one for relationship push. For example, add the following mapping entry.

```
<tql name="SM Layer2 Topology Relations Push 2.0"
    resourceCollectionName="Relationships" resourceName="Relationship" />
```

4. Change the TQL query name to the name of the query you created for the relationship type. For example, `Cost CostCategory Ownership Relations Push`.

```
<tql name="Cost CostCategory Ownership Relations Push"
    resourceCollectionName="Relationships" resourceName="Relationship" />
```

5. Click **Save** to save the configuration file.

How to Create an XML Configuration File for a Relationship Type

To create an XML configuration file for a relationship type:

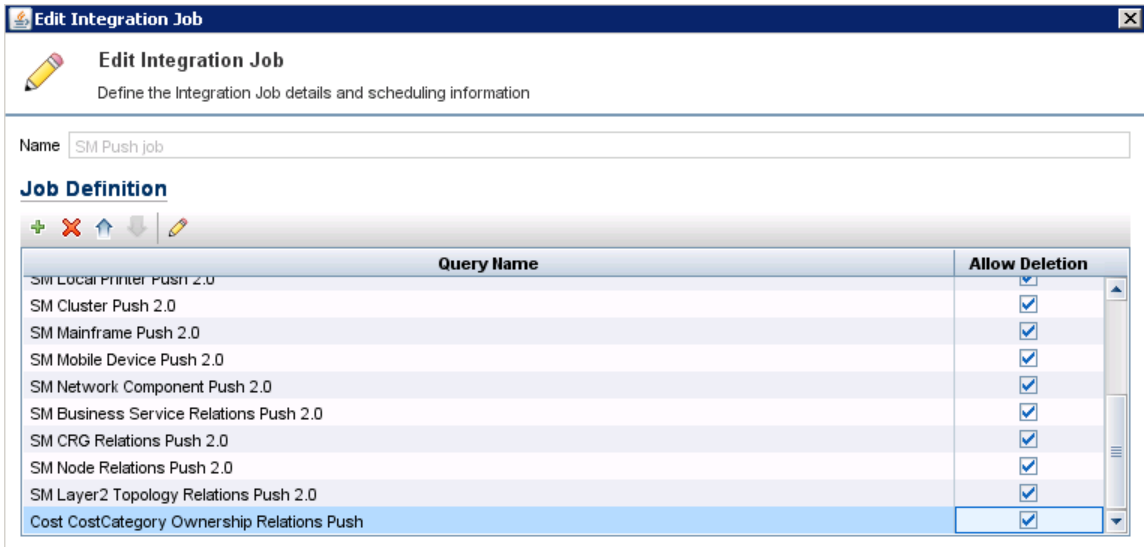
1. Log in to UCMDB with an administrator account.
2. Navigate to **Data Flow Management** > **ServiceManagerEnhancedAdapter9-x**, and select the adapter.
3. Click the **Create New Resource** icon .
4. Select **New Configuration File**.
5. Enter the full file name: `<AdapterID>/mappings/push/<filename>`. For example, `ServiceManagerEnhancedAdapter9-x/mappings/push/SM Cost CostCategory Ownership.xml`.
6. Click **OK** to save to the new file. The new file appears in the list of configuration files.
7. Open the new file in the XML Editor of the Visual Mapping interface.
8. Copy the content from an existing mapping file to overwrite the content of the new file in the XML Editor pane, and update query name to the name of the query you created (`Cost CostCategory Ownership Relations Push`).

```
<?xml version="1.0" encoding="UTF-8"?>
    <integration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="../mappings_schema.xsd">
        <info>
            <source name="UCMDB"     version="10.20" vendor="HP"/>
```

```
        <target name="SM"       version="9.40"    vendor="HP"/>
    </info>
    <import>
        <scriptFile path="mappings.scripts.SMPushFunctions"/>
    </import>
    <!--
        Push Cost to Cost Category Relations.
    -->
    <target_entities>
        <source_instance query-name="Cost CostCategory Ownership Relations
Push" root-element-name="Root">
            <target_entity name="Relationship">
                <target_mapping name="RelationshipType" datatype="STRING"
value="SMPushFunctions.getDisplayName(Root['element_type'],ClassModel)"/>
                <target_mapping name="ParentCI" datatype="STRING"
value="SMPushFunctions.getEndId(OutputCI.getExternalId().getEnd1Id())"/>
                <target_mapping name="ChildCIs" datatype="STRING_LIST"
value="[SMPushFunctions.getEndId(OutputCI.getExternalId().getEnd2Id())]"/>
            </target_entity>
        </source_instance>
    </target_entities>
</integration>
```

9. Click the **Save** icon to save the new configuration file.

Now, you have added the new relationship type to the integration. Next, you need to add the new relationship query to a data push job, as shown in the following figure. For detailed steps, see "How to Add a Custom Query to an Integration Job" on the next page.



Caution: Before you run the relationship push job, make sure you have already added the relevant

CI types (**Cost** and **CostCategory** in this example) to the integration and pushed the relevant CIs to Service Manager.

How to Add a Custom Query to an Integration Job

In order for the integration to synchronize your custom CI types, attributes, and relationships between your Service Manager and UCMDB systems, you must add your custom queries to a data push or population job.

The following steps illustrate how to add an custom query named **rdbmsData**, which you created previously (see "How to Create a Query to Synchronize the CI Type" on page 159), to a data push job. The steps for adding a population query to a population job are similar.

To add a custom query to a data push or population job:

1. Log in to UCMDB as an administrator.
2. Navigate to **Data Flow Management** > **Integration Studio**.
3. Click the name of your Service Manager integration point. For example, **sm_integration**.
4. Click the **Data Push** tab.

   Note: To add a query to a population job, click the **Population** tab instead.

5. Click the name of the integration job. For example, **SM Configuration Item Push job**.
6. Click the **Edit** icon  .
7. Click the **Add** icon. 
8. Click **Integration** > **Service Manager** > **Push** > **rdbmsData**.

   Note: For population, navigate to **Integration** > **Service Manager** > **Population**, and click the population query instead.

9. Click **OK** to add the custom query.

10. Enable the **Allow Deletion** option for the query to allow the integration job to delete removed data.

11. Click **OK** to close the Update Job Definition window.

How to Add a CI Type, Attribute or Relationship Type to the Integration for Population

All out-of-the-box mapping files for Population can be found by navigating to **Data Flow Management** > **Adapter Management** > **ServiceManagerEnhancedAdapter9-x** > **ServiceManagerEnhancedAdapter9-x/mappings/population/<*Name of Mapping File*>**.

With the Visual Mapping tool, you can add a CI Type, CI attribute, or relationship to the integration for Population by following similar steps to those for data push. Additionally, you also need to add your population queries to a population job. Refer to the following steps for data push:

"How to Add a CI Attribute to the Integration for Data Push" on page 141

"How to Add a CI Type to the Integration for Data Push" on page 155

"How to Add a CI Relationship Type to the Integration for Data Push" on page 172

"How to Add a Custom Query to an Integration Job" on the previous page

How to Enable or Disable UCMDB ID Pushback for a CI Type

When a new CI is created in UCMDB, a UCMDB ID value is assigned to the CI. When CIs are synchronized from Service Manager (SM) to UCMDB through population, the UCMDB ID Pushback feature can push their UCMDB ID values back to SM. The UCMDB ID field is then used as a flag to indicate if a CI has already been synchronized to UCMDB through population.

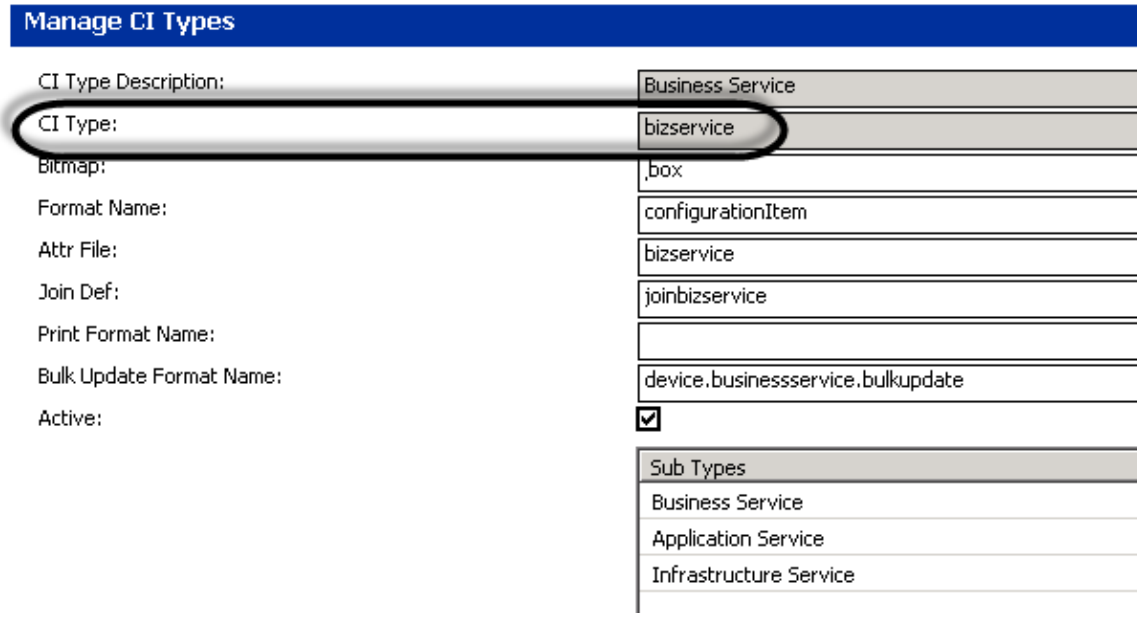
There are also cases when you do not want to enable the UCMDB ID pushback feature. For example, the scheduled_downtime CI type in UCMDB does not physically exist in SM. Instead, the integration retrieves Scheduled Downtime information from several entities in SM and then synchronizes the information to scheduled_downtime CIs in UCMDB. For this reason, the UCMDB ID Pushback feature is disabled by default for scheduled_downtime CIs; otherwise a pushback error will occur during population.

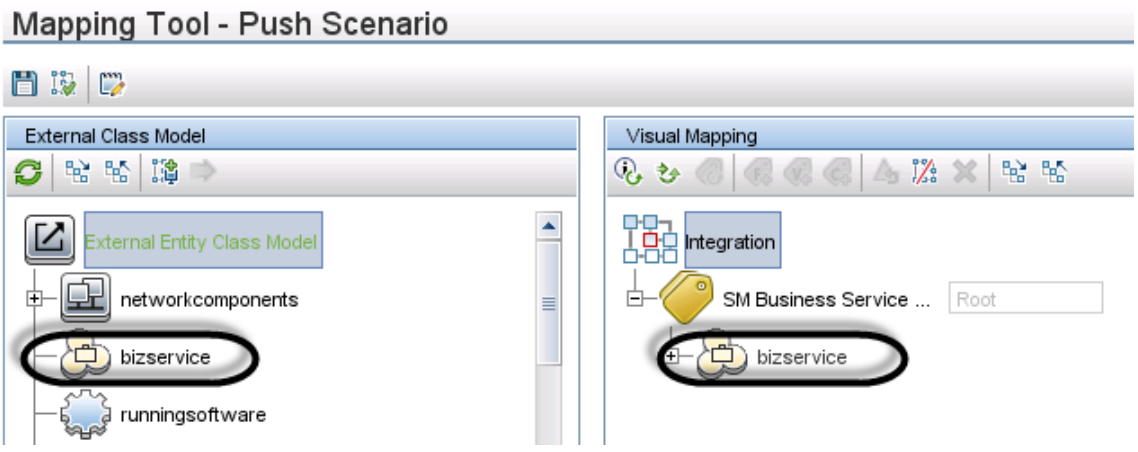
To enable UCMDB ID pushback for a CI type, follow these steps:

1. Add the following entries to your population configuration file for the CI type (for example, **My Business Service Population.xml**).

```
<target_mapping name="sm_id" datatype="STRING" value="bizservice['CIName']"/>
<target_mapping name="global_id" datatype="STRING" value="bizservice['UCMDBId']
    "/>
```

Note: In this example, **bizservice** is the CI Type display name defined in Service Manager and also the external class model name displayed in the UCMDB Visual Mapping interface, as shown in the following figures.





2. Save the XML configuration file.

3. Open the smPopConf.xml file, and make sure that the UCMDB ID Pushback setting for the CI type is either not present or is set to true:

   ```
   <pushback>
      <type name="business_service" enable="true"/>
   </pushback>
   ```

   Where: <type name> is the name of the CI type in UCMDB.

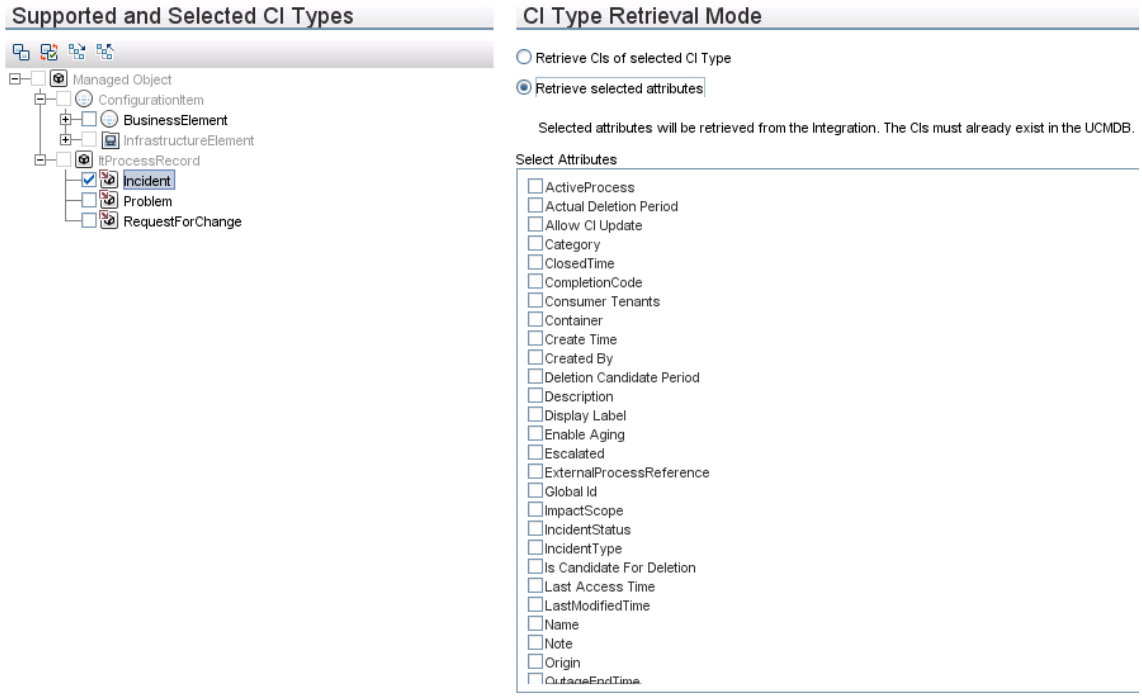
To disable UCMDB ID Pushback feature for a CI type, follow these steps:

1. Open the smPopConf.xml file.
2. In the <pushback> section, insert one line for the CI type:

   ```
   <config>
         <pushback>
             <type name="scheduled_downtime" enable="false"/>
             <type name="business_service" enable="false"/>
         </pushback>
         <mapping>
             <tql name="SM Business Service Population 2.0">
             ...
             </tql>
              …
         </mapping>
    </config>
   ```

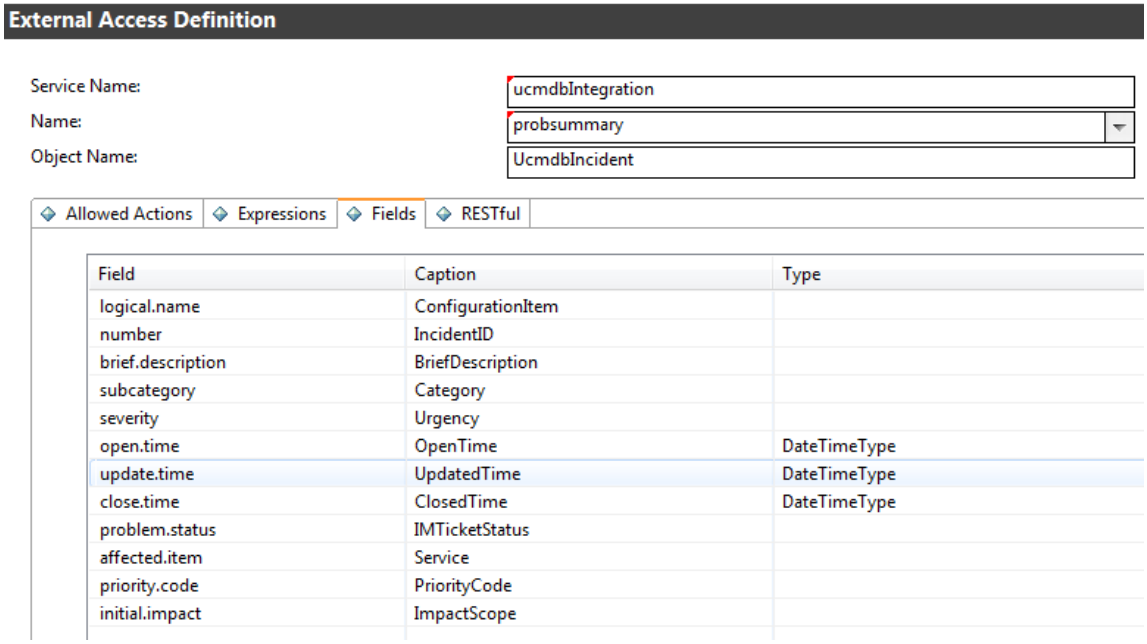
How to Add an Attribute of a Supported CI Type for Federation

Out-of-the-box, the integration supports federation for three external CI types in UCMDB: Incident, Problem, and RequestForChange. For each of the supported CI types, there is a list of attributes in UCMDB that you can map to Service Manager web service objects for federation. The following figure shows the out-of-the-box UCMDB CI attributes available for the Incident CI type.



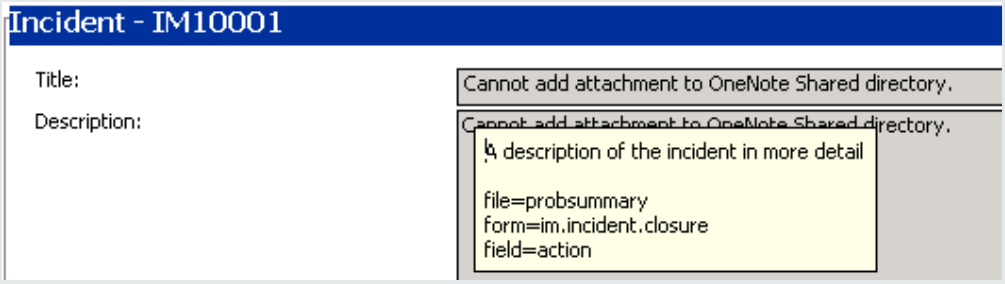
For example, to add an SM Incident attribute for federation, you need to expose the field in the SM UcmdbIncident web service object and then map it to an appropriate UCMDB attribute (if one does not already exist, you need to create it in UCMDB first).

The following figure shows the fields that are exposed in the `UcmdbIncident` web service object in Service Manager.



You can expose more fields so that more Incident attributes can be federated to UCMDB. As an example, the following describes how to add the "action" field in the Service Manager **probsummary** (Incident) file for federation, by mapping it to a new UCMDB attribute named **details**.

Note: On the Incident form in Service Manager, the "action" field is labeled "Description," which is used to describe the incident record in more detail. See the following figure.



To add an attribute of a supported CI type for federation:

1. **Add the SM attribute to its web service object.**
   The following example describes how to expose the SM "action" field of Incident in the UcmdbIncident web service object.

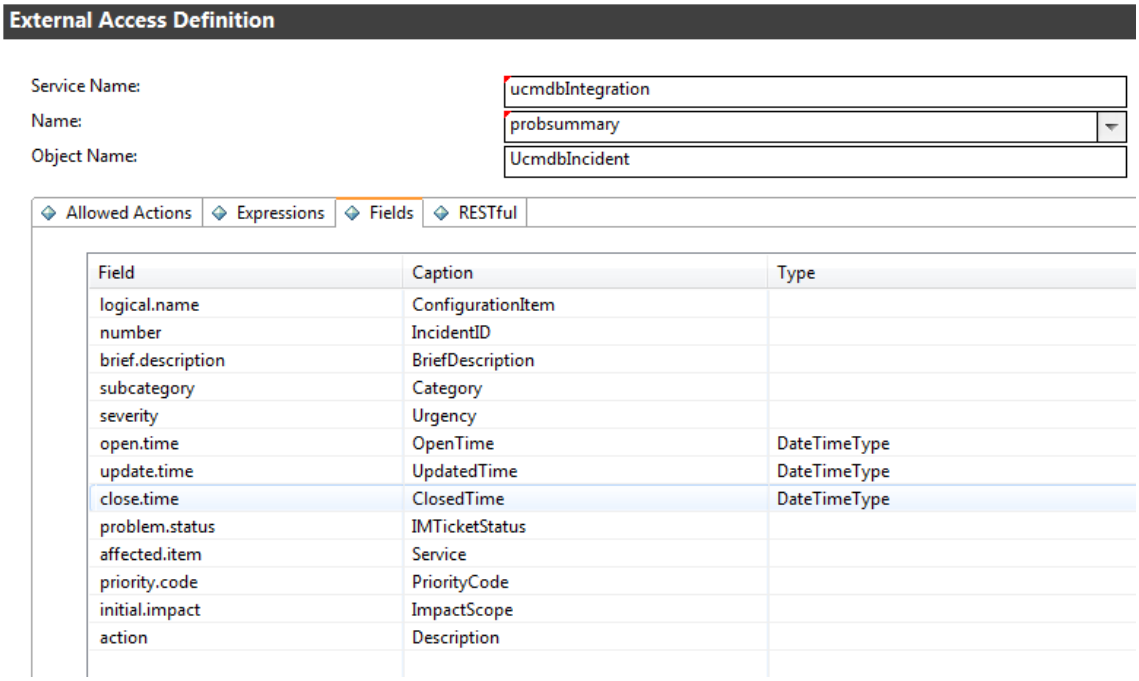
   a. Log in to Service Manager as a system administrator.

   b. Navigate to **Tailoring** > **Web Services** > **Web Service Configuration**.

   c. Enter the following field values, and then click **Search**.
      - Service Name: `ucmdbIntegration`
      - Name: `probsummary`

      The UcmdbIncident web service object is displayed.

   d. On the **Fields** tab, add the following row:
      - Field: `action`
      - Caption: `Description`



e. Save the web service object.

2. **Map the SM attribute to a UCMDB attribute.**
   The following example describes how to map the SM "action" attribute to a new UCMDB attribute named **details**.

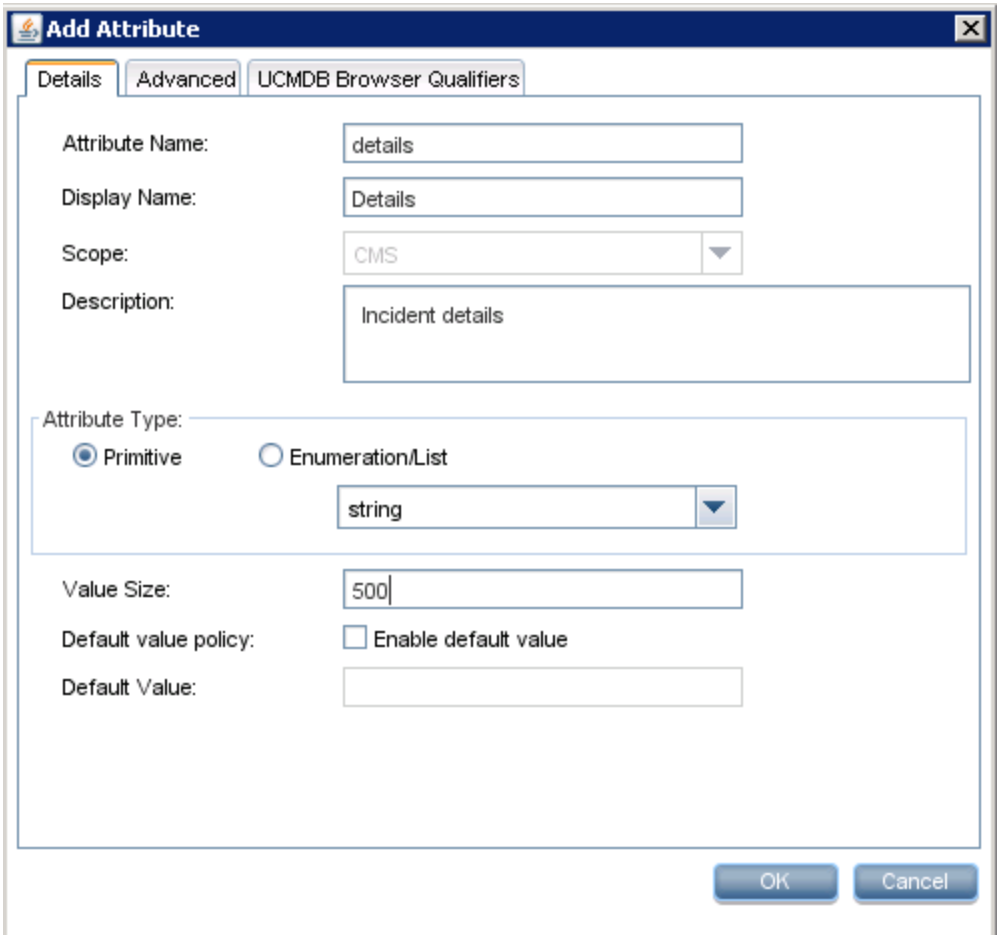
   a. Log in to UCMDB as an administrator.

   b. Navigate to **Modeling** > **CI Type Manager**.

   c. Navigate to **ItProcessRecord** > **Incident**, and open its properties pane.

   d. Click the **Add** icon to add a new attribute named **details** to the Incident CI type.

      - name: `details`
      - Display Name: `Details`
      - Description: `Incident details`
      - Attribute Type: select **Primitive** > **String**
      - Value Size: 500



e. Save the Incident CI type record.

f. Navigate to **Data Flow Management** > **Adapter Management** > **ServiceManagerEnhancedAdapter9-x** > **Configuration Files**.

g. Edit the relevant federation mapping file (**ServiceManagerEnhancedAdapter9-x/mappings/federation/SM Incident 2.0.xml**) to add a mapping entry for the new attribute.

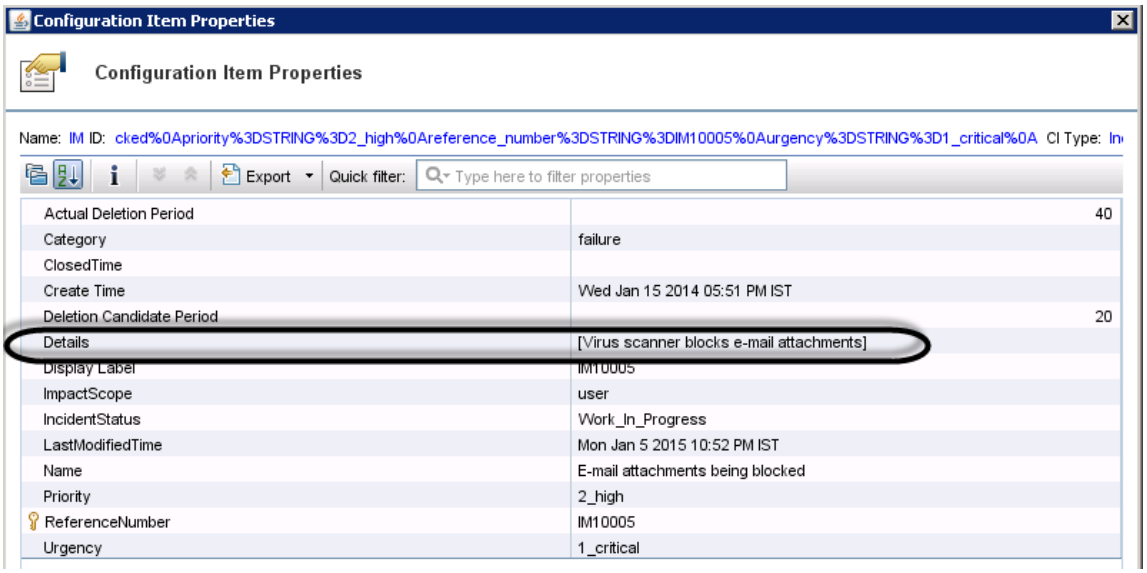
```
<target_entities>
            <source_instance query-name="SM Incident 2.0" root-element-
    name="ucmdbIncident">
                <target_entity name="incident">
                    <target_mapping name="reference_number"
    datatype="STRING"     value="ucmdbIncident['IncidentID']"/>
                    <target_mapping name="name"
    datatype="STRING"     value="ucmdbIncident['BriefDescription']"/>
                    <target_mapping name="priority"
    datatype="STRING"     value="SMFederationFunctions.getEnumValue
    (ucmdbIncident['PriorityCode'],'Priority')"/>
```

```
                <target_mapping name="incident_status"
datatype="STRING"
value="SMFederationFunctions.firstLetterToLowerAndReplaceSpaceWithUnders
core(ucmdbIncident['IMTicketStatus'])"/>
                <target_mapping name="category"
datatype="STRING"
value="SMFederationFunctions.replaceSpaceWithUnderscore(ucmdbIncident
['Category'])"/>
                <target_mapping name="closed_time"
datatype="DATE"     value="SMFederationFunctions.convertDate
(ucmdbIncident['ClosedTime'])"/>
                <target_mapping name="create_time"
datatype="DATE"     value="SMFederationFunctions.convertDate
(ucmdbIncident['OpenTime'])"/>
                <target_mapping name="last_modified_time"
datatype="DATE"     value="SMFederationFunctions.convertDate
(ucmdbIncident['UpdatedTime'])"/>
                <target_mapping name="impact_scope"
datatype="STRING"     value="SMFederationFunctions.getEnumValue
(ucmdbIncident['ImpactScope'],'ImpactScope')"/>
                <target_mapping name="urgency"
datatype="STRING"     value="SMFederationFunctions.getEnumValue
(ucmdbIncident['Urgency'],'Urgency')"/>
                <target_mapping name="details" datatype="STRING"
value="ucmdbIncident['Description']"/>
            </target_entity>
        </source_instance>
    </target_entities>
```

h. Click **Save** to save the file.

Now the Description (field name: action) attribute of SM Incident has been added to the integration for federation. You can run an Incident federation query in the UCMDB Modeling Studio to see if the SM Description data is properly federated. For information about how to run a federation query, see "Examples of Using Federation" on page 50.

The following figure shows an example where the Description of an SM incident record has been federated to UCMDB as **Details**.


Configuration Item Properties
Configuration Item Properties
Name: IM ID: cked%0Apriority%3DSTRING%3D2_high%0Areference_number%3DSTRING%3DIM10005%0Aurgency%3DSTRING%3D1_critical%0A CI Type: In
Export
Quick filter:
Type here to filter properties
Actual Deletion Period 40
Category failure
ClosedTime
Create Time Wed Jan 15 2014 05:51 PM IST
Deletion Candidate Period 20
Details [Virus scanner blocks e-mail attachments]
Display Label IM10005
ImpactScope user
IncidentStatus Work_In_Progress
LastModifiedTime Mon Jan 5 2015 10:52 PM IST
Name E-mail attachments being blocked
Priority 2_high
ReferenceNumber IM10005
Urgency 1_critical

Chapter 6: Troubleshooting

When data push and population errors occur, you can check the error messages and the integration log files to identify the root causes and fix the errors. This chapter describes the general troubleshooting steps, as well as typical errors and solutions.

This section includes:

Troubleshooting Data Push Issues

When data push errors or problems occur, you can check the error messages and the log file to figure out the root causes and then fix the errors.

This integration uses the following error codes for data push.

SM Error Code	Description	UCMDB Error Code
-1	Server Application Error from SM	900008
-4	401 Authorization error from SM	900001
3	Data Locked Error from SM	900003
9	Restful configuration not found	900009
51	Changed By Another Process Error from SM	900004
70	Invalid Action Error from SM	900005
71	Validation Check Failed error from SM	900002
881	Invalid Data Error from SM	900007
882	Cannot find CI in SM for relationship operation	900010
	Failed to create message in SM	900013
	Communication with SM Failed	900014
	Other Error from SM	900099

When a data push job has failed, the job status becomes **Failed**. Troubleshoot the failed job as follows:

- Check the error messages of the failed job in the Universal CMDB studio.
  See "How to Check the Error Message of a Failed Push Job" below.
- Check the log file for more details.
  See "How to Check the Push Log File" on page 192.

When a data push job was completed with failures of partial records, the job status becomes **Completed**. Troubleshoot the failed records as follows:

- Check the error messages of failed CIs in the Universal CMDB studio.
  See "How to Check the Error Messages of Failed CIs or Relationships in a Push Job" on page 192.
- Check the log file for more details.
  See "How to Check the Push Log File" on page 192.

How to Check the Error Message of a Failed Push Job

To check the error message of a failed job:

1. Log in to UCMDB as an administrator.
2. Navigate to **Data Flow Management** > **Integration Studio**.
3. Select the integration point for this integration from Integration Point list.
4. Click the **Data Push** tab.
5. Select the job from Integration Jobs.
6. Click the **Job Errors** sub-tab, and double-click the Severity of a message from the list.
   A popup window displays the detailed error message of this failed job. The following is an excerpt of a sample error message indicating that no mapping configuration was found for the push TQL query.

```
java.lang.RuntimeException: No mapping is found for TQL: "SM Business Service
    Push 2.0", or Cannot retrieve the mapping from SM side by QueryNodeName
    [bizservice", please configure in smPushConf.xml or SM configuration
           at
    com.mercury.topaz.fcmdb.adapters.serviceDeskAdapter.push.SmGenericPusher.pus
    h(SmGenericPusher.java:119)
           ... 35 more
    --- End of probe-side exception ---
```

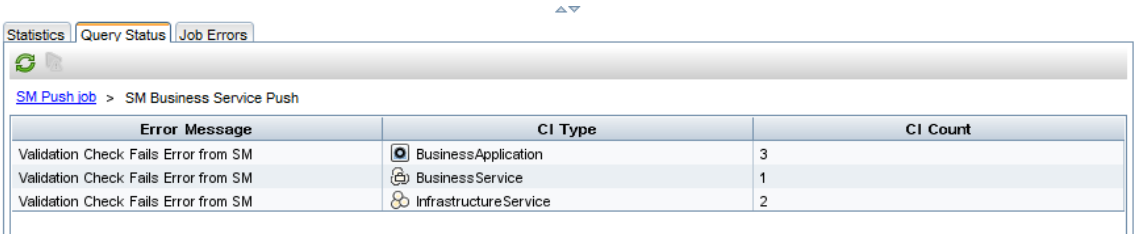
```
        at
com.hp.ucmdb.discovery.probe.agents.probemgr.adhoctasks.AdHocProbeRequestOpe
ration.convertThrowableToStringSafeException
(AdHocProbeRequestOperation.java:86)
        at
com.hp.ucmdb.discovery.probe.agents.probemgr.adhoctasks.AdHocProbeRequestOpe
ration.performAction(AdHocProbeRequestOperation.java:77)
        at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskdispatcher.AdHocTaskDispatc
her.dispatchTask(AdHocTaskDispatcher.java:70)
        at sun.reflect.GeneratedMethodAccessor59.invoke(Unknown Source)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke
(DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:601)
        at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2
(StandardMBeanIntrospector.java:111)
        at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2
(StandardMBeanIntrospector.java:45)
        at com.sun.jmx.mbeanserver.MBeanIntrospector.invokeM
(MBeanIntrospector.java:235)
        at com.sun.jmx.mbeanserver.PerInterface.invoke(PerInterface.java:138)
        at com.sun.jmx.mbeanserver.MBeanSupport.invoke(MBeanSupport.java:252)
        at javax.management.StandardMBean.invoke(StandardMBean.java:405)
        at com.sun.jmx.interceptor.DefaultMBeanServerInterceptor.invoke
(DefaultMBeanServerInterceptor.java:819)
        at com.sun.jmx.mbeanserver.JmxMBeanServer.invoke(JmxMBeanServer.java:792)
        at javax.management.MBeanServerInvocationHandler.invoke
(MBeanServerInvocationHandler.java:305)
        at org.springframework.jmx.access.MBeanClientInterceptor.doInvoke
(MBeanClientInterceptor.java:405)
        at org.springframework.jmx.access.MBeanClientInterceptor.invoke
(MBeanClientInterceptor.java:353)
        at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
(ReflectiveMethodInvocation.java:172)
        at org.springframework.aop.framework.JdkDynamicAopProxy.invoke
(JdkDynamicAopProxy.java:202)
        at com.sun.proxy.$Proxy57.dispatchTask(Unknown Source)
        at
com.hp.ucmdb.discovery.probe.agents.probegw.managementtasks.adhoctasks.Adhoc
Thread.run(AdhocThread.java:54)
        ... 3 more
```

How to Check the Error Messages of Failed CIs or Relationships in a Push Job

When a data push job is completed with failures of partial records, in the Universal CMDB studio, you can check the detail error message for each failed record.

To check the error messages of failed records in a data push job:

1. Log in to UCMDB as an administrator.
2. Navigate to **Data Flow Management** > **Integration Studio**.
3. Select the integration point for this integration from Integration Point list.
4. Click the **Data Push** tab.
5. Select the job from Integration Jobs.
6. Click the **Query Status** sub-tab.
7. Double-click a query with failures. The Error Message and CI Count for each failed CI Type are displayed.



8. Double-click an error message. A list of failed records is displayed.
9. Double-click a failed record.
   The detailed error message of the record is displayed.

How to Check the Push Log File

You need to set the Development adapter log level to DEBUG or TRACE, so that you can check the push result tree nodes of UCMDB, and then send messages to and receive messages from Service Manager. Alternatively, you can set the log level to TRACE so as to check the conversion period based on a mapping file.

Note: To troubleshoot push, population, and federation issues, you need to set the Development adapter log level to DEBUG or TRACE in the **fcmdb.properties** and **fcmdb.push.properties** files, which are located in the \conf\log folder of your Data Flow Probe or Integration Service installation. Additionally, you can view push, population and federation log information in the **fcmdb.push.all.log** file, which is located in the \runtime\log folder of your Data Flow Probe or Integration Service installation. Your Data Flow Probe or Integration Service may be installed on the same host as your UCMDB Server or on a separate host.

To set the Development adapter log level to DEBUG or TRACE:

1. Log in as an administrator to the host on which your UCMDB Data Flow Probe or Integration Service is installed.

2. Navigate to the <*UCMDB installation folder*>\DataFlowProbe\conf\log folder or the <*UCMDB installation folder*\UCMDBServer\integrations\conf\log. For example:

   C:\hp\UCMDB\DataFlowProbe\conf\log\

   Or C:\hp\UCMDB\UCMDBServer\integrations\conf\log

3. Open the **fcmdb.properties** file in a text editor, and change the log level to `DEBUG` or `TRACE`. For example:

   ```
   #loglevel can be any of TRACE DEBUG INFO WARN ERROR FATAL
       loglevel=DEBUG
       def.file.max.size=5000KB
       def.files.backup.count=10
       msg.layout=%d  %-5p  [%t] - %m%n
   ```

4. Open the **fcmdb.push.properties** files in a text editor, and change the log4j log level to `DEBUG` or `TRACE`. For example:

   ```
   ### UCMDB log4j Properties
       log.file.path=log/${log.folder.path.output}
       #loglevel can be any of TRACE DEBUG INFO WARN ERROR FATAL
       loglevel=DEBUG
       def.file.max.size=5000KB
       def.files.backup.count=10
       msg.layout=%d  %-5p  [%t] - %m%n
   ```

5. Save the files.

To check the push log file:

1. Log in as an administrator to the host on which your UCMDB Data Flow Probe or Integration Service is installed.
2. Navigate to the <*UCMDB installation folder*>\DataFlowProbe\runtime\log folder or the <*UCMDB installation folder*\UCMDBServer\integrations\runtime\log folder.
3. Open the **fcmdb.push.all.log** file in a text editor.

Typical Push Errors and Solutions

This section describes typical error messages that may occur during data push, as well as their solutions.

This section includes:

Query not Configured in smPushConf.xml

Sample Configuration

The TQL queries used for pushing relationships must be configured in the smPushConf.xml file.

```
<config>
          <mapping>
                  <!--
                     Wildcard is supported for TQL names while you configure the mapping now,
                     you can add the '*' at the end of the TQL name while configuring the
   mapping.
                     We use the wildcard for mapping in OOTB, so each time you do save as to a
   TQL,
                     you could still push it to SM without manual changing the mapping.
                     For e.g., if you have saved the <TQL_name> query to <TQL_name>_1, and <TQL_
   name>_2,
                     the TQL name is specified as <TQL_name>* in this configuration file, and
   the
                     integration will automatically use this mapping entry on all of the three
   TQLs.
                     However, using the exact TQL name to configure the mapping is still
   supported.
                  -->
           <tql name="SM Business Service Relations Push 2.0"
   resourceCollectionName="Relationships" resourceName="Relationship"/>
```

```
        <tql name="SM CRG Relations Push 2.0"
resourceCollectionName="Relationships" resourceName="Relationship"/>
        <tql name="SM Node Relations Push 2.0"
resourceCollectionName="Relationships" resourceName="Relationship"/>
        <tql name="SM Layer2 Topology Relations Push 2.0"
resourceCollectionName="Relationships" resourceName="Relationship"/>
      </mapping>
</config>
```

Error Message

The push job fails with a "Failed" status. From both the log file and the detail error message of the failed job in the Universal CMDB studio (see "How to Check the Error Message of a Failed Push Job" on page 190), you receive an error like the following:

```
java.lang.RuntimeException: No mapping is found for TQL: "SM Business Service Push
    2.0", or Cannot retrieve the mapping from SM side by QueryNodeName [bizservice",
    please configure in smPushConf.xml or SM configuration
        at
    com.mercury.topaz.fcmdb.adapters.serviceDeskAdapter.push.SmGenericPusher.push
    (SmGenericPusher.java:119)
        ... 35 more
    --- End of probe-side exception ---

        at
    com.hp.ucmdb.discovery.probe.agents.probemgr.adhoctasks.AdHocProbeRequestOperati
    on.convertThrowableToStringSafeException(AdHocProbeRequestOperation.java:86)
        at
    com.hp.ucmdb.discovery.probe.agents.probemgr.adhoctasks.AdHocProbeRequestOperati
    on.performAction(AdHocProbeRequestOperation.java:77)
        at
    com.hp.ucmdb.discovery.probe.agents.probemgr.taskdispatcher.AdHocTaskDispatcher.
    dispatchTask(AdHocTaskDispatcher.java:70)
        at sun.reflect.GeneratedMethodAccessor59.invoke(Unknown Source)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke
    (DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:601)
        at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2
    (StandardMBeanIntrospector.java:111)
        at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2
    (StandardMBeanIntrospector.java:45)
        at com.sun.jmx.mbeanserver.MBeanIntrospector.invokeM
    (MBeanIntrospector.java:235)
        at com.sun.jmx.mbeanserver.PerInterface.invoke(PerInterface.java:138)
        at com.sun.jmx.mbeanserver.MBeanSupport.invoke(MBeanSupport.java:252)
        at javax.management.StandardMBean.invoke(StandardMBean.java:405)
        at com.sun.jmx.interceptor.DefaultMBeanServerInterceptor.invoke
    (DefaultMBeanServerInterceptor.java:819)
        at com.sun.jmx.mbeanserver.JmxMBeanServer.invoke(JmxMBeanServer.java:792)
```

```
        at javax.management.MBeanServerInvocationHandler.invoke
    (MBeanServerInvocationHandler.java:305)
        at org.springframework.jmx.access.MBeanClientInterceptor.doInvoke
    (MBeanClientInterceptor.java:405)
        at org.springframework.jmx.access.MBeanClientInterceptor.invoke
    (MBeanClientInterceptor.java:353)
        at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
    (ReflectiveMethodInvocation.java:172)
        at org.springframework.aop.framework.JdkDynamicAopProxy.invoke
    (JdkDynamicAopProxy.java:202)
        at com.sun.proxy.$Proxy57.dispatchTask(Unknown Source)
        at
    com.hp.ucmdb.discovery.probe.agents.probegw.managementtasks.adhoctasks.AdhocThre
    ad.run(AdhocThread.java:54)
        ... 3 more
```

Solution

Search for text string `No mapping is found for TQL` to find the name of the query that is not yet configured, and then add a mapping entry for the query in the `smPushConf.xml` file.

Mapping File not Well Formed

Sample Configuration

The "target_entity name" should be **bizservice** (which is the CI type display name defined in Service Manager, and also the external class model name displayed in the UCMDB Visual Mapping tool interface); however you have configured a wrong name **businessservice**.

```
<?xml version="1.0" encoding="UTF-8"?>
    <integration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="../mappings_schema.xsd">
        <info>
            <source name="UCMDB"   version="10.20" vendor="HP"/>
            <target name="SM"      version="9.40"  vendor="HP"/>
        </info>
        <import>
            <scriptFile path="mappings.scripts.SMPushFunctions"/>
        </import>
        <!--
           Push uCMDB CIT to SM Business Service.
        -->
        <target_entities>
            <source_instance query-name="SM Business Service Push 2.0" root-element-
    name="Root" >
                <target_entity name="businessservice">
                    <target_mapping name="UCMDBId" datatype="STRING" value="Root
```

```
['global_id']"/>
                <target_mapping name="CustomerId" datatype="STRING"
value="SMPushFunctions.getCustomerId(CustomerInformation)"/>
                <target_mapping name="Type" datatype="STRING"
value="'bizservice'"/>
                <target_mapping name="Subtype" datatype="STRING"
value="SMPushFunctions.getSMSubType(Root['element_
type'],ClassModel,'BizService')"/>
                <target_mapping name="ServiceProvider" datatype="STRING"
value="Root['provider']"/>
                <target_mapping name="ServiceName" datatype="STRING" value="Root
['display_label']"/>
                <target_mapping name="CIIdentifier" datatype="STRING"
value="Root['display_label']"/>
            </target_entity>
        </source_instance>
    </target_entities>
</integration>
```

Error Message

The data push job fails with a “Failed” status. From both the log file and the detailed error message of the failed job in the Universal CMDB studio (see "How to Check the Error Message of a Failed Push Job" on page 190), you receive an error like the following:

```
java.lang.RuntimeException: No mapping is found for TQL: "SM Business Service Push
    2.0", or Cannot retrieve the mapping from SM side by QueryNodeName
    [businessservice", please configure in smPushConf.xml or SM configuration
        at
    com.mercury.topaz.fcmdb.adapters.serviceDeskAdapter.push.SmGenericPusher.push
    (SmGenericPusher.java:119)
        ... 35 more
    --- End of probe-side exception ---

        at
    com.hp.ucmdb.discovery.probe.agents.probemgr.adhoctasks.AdHocProbeRequestOperati
    on.convertThrowableToStringSafeException(AdHocProbeRequestOperation.java:86)
        at
    com.hp.ucmdb.discovery.probe.agents.probemgr.adhoctasks.AdHocProbeRequestOperati
    on.performAction(AdHocProbeRequestOperation.java:77)
        at
    com.hp.ucmdb.discovery.probe.agents.probemgr.taskdispatcher.AdHocTaskDispatcher.
    dispatchTask(AdHocTaskDispatcher.java:70)
        at sun.reflect.GeneratedMethodAccessor59.invoke(Unknown Source)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke
    (DelegatingMethodAccessorImpl.java:43)
        at java.lang.reflect.Method.invoke(Method.java:601)
        at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2
    (StandardMBeanIntrospector.java:111)
```

```
        at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2
(StandardMBeanIntrospector.java:45)
        at com.sun.jmx.mbeanserver.MBeanIntrospector.invokeM
(MBeanIntrospector.java:235)
        at com.sun.jmx.mbeanserver.PerInterface.invoke(PerInterface.java:138)
        at com.sun.jmx.mbeanserver.MBeanSupport.invoke(MBeanSupport.java:252)
        at javax.management.StandardMBean.invoke(StandardMBean.java:405)
        at com.sun.jmx.interceptor.DefaultMBeanServerInterceptor.invoke
(DefaultMBeanServerInterceptor.java:819)
        at com.sun.jmx.mbeanserver.JmxMBeanServer.invoke(JmxMBeanServer.java:792)
        at javax.management.MBeanServerInvocationHandler.invoke
(MBeanServerInvocationHandler.java:305)
        at org.springframework.jmx.access.MBeanClientInterceptor.doInvoke
(MBeanClientInterceptor.java:405)
        at org.springframework.jmx.access.MBeanClientInterceptor.invoke
(MBeanClientInterceptor.java:353)
        at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
(ReflectiveMethodInvocation.java:172)
        at org.springframework.aop.framework.JdkDynamicAopProxy.invoke
(JdkDynamicAopProxy.java:202)
        at com.sun.proxy.$Proxy57.dispatchTask(Unknown Source)
        at
com.hp.ucmdb.discovery.probe.agents.probegw.managementtasks.adhoctasks.AdhocThre
ad.run(AdhocThread.java:54)
        ... 3 more
```

Solution

Search for the relevant CI type in Service Manager to find the correct display name, and then modify the mapping file with the correct name.

Tip: You can easily fix such kind of validation issues by using the UCMDB Visual Mapping tool to generate the mapping file.

Troubleshooting Population Issues

When population errors or problems occur, you can check the error messages and the population log file to identify the root causes and then solve the problems.

When a population job has failed, the job status becomes **Failed**. Troubleshoot the failed job as follows:

- Check the error message of the failed job in the Universal CMDB studio.
  See "How to Check the Error Message of a Failed Population Job" below and "Typical Population Error Messages and Solutions" on the next page.
- Check the log file for more details.
  See "How to Check the Population Log File" below.

How to Check the Error Message of a Failed Population Job

While a population job fails, you can check the detailed error messages in the Universal CMDB studio.

To check the error message of a failed population job:

1. Log in to UCMDB as an administrator.
2. Navigate to **Data Flow Management** > **Integration Studio**.
3. Select the integration point for this integration.
4. Click the **Population** tab.
5. Select the failed job from Integration Jobs, and click the **Job Errors** subtab.
6. Double-click an error message from the list.
   A pop-up window opens to display the error details.

How to Check the Population Log File

You can set the Development adapter log level to DEBUG so that you can check the population result tree nodes of UCMDB, and send messages to and receive messages from Service Manager. Alternatively, you can set the log level to TRACE so as to check the conversion period based on a mapping file.

Note: To troubleshoot push, population, and federation issues, you need to set the Development adapter log level to DEBUG or TRACE in the **fcmdb.properties** and **fcmdb.push.properties** files, which are located in the \conf\log folder of your Data Flow Probe or Integration Service installation. Additionally, you can view push, population and federation log information in the **fcmdb.push.all.log** file, which is located in the \runtime\log folder of your Data Flow Probe or Integration Service installation. Your Data Flow Probe or Integration Service may be installed on the same host as your UCMDB Server or on a separate host.

To set the Development adapter log level and view the population log file, follow the steps described in "How to Check the Push Log File" on page 192.

Typical Population Error Messages and Solutions

The following describes typical error messages that may occur during population, and their solutions.

This section includes:

No TQL Query Configured in smPopConf.xml

Error Message

If you have not yet added a TQL query to your job, you cannot select this query from the list while you create or update your job.

If you have already added this TQL query to your job, but forgot to add the configuration for this query in smPopConf.xml, you will get a "Failed" status while you run this population job. In addition, in the Universal CMDB studio, you will get an error message like the following (see "How to Check the Error Message of a Failed Population Job" on the previous page).

```
running population. Destination ID: sm, Failed during query: SM Business
    Application Population 2.0, all queries: [SM Business Application Population
    2.0, SM Business Service Population 2.0],  finished queries: [].
    ERROR: com.mercury.topaz.cmdb.shared.base.CmdbException: [ErrorCode [802]
    General Integration Error{sm}]
    appilog.framework.shared.manage.impl.MamResponseException: [ErrorCode [802]
    General Integration Error{sm}]
    CMDB Operation Internal Error: class
    com.mercury.topaz.cmdb.shared.fcmdb.dataAccess.exception.AdapterAccessGeneralExc
    eption : Unsupported Query [SM Business Application Population 2.0], only
    population TQL is supported : operation Data Access Adapter Query: Retrieve
    Changed Data
           at
    com.mercury.topaz.cmdb.shared.manage.operation.impl.AbstractCommonOperation.exec
    ute(AbstractCommonOperation.java:160)
           at
    com.hp.ucmdb.dataAccess.manager.DataAccessAdapterManagerProbeImpl.executeOperati
    on(DataAccessAdapterManagerProbeImpl.java:50)
           at
```

```
com.hp.ucmdb.discovery.probe.agents.probemgr.adapters.DataAccessAdaptersFacade.i
nvokeOperation(DataAccessAdaptersFacade.java:406)
       at
com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runChangesOnPo
pulateChangesAdapter(AdapterService.java:1262)
       at
com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runQueriesOnPo
pulateChangesAdapter(AdapterService.java:1102)
       at com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runQueries
(AdapterService.java:354)
       at
com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runDiscovery
(AdapterService.java:198)
       at com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.discover
(AdapterService.java:149)
       at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecuter.JobExecuter.launchTask
(JobExecuter.java:1188)
       at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecuter.JobExecuter$JobExecute
rWorker.launch(JobExecuter.java:945)
```

In the fcmdb.push.all.log file, you can find more details like the following:

```
2014-11-20 10:40:50,949 [JobExecuterWorker-0:DS_sm_SM BS pop] ERROR - sm >> Fail to
   Create or Return PopulationConnectorOutput
   com.hp.ucmdb.federationspi.exception.DataAccessGeneralException: Unsupported
   Query [SM Business Application Population 2.0], only population TQL is supported
       at com.hp.ucmdb.adapter.smpush.ServiceManagerGenericAdapter.populate
   (ServiceManagerGenericAdapter.java:337)
       at com.hp.ucmdb.adapters.GenericAdapter.getChanges(GenericAdapter.java:881)
       at
   com.hp.ucmdb.dataAccess.operations.DataAccessAdapterQueryRetrieveChanges.getChan
   gesResult(DataAccessAdapterQueryRetrieveChanges.java:50)
       at
   com.hp.ucmdb.dataAccess.operations.DataAccessAdapterQueryRetrieveChanges.doDataA
   ccessQueryExecute(DataAccessAdapterQueryRetrieveChanges.java:38)
       at
   com.hp.ucmdb.dataAccess.operations.AbstractDataAccessLifeCycleAdapterQuery.doLif
   eCycleExecute(AbstractDataAccessLifeCycleAdapterQuery.java:34)
       at
   com.hp.ucmdb.dataAccess.operations.AbstractDataAccessLifeCycleAdapterOperation.d
   oDataAccessExecute(AbstractDataAccessLifeCycleAdapterOperation.java:57)
       at
   com.hp.ucmdb.dataAccess.operations.AbstractDataAccessAdapterOperation.dataAccess
   Execute(AbstractDataAccessAdapterOperation.java:59)
       at
   com.hp.ucmdb.dataAccess.operations.AbstractDataAccessAdapterOperation.doExecute
   (AbstractDataAccessAdapterOperation.java:37)
       at
```

```
com.mercury.topaz.cmdb.shared.manage.operation.impl.AbstractFrameworkOperation.c
ommonExecute(AbstractFrameworkOperation.java:17)
	at
com.mercury.topaz.cmdb.shared.manage.operation.impl.AbstractCommonOperation$Oper
ationExecuteFlowTrackingCommand.execute(AbstractCommonOperation.java:87)
	at
com.mercury.topaz.cmdb.shared.manage.operation.impl.AbstractCommonOperation$Oper
ationExecuteFlowTrackingCommand.execute(AbstractCommonOperation.java:60)
	at com.mercury.topaz.cmdb.shared.manage.flowmanagement.api.FlowManager.execute
(FlowManager.java:227)
	at
com.mercury.topaz.cmdb.shared.manage.operation.flow.OperationInFlowDefaultExecut
or.execute(OperationInFlowDefaultExecutor.java:23)
	at
com.mercury.topaz.cmdb.shared.manage.operation.impl.AbstractCommonOperation.exec
ute(AbstractCommonOperation.java:158)
	at
com.hp.ucmdb.dataAccess.manager.DataAccessAdapterManagerProbeImpl.executeOperati
on(DataAccessAdapterManagerProbeImpl.java:50)
	at
com.hp.ucmdb.discovery.probe.agents.probemgr.adapters.DataAccessAdaptersFacade.i
nvokeOperation(DataAccessAdaptersFacade.java:406)
	at
com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runChangesOnPo
pulateChangesAdapter(AdapterService.java:1262)
	at
com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runQueriesOnPo
pulateChangesAdapter(AdapterService.java:1102)
	at com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runQueries
(AdapterService.java:354)
	at
com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runDiscovery
(AdapterService.java:198)
	at com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.discover
(AdapterService.java:149)
	at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecuter.JobExecuter.launchTask
(JobExecuter.java:1188)
	at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecuter.JobExecuter$JobExecute
rWorker.launch(JobExecuter.java:945)
	at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecuter.JobExecuter$JobExecute
rWorker.executeTask(JobExecuter.java:867)
	at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecuter.JobExecuter$JobExecute
rWorker.run(JobExecuter.java:728)
```

Solution

Search for text string “Unsupported Query by this adapter” to find out the TQL query name that has not yet been configured, and then configure it in the smPopConf.xml file.

Nonexistent Mapping File Name Defined for a TQL Query in smPopConf.xml

Error Message

You will get a “Failed” status while you run the population job. In addition, from the Universal CMDB studio, you will get an error message like the following:

```
running population. Destination ID: sm, Failed during query: SM Business Service
    Population 2.0, all queries: [SM Business Application Population 2.0, SM
    Business Service Population 2.0],  finished queries: [SM Business Application
    Population 2.0].
    ERROR: com.mercury.topaz.cmdb.shared.base.CmdbException: [ErrorCode [802]
    General Integration Error{sm}]
    appilog.framework.shared.manage.impl.MamResponseException: [ErrorCode [802]
    General Integration Error{sm}]
    CMDB Operation Internal Error: class
    com.mercury.topaz.cmdb.shared.fcmdb.dataAccess.exception.AdapterAccessGeneralExc
    eption : Query Definition [SM Business Service Population 2.0] has no matching
    mapping. Make sure the mapping exist exists under folder 'mapping' and contains
    the exact query name and root name. Available mappings:
    [QueryRoot{queryName='SM Business Application Population 2.0',
    rootName='bizservice'}] : operation Data Access Adapter Query: Retrieve Changed
    Data
           at
    com.mercury.topaz.cmdb.shared.manage.operation.impl.AbstractCommonOperation.exec
    ute(AbstractCommonOperation.java:160)
           at
    com.hp.ucmdb.dataAccess.manager.DataAccessAdapterManagerProbeImpl.executeOperati
    on(DataAccessAdapterManagerProbeImpl.java:50)
           at
    com.hp.ucmdb.discovery.probe.agents.probemgr.adapters.DataAccessAdaptersFacade.i
    nvokeOperation(DataAccessAdaptersFacade.java:406)
           at
    com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runChangesOnPo
    pulateChangesAdapter(AdapterService.java:1262)
           at
    com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runQueriesOnPo
    pulateChangesAdapter(AdapterService.java:1102)
           at com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runQueries
    (AdapterService.java:354)
```

```
        at
com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.runDiscovery
(AdapterService.java:198)
        at com.hp.ucmdb.discovery.probe.services.dynamic.core.AdapterService.discover
(AdapterService.java:149)
        at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecuter.JobExecuter.launchTask
(JobExecuter.java:1188)
        at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecuter.JobExecuter$JobExecute
rWorker.launch(JobExecuter.java:945)
        at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecuter.JobExecuter$JobExecute
rWorker.executeTask(JobExecuter.java:867)
        at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskexecuter.JobExecuter$JobExecute
rWorker.run(JobExecuter.java:728)
```

Solution

Search for text “has no matching mapping” to find out the missing mapping file name, and then add the mapping file to the adapter package.

Troubleshooting Federation Issues

When a federation error occurs, you can check the error message in the federation log file to identify the root cause and then solve the problem.

For more information, see the following:

"How to Check the Error Message of a Failed Federation Request" below

"Typical Federation Error Messages and Solutions" on the next page

How to Check the Error Message of a Failed Federation Request

When a federation query fails, a pop-up window similar to the following appears in the UCMDB studio.

Note: To troubleshoot push, population, and federation issues, you need to set the Development adapter log level to DEBUG or TRACE in the **fcmdb.properties** and **fcmdb.push.properties** files, which are located in the \conf\log folder of your Data Flow Probe or Integration Service installation. Additionally, you can view push, population and federation log information in the **fcmdb.push.all.log** file, which is located in the \runtime\log folder of your Data Flow Probe or Integration Service installation. Your Data Flow Probe or Integration Service may be installed on the same host as your UCMDB Server or on a separate host.

To set the Development adapter log level and view the detailed error message of a failed federation query, follow the steps described in "How to Check the Push Log File" on page 192.

Typical Federation Error Messages and Solutions

The following sections describe typical errors that may occur during federation and their solutions.

"Wrong Configuration for a Federation CI Type in smFedConf.xml" below

"Mapping File for the Federation TQL Query Is not Well Formed" on page 207

Wrong Configuration for a Federation CI Type in smFedConf.xml

Error Message

If you have not correctly configured a CI type in the smFedConf.xml file, UCMDB cannot federate CIs of this CI type. When you run a related federation query, the UCMDB studio will pop up an error window; additionally, you can find an error message that resembles the following in the federation log file:

```
2014-11-20 13:22:52,055 [AdHoc:AD_HOC_TASK_PATTERN_ID-51-1416460969254] ERROR - sm
    >> Failed to retrieve or parsing Root/parent CI message from SM side, exit from
    federation!
    java.lang.NullPointerException
           at
    com.mercury.topaz.fcmdb.adapters.serviceDeskAdapter.federation.SmGenericFederato
    r.processRootCiQueryByPage(SmGenericFederator.java:461)
```

```
	at
com.mercury.topaz.fcmdb.adapters.serviceDeskAdapter.federation.SmGenericFederato
r.generateRootRtnResult(SmGenericFederator.java:162)
	at
com.mercury.topaz.fcmdb.adapters.serviceDeskAdapter.federation.SmGenericFederato
r.generateFederationRtnResult(SmGenericFederator.java:131)
	at
com.mercury.topaz.fcmdb.adapters.serviceDeskAdapter.federation.SmGenericFederato
r.generateFederationOutput(SmGenericFederator.java:120)
	at
com.mercury.topaz.fcmdb.adapters.serviceDeskAdapter.federation.SmGenericFederato
r.getNextResultChunk(SmGenericFederator.java:578)
	at com.hp.ucmdb.adapter.smpush.ServiceManagerGenericAdapter.populate
(ServiceManagerGenericAdapter.java:363)
	at com.hp.ucmdb.adapters.GenericAdapter.getDataResult(GenericAdapter.java:740)
	at
com.hp.ucmdb.discovery.probe.processor.FederationTopologyGetDataResultProbeReque
stProcessor.processRequest
(FederationTopologyGetDataResultProbeRequestProcessor.java:40)
	at
com.hp.ucmdb.discovery.probe.processor.FederationTopologyGetDataResultProbeReque
stProcessor.processRequest
(FederationTopologyGetDataResultProbeRequestProcessor.java:26)
	at com.hp.ucmdb.discovery.probe.processor.AbstractProbeProcessor.process
(AbstractProbeProcessor.java:56)
	at com.hp.ucmdb.discovery.probe.processor.AbstractProbeProcessor.process
(AbstractProbeProcessor.java:19)
	at
com.hp.ucmdb.discovery.probe.agents.probemgr.adhoctasks.AdHocProbeRequestOperati
on.performAction(AdHocProbeRequestOperation.java:63)
	at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskdispatcher.AdHocTaskDispatcher.
dispatchTask(AdHocTaskDispatcher.java:70)
	at sun.reflect.GeneratedMethodAccessor75.invoke(Unknown Source)
	at sun.reflect.DelegatingMethodAccessorImpl.invoke
(DelegatingMethodAccessorImpl.java:43)
	at java.lang.reflect.Method.invoke(Method.java:601)
	at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2
(StandardMBeanIntrospector.java:111)
	at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2
(StandardMBeanIntrospector.java:45)
	at com.sun.jmx.mbeanserver.MBeanIntrospector.invokeM
(MBeanIntrospector.java:235)
	at com.sun.jmx.mbeanserver.PerInterface.invoke(PerInterface.java:138)
	at com.sun.jmx.mbeanserver.MBeanSupport.invoke(MBeanSupport.java:252)
	at javax.management.StandardMBean.invoke(StandardMBean.java:405)
	at com.sun.jmx.interceptor.DefaultMBeanServerInterceptor.invoke
(DefaultMBeanServerInterceptor.java:819)
	at com.sun.jmx.mbeanserver.JmxMBeanServer.invoke(JmxMBeanServer.java:792)
```

```
        at javax.management.MBeanServerInvocationHandler.invoke
    (MBeanServerInvocationHandler.java:305)
        at org.springframework.jmx.access.MBeanClientInterceptor.doInvoke
    (MBeanClientInterceptor.java:405)
        at org.springframework.jmx.access.MBeanClientInterceptor.invoke
    (MBeanClientInterceptor.java:353)
        at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
    (ReflectiveMethodInvocation.java:172)
        at org.springframework.aop.framework.JdkDynamicAopProxy.invoke
    (JdkDynamicAopProxy.java:202)
        at com.sun.proxy.$Proxy51.dispatchTask(Unknown Source)
        at
    com.hp.ucmdb.discovery.probe.agents.probegw.managementtasks.adhoctasks.AdhocThre
    ad.run(AdhocThread.java:54)
        at org.eclipse.jetty.util.thread.QueuedThreadPool.runJob
    (QueuedThreadPool.java:599)
        at org.eclipse.jetty.util.thread.QueuedThreadPool$3.run
    (QueuedThreadPool.java:534)
        at java.lang.Thread.run(Thread.java:722)
```

Solution

Open the smFedConf.xml file (which is located in the Configuration Files folder of the ServiceManagerEnhancedAdapter9-x adapter), and make sure the configuration for the CI type is correct.

Mapping File for the Federation TQL Query Is not Well Formed

Error Message

There are cases where the mapping file for a federation TQL query is not well formed. For example, target_mapping_name “priority” in the following example federation mapping file specifies a wrong Groovy function name (SMFederationFunctions.getEnumValue1):

```
<?xml version="1.0" encoding="UTF-8"?>
    <integration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:noNamespaceSchemaLocation="../mappings_schema.xsd">
        <info>
            <source name="SM"         version="9.40"    vendor="HP"/>
            <target name="UCMDB"    version="10.20"    vendor="HP"/>
        </info>
        <import>
            <scriptFile path="mappings.scripts.SMFederationFunctions"/>
        </import>
        <target_entities>
            <source_instance query-name="SM Incident 2.0" root-element-
    name="ucmdbIncident">
```

```
            <target_entity name="incident">
                <target_mapping name="reference_number"     datatype="STRING"
 value="ucmdbIncident['IncidentID']"/>
                <target_mapping name="name"                 datatype="STRING"
 value="ucmdbIncident['BriefDescription']"/>
                <target_mapping name="priority"             datatype="STRING"
 value="SMFederationFunctions.getEnumValue1(ucmdbIncident
['PriorityCode'],'Priority')"/>
                <target_mapping name="incident_status"         datatype="STRING"
    value="SMFederationFunctions.firstLetterToLowerAndReplaceSpaceWithUnderscore
(ucmdbIncident['IMTicketStatus'])"/>
                <target_mapping name="category"             datatype="STRING"
 value="SMFederationFunctions.replaceSpaceWithUnderscore(ucmdbIncident
['Category'])"/>
                <target_mapping name="closed_time"             datatype="DATE"
  value="SMFederationFunctions.convertDate(ucmdbIncident['ClosedTime'])"/>
                <target_mapping name="create_time"             datatype="DATE"
  value="SMFederationFunctions.convertDate(ucmdbIncident['OpenTime'])"/>
                <target_mapping name="last_modified_time"     datatype="DATE"
 value="SMFederationFunctions.convertDate(ucmdbIncident['UpdatedTime'])"/>
                <target_mapping name="impact_scope"         datatype="STRING"
 value="SMFederationFunctions.getEnumValue(ucmdbIncident
['ImpactScope'],'ImpactScope')"/>
                <target_mapping name="urgency"                 datatype="STRING"
    value="SMFederationFunctions.getEnumValue(ucmdbIncident
['Urgency'],'Urgency')"/>
            </target_entity>
        </source_instance>
    </target_entities>
</integration>
```

When you try to federate incidents, no records are retrieved from Service Manager. If you check the fcmdb.push.all.log file, you can find a detailed error message that resembles the following for the federation query:

```
2014-11-20 14:09:58,670 [AdHoc:AD_HOC_TASK_PATTERN_ID-66-1416463796366] ERROR -  >>
   Failed executing value [SMFederationFunctions.getEnumValue1(ucmdbIncident
   ['PriorityCode'],'Priority')] of mapping <target_mapping name="priority">
   <target_ci_type name="incident"> , Root cmdbId [null]
   groovy.lang.MissingMethodException: No signature of method: static
   mappings.scripts.SMFederationFunctions.getEnumValue1() is applicable for
   argument types: (java.lang.String, java.lang.String) values: [1, Priority]
   Possible solutions: getEnumValue(java.lang.String, java.lang.String)
          at groovy.lang.MetaClassImpl.invokeStaticMissingMethod(MetaClassImpl.java:1359)
          at groovy.lang.MetaClassImpl.invokeStaticMethod(MetaClassImpl.java:1345)
          at org.codehaus.groovy.runtime.callsite.StaticMetaClassSite.call
   (StaticMetaClassSite.java:50)
          at org.codehaus.groovy.runtime.callsite.CallSiteArray.defaultCall
   (CallSiteArray.java:42)
          at org.codehaus.groovy.runtime.callsite.AbstractCallSite.call
```

```
(AbstractCallSite.java:108)
	at org.codehaus.groovy.runtime.callsite.AbstractCallSite.call
(AbstractCallSite.java:120)
	at Mapping_4a8761e7adc009e8a7d268c2f1349ab4.run(Mapping_
4a8761e7adc009e8a7d268c2f1349ab4.groovy:1)
	at
com.hp.ucmdb.adapters.instance.mapping.AbstractResultTreeNodeMapper.calculatePro
perties(AbstractResultTreeNodeMapper.java:231)
	at
com.hp.ucmdb.adapters.instance.mapping.AbstractResultTreeNodeMapper.processTarge
tEntity(AbstractResultTreeNodeMapper.java:301)
	at com.hp.ucmdb.adapters.instance.mapping.RtnToRtnMapper.processTargetEntities
(RtnToRtnMapper.java:214)
	at
com.hp.ucmdb.adapters.instance.mapping.RtnToRtnMapper.processSourceInstanceWrapp
er(RtnToRtnMapper.java:101)
	at com.hp.ucmdb.adapters.instance.mapping.RtnToRtnMapper.buildResultTreeNode
(RtnToRtnMapper.java:76)
	at
com.hp.ucmdb.adapters.GenericAdapter$PopulationFederationChunkProcessor.invoke
(GenericAdapter.java:1213)
	at com.hp.ucmdb.adapters.GenericAdapter.getDataResult(GenericAdapter.java:743)
	at
com.hp.ucmdb.discovery.probe.processor.FederationTopologyGetDataResultProbeReque
stProcessor.processRequest
(FederationTopologyGetDataResultProbeRequestProcessor.java:40)
	at
com.hp.ucmdb.discovery.probe.processor.FederationTopologyGetDataResultProbeReque
stProcessor.processRequest
(FederationTopologyGetDataResultProbeRequestProcessor.java:26)
	at com.hp.ucmdb.discovery.probe.processor.AbstractProbeProcessor.process
(AbstractProbeProcessor.java:56)
	at com.hp.ucmdb.discovery.probe.processor.AbstractProbeProcessor.process
(AbstractProbeProcessor.java:19)
	at
com.hp.ucmdb.discovery.probe.agents.probemgr.adhoctasks.AdHocProbeRequestOperati
on.performAction(AdHocProbeRequestOperation.java:63)
	at
com.hp.ucmdb.discovery.probe.agents.probemgr.taskdispatcher.AdHocTaskDispatcher.
dispatchTask(AdHocTaskDispatcher.java:70)
	at sun.reflect.GeneratedMethodAccessor75.invoke(Unknown Source)
	at sun.reflect.DelegatingMethodAccessorImpl.invoke
(DelegatingMethodAccessorImpl.java:43)
	at java.lang.reflect.Method.invoke(Method.java:601)
	at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2
(StandardMBeanIntrospector.java:111)
	at com.sun.jmx.mbeanserver.StandardMBeanIntrospector.invokeM2
(StandardMBeanIntrospector.java:45)
	at com.sun.jmx.mbeanserver.MBeanIntrospector.invokeM
```

```
(MBeanIntrospector.java:235)
       at com.sun.jmx.mbeanserver.PerInterface.invoke(PerInterface.java:138)
       at com.sun.jmx.mbeanserver.MBeanSupport.invoke(MBeanSupport.java:252)
       at javax.management.StandardMBean.invoke(StandardMBean.java:405)
       at com.sun.jmx.interceptor.DefaultMBeanServerInterceptor.invoke
(DefaultMBeanServerInterceptor.java:819)
       at com.sun.jmx.mbeanserver.JmxMBeanServer.invoke(JmxMBeanServer.java:792)
       at javax.management.MBeanServerInvocationHandler.invoke
(MBeanServerInvocationHandler.java:305)
       at org.springframework.jmx.access.MBeanClientInterceptor.doInvoke
(MBeanClientInterceptor.java:405)
       at org.springframework.jmx.access.MBeanClientInterceptor.invoke
(MBeanClientInterceptor.java:353)
       at org.springframework.aop.framework.ReflectiveMethodInvocation.proceed
(ReflectiveMethodInvocation.java:172)
       at org.springframework.aop.framework.JdkDynamicAopProxy.invoke
(JdkDynamicAopProxy.java:202)
       at com.sun.proxy.$Proxy51.dispatchTask(Unknown Source)
       at
com.hp.ucmdb.discovery.probe.agents.probegw.managementtasks.adhoctasks.AdhocThre
ad.run(AdhocThread.java:54)
       at org.eclipse.jetty.util.thread.QueuedThreadPool.runJob
(QueuedThreadPool.java:599)
       at org.eclipse.jetty.util.thread.QueuedThreadPool$3.run
(QueuedThreadPool.java:534)
       at java.lang.Thread.run(Thread.java:722)
```

Solution

Search for word "Exception" in the fcmdb.push.all.log file to find the wrong method name, and then correct it in the corresponding federation mapping file.

Send Documentation Feedback

If you have comments about this document, you can contact the documentation team by email. If an email client is configured on this system, click the link above and an email window opens with the following information in the subject line:

Feedback on Universal CMDB Integration Guide (Using Service Manager Enhanced Generic Adapter) (Service Manager Service Manager 9.40; Universal CMDB 10.20 or later)

Just add your feedback to the email and click send.

If no email client is available, copy the information above to a new message in a web mail client, and send your feedback to ovdoc-itsm@hp.com.

We appreciate your feedback!


Java™
COMPATIBLE