



Configuration Management System (CMS)

Software Version: Content Pack 28.00 (CP28)

Discovery and Integrations Content Guide - Micro Focus Integrations

Document Release Date: August 2018
Software Release Date: August 2018



Legal Notices

Disclaimer

Certain versions of software and/or documents ("Material") accessible here may contain branding from Hewlett-Packard Company (now HP Inc.) and Hewlett Packard Enterprise Company. As of September 1, 2017, the Material is now offered by Micro Focus, a separately owned and operated company. Any reference to the HP and Hewlett Packard Enterprise/HPE marks is historical in nature, and the HP and Hewlett Packard Enterprise/HPE marks are the property of their respective owners.

Warranty

The only warranties for products and services of Micro Focus and its affiliates and licensors ("Micro Focus") are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. Micro Focus 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. Except as specifically indicated otherwise, a valid license from Micro Focus is 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

© 2011 - 2018 Micro Focus or one of its affiliates.

Trademark Notices

MICRO FOCUS and the Micro Focus logo, among others, are trademarks or registered trademarks of Micro Focus (IP) Limited or its subsidiaries in the United Kingdom, United States and other countries. All other marks are the property of their respective owners.

Adobe™ is a trademark of Adobe Systems Incorporated.

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

UNIX® is a registered trademark of The Open Group.

Documentation Updates

To check for recent updates or to verify that you are using the most recent edition of a document, go to: <https://softwaresupport.softwaregrp.com>.

This site requires that you register for a Software Passport and to sign in. To register for a Software Passport ID, click **Register for Software Passport** on the Micro Focus Support website at <https://softwaresupport.softwaregrp.com>.

You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your Micro Focus sales representative for details.

Support

Visit the Micro Focus Support site at: <https://softwaresupport.softwaregrp.com>.

This website provides contact information and details about the products, services, and support that Micro Focus offers.

Micro Focus 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 Micro Focus 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 a Software Passport user and to sign in. Many also require a support contract. To register for a Software Passport ID, click **Register for Software Passport** on the Micro Focus Support website at <https://softwaresupport.softwaregrp.com>.

To find more information about access levels, go to: <https://softwaresupport.softwaregrp.com/web/softwaresupport/access-levels>.

Integration Catalog accesses the Micro Focus Integration Catalog website. This site enables you to explore Micro Focus Product Solutions to meet your business needs, includes a full list of Integrations between Micro Focus Products, as well as a listing of ITIL Processes. The URL for this website is <https://softwaresupport.softwaregrp.com/km/KM01702731>.

Contents

Chapter 1: Micro Focus Application Portfolio Management (APM)	
Push Integration	9
Overview	9
How Data is Synchronized Between Micro Focus APM and UCMDB	10
Supported Versions	11
How to Integrate UCMDB and Micro Focus APM	11
Create an Integration Point between Micro Focus APM and UCMDB	12
(Optional) Push CI Data from UCMDB to Micro Focus APM	15
Schedule Data Push Jobs	18
View UCMDB Data in Micro Focus APM	19
Customize the Integration	19
Overview	19
Customize an Existing Mapping	20
Add a New Mapping to the Integration	21
Developer References	24
Adapter	24
Default Entity and Field Mappings between Micro Focus APM and UCMDB	25
Default Field Mappings between Micro Focus APM Application and UCMDB BusinessApplication	26
Default Field Mappings between Micro Focus APM Process and UCMDB BusinessProcess	28
Default Field Mappings between Micro Focus APM Location and UCMDB Location	29
Default Field Mappings between Micro Focus APM Server and UCMDB Node	30
REST APIs Called in the Integration	31
Delete a Request	31
Troubleshooting and Limitations – APM Push Integration	32
Limitations	33
Chapter 2: Micro Focus Asset Manager Integration	36
Chapter 3: Micro Focus Configuration Manager – Federating KPI Data	38

Overview	39
How to Consume Federated KPI Data from Configuration Manager	40
Troubleshooting and Limitations – Federating KPI Data	42
Chapter 4: Micro Focus Configuration Manager – Federating Policy	
Data	43
Overview	44
How to Consume Federated Policy Data from Configuration Manager	45
Troubleshooting and Limitations – Federating Policy Data	49
Chapter 5: Micro Focus Discovery and Dependency Mapping	
Inventory Integration	51
Overview	52
Supported Versions	52
DDMI Adapter	52
How to Populate the CMDB with Data from DDMI	54
How to Federate Data with DDMI	55
How to Customize the Integration Data Model in UCMDB	56
Predefined Queries for Population Jobs	57
DDMI Adapter Configuration Files	58
Troubleshooting and Limitations – DDMI Integration	58
Chapter 6: Micro Focus IT Executive Scorecard	60
Overview	61
Supported Versions	61
How to Push Data from UCMDB to Executive Scorecard	61
Executive Scorecard Adapter	64
Chapter 7: Micro Focus Network Automation (NA) Integration	65
Overview	66
Supported Versions	66
Topology	66
How to Pull Data Topology from a Micro Focus NA Server using a Java Client	67
How to Pull LastSnapshotAttemptDate from NA	71
Pull Topology from Network Automation Adapter	72
Limitations – NA Integration	74
Chapter 8: Micro Focus Network Node Manager (NNMi) Integration ..	75
Overview	76

Use Cases	76
Supported Versions	76
NNMi - UCMDB Integration Architecture	77
Topology	78
Pull Topology from NNMi Adapter	78
Pull Topology from NNMi by REST API Adapter	79
How to Run NNMi–UCMDB Integration	79
How to Manually Add the IpAddress CI of the NNMi Server	82
How to Set Up NNMi–UCMDB Integration	83
Pull Topology from NNMi Adapter	84
Pull Topology from NNMi by REST API Adapter	89
NNMi Update IDs Adapter	91
How to Customize Integration	92
Troubleshooting and Limitations – NNMi Integration	96
Chapter 9: Micro Focus SMA-X Push Integration	99
Overview	100
How Data is Synchronized Between UCMDB and SMA-X	100
Supported Versions	101
How to Integrate UCMDB and SMA-X	101
Set up UCMDB	101
Push CI Data from UCMDB to SMA-X	104
Schedule Data Push Jobs	107
Tailor the Integration	108
Integration Architecture	108
Data Flow Architecture	108
Integration TQL Queries	109
SMA-X Rules and Flows	109
Data Mapping	109
Push Mapping	109
Change Adapter Settings	111
Customize an Existing Mapping	112
Add a New Mapping to the Integration	113
Troubleshooting and Limitations – SMA-X Push Integration	116
Chapter 10: Micro Focus Service Manager Integration	117
Chapter 11: Micro Focus Storage Essentials (SE) Integration	118
Overview	120

Supported Versions	120
How to Perform the SE Integration	120
Storage Essentials Integration Packages	122
Adapter Parameters	123
Discovered CITs and Relationships	123
Node Details	125
SAN Topology	126
Storage Topology	127
Views	127
Storage Array Details	128
FC Switch Details	128
FC Switch Virtualization	129
Storage Pool Details	129
Host Storage Details	130
SAN External Storage	130
SAN Topology	131
Storage Topology	132
FC Port to FC Port	132
Impact Analysis Rules	133
FC Port to FC Port	133
FC Switch Devices to FC Switch	133
Host Devices to Host	134
Logical Volume to Logical Volume	134
Storage Array Devices to Storage Array	135
Reports	135
Storage Array Configuration	135
Host Configuration	136
Storage Array Dependency	136
Host Storage Dependency	136
Storage Pool Configuration	137
Troubleshooting and Limitations – SE Integration	137
Chapter 12: Micro Focus Storage Operations Manager (SOM)	
Integration	139
Overview	140
Supported Versions	140
How to Perform the SOM Integration	140

Install the Integration	141
Enable the Integration	141
Adapter Properties	144
Discovered CITs and Relationships	145
Node Details	146
SAN Topology	147
Storage Topology	148
Views	148
Storage Array Details	149
FC Switch Details	150
FC Switch Virtualization	150
Storage Pool Details	151
Host Storage Details	151
SAN External Storage	152
SAN Topology	152
Storage Topology	153
FC Port to FC Port	153
Impact Analysis Rules	154
FC Port to FC Port	155
FC Switch Devices to FC Switch	155
Host Devices to Host	156
Logical Volume to Logical Volume	156
Storage Array Devices to Storage Array	156
Reports	157
Storage Array Configuration	157
Host Configuration	158
Storage Array Dependency	158
Host Storage Dependency	159
Storage Pool Configuration	159
Troubleshooting the SOM Integration	159
Known Issues	160
Send documentation feedback	162

Chapter 1: Micro Focus Application Portfolio Management (APM) Push Integration

This chapter includes:

Overview	9
How Data is Synchronized Between Micro Focus APM and UCMDB	10
Supported Versions	11
How to Integrate UCMDB and Micro Focus APM	11
Create an Integration Point between Micro Focus APM and UCMDB	12
(Optional) Push CI Data from UCMDB to Micro Focus APM	15
Schedule Data Push Jobs	18
View UCMDB Data in Micro Focus APM	19
Customize the Integration	19
Overview	19
Customize an Existing Mapping	20
Add a New Mapping to the Integration	21
Developer References	24
Adapter	24
Default Entity and Field Mappings between Micro Focus APM and UCMDB	25
REST APIs Called in the Integration	31
Troubleshooting and Limitations – APM Push Integration	32
Limitations	33

Overview

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

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

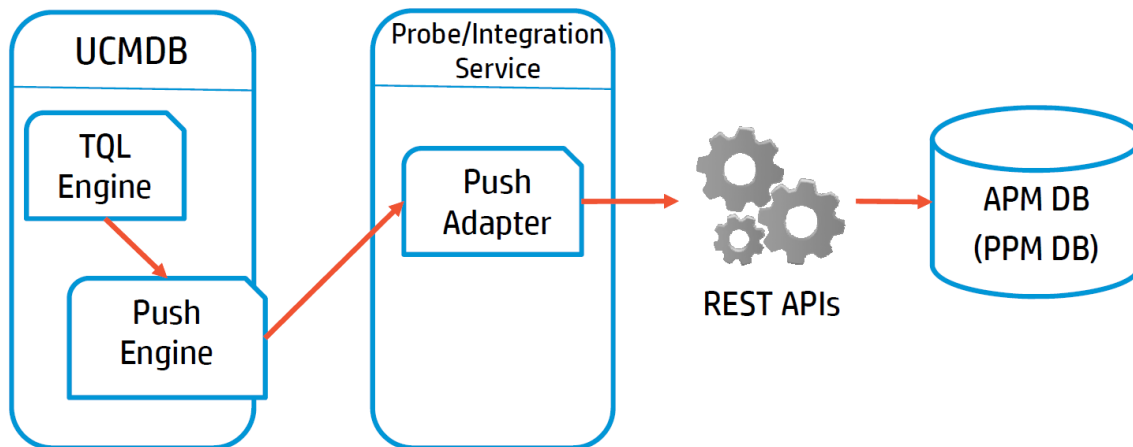
The following table provides an overview of the Micro Focus APM integration with UCMDB:

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

How Data is Synchronized Between Micro Focus APM and UCMDB

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

The following graphic shows the high-level components of the integration:



Note: The Push Adapter is executed in the Data Flow Probe/Integration Service process.

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

The **UCMDB Push Engine**:

- Retrieves the required data from UCMDB, using the given TQL query.
- Filters the data to include only the data that has changed since the last execution of this synchronization.

- Splits the data into multiple chunks without breaking consistency.
- Sends the information to the Probe/Adapter.

The Push Adapter is a generic framework for easily configuring push adapters, using only XML and [Groovy](#). It allows easy mapping of the data from the UCMDDB data model into the Micro Focus APM data model, and the transfer of this converted data into the Micro Focus APM database through REST APIs called from Micro Focus APM.

For more information about push adapter, see **Developing Push Adapters** in the *Developer Reference section of the UCMDDB Help*.

For details about REST APIs that this integration call from Micro Focus APM, see ["REST APIs Called in the Integration" on page 31](#).

For entity mappings and field mappings between Micro Focus APM and UCMDDB, see ["Default Entity and Field Mappings between Micro Focus APM and UCMDDB" on page 25](#).

Supported Versions

The APM adapter supports the following:

- Universal CMDB version 10.00 and later
- Project and Portfolio Management Center (PPM Center) version 9.22 (and later) where APM for PPM 9.20 is installed

How to Integrate UCMDDB and Micro Focus APM

To set up integration between UCMDDB and Micro Focus APM, you must complete the following steps:

- ["Create an Integration Point between Micro Focus APM and UCMDDB" on the next page](#)
- ["\(Optional\) Push CI Data from UCMDDB to Micro Focus APM" on page 15](#)
- ["Schedule Data Push Jobs" on page 18](#)

Create an Integration Point between Micro Focus APM and UCMDB



1. Log on to UCMDB as an administrator.
2. Navigate to **Data Flow Management > Integration Studio**.




UCMDB displays a list of existing integration points.

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

The New Integration Point dialog box opens.

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

Field (*Required)	Description
Integration Properties section	
*Integration Name	Type the name (unique key) of the integration point.
Integration Description	Type a description of the current integration point.
*Adapter	Click the Select Adapter  button and select HP Software Products / Micro Focus Products > APM > APM Push Adapter from the Select Adapter dialog.
*Is Integration Activated?	Select this option to indicate the integration point is active.
Adapter Properties section	
*Protocol Type	Select http or https from the drop-down list
*Hostname/IP	Type the hostname or IP Address of the PPM Server. For example, 16.166.16.16 or hostname.
*Port	Type the communication port of the PPM Server. The default value is 80. Example: 30000
*Path	Type the path of the PPM Server. The default value is itg.
*Credentials ID	Click the Select Credential Id  button, then select Generic

Field (*Required)	Description
	<p>Protocol or DefaultDomain > Generic Protocol in the Protocol pane, and from the Credentials list, select a credential or create a new credential that is to be used by UCMDB to access APM.</p> <p>To create a credential for this integration point,</p> <ol style="list-style-type: none"> Click the Select Credential Id  button. The Choose Credentials dialog opens. Click the Create new connection details for selected protocol type  button. The Generic Protocol Parameters dialog opens. Provide values for the following fields and click OK: <ul style="list-style-type: none"> Network Scope: Use the default value ALL. User Label: Type a label for the credential. User Name: Provide the user name for the APM account that is to be used by UCMDB to access APM. Password: Click  and provide the password for the APM account that is to be used by UCMDB to access APM. Click OK twice. <p>Tip: Users of any of the following security groups can be used as Credentials ID:</p> <ul style="list-style-type: none"> APM User APM Administrator APM Analyst
*Data Flow Probe	<p>The name of the Data Flow Probe/Integration service used to execute the synchronization from.</p> <p>Select IntegrationService for this integration.</p> <p>Note: If the IntegrationService option does not exist, consult with your UCMDB administrator for the best selection for your requirements.</p>
Additional Probes	Not required for this integration point.

Below is an example of the completed dialog:

The screenshot shows the 'New Integration Point' dialog box. It has a title bar with a close button. The main content area is divided into two sections: 'Integration Properties' and 'Adapter Properties'. The 'Integration Properties' section includes fields for 'Integration Name' (APM_UCMDB_integration_1), 'Integration Description' (APM-UCMDB integration point 1), 'Adapter' (APM Push Adapter), and a checked 'Is Integration Activated' checkbox. The 'Adapter Properties' section includes fields for 'Protocol Type' (http), 'Hostname/IP' (16.166.16.16), 'Port' (30000), 'Path' (itg), 'Credentials ID' (Generic Protocol: Generic Protocol Credential 1), 'Data Flow Probe' (IntegrationService), and an empty 'Additional Probes' field. At the bottom, there is a 'Test connection' button and 'OK' and 'Cancel' buttons.

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

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

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

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

If needed you may create or edit the existing job. To create or edit and run a customized data push job, see ["\(Optional\) Push CI Data from UCMDB to Micro Focus APM" on the next page.](#)

For instructions about scheduling the data push job, see ["Schedule Data Push Jobs" on page 18.](#)

(Optional) Push CI Data from UCMDB to Micro Focus APM

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



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

1. Log on to UCMDB as an administrator.
2. Navigate to **Data Flow Management > Integration Studio**.

UCMDB displays a list of existing integration points.

3. Select the integration point you created for APM.
4. Select the default data push job **APM Push**.

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

- a. Click the **New Integration Job**  button in the right panel.
- b. In the **Name** field, type a unique name for the job.
- c. Click the **Add Query**  button to add existing TQL queries to the job.

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

Note: To access these OOTB TQL queries for push, navigate to **Modelling > Modeling Studio > Resources**, select **Queries** from the drop-down list for the **Resource Type** field and then navigate to **Root > Integration > APM Push**.

TQL Query	Description
APM Location Push	Pushes Location CIs. Mapping XML: pushMappingAPMLocation.xml
APM Process Push	Pushes BusinessProcess CIs.

TQL Query	Description
	Mapping XML: pushMappingAPMProcess.xml
APM Process Relation Clear Push	<p>Clears old relations between processes in APM.</p> <p>This TQL query synchronization must have been run BEFORE the 'APM Process Relation Push' TQL query synchronization.</p> <p>Mapping XML: pushMappingAPMProcessRClear.xml</p>
APM Process Relation Push	<p>Pushes relations between Processes (pushed by APM Process Push) to other business elements or to processes.</p> <p>BusinessProcess CIs must have been pushed before this TQL query synchronization in the 'APM Process Push' TQL query synchronization.</p> <p>Mapping XML: pushMappingAPMProcessR.xml</p>
APM Server Push	<p>Pushes Node CIs (Computers, Network Devices, etc.).</p> <p>Mapping XML: pushMappingAPMServer.xml</p>
APM Server Relation Clear Push	<p>Clears old relations between Servers in APM.</p> <p>This TQL query synchronization must have been run BEFORE the 'APM Server Relation Push' TQL query synchronization.</p> <p>Mapping XML: pushMappingAPMServerRClear.xml</p>
APM Server Relation Push	<p>Pushes relations between Servers (pushed by APM Server Push) to other servers.</p> <p>Node CIs must have been pushed before this TQL query synchronization in the 'APM Server Push' TQL query synchronization.</p> <p>Mapping XML: pushMappingAPMServerR.xml</p>
APM Application Push	<p>Pushes BusinessApplication CIs.</p> <p>Location, BusinessProcess, and/or Node CIs must have been pushed before this TQL query synchronization.</p> <p>Mapping XML: pushMappingAPMApplication.xml</p>
APM Application Relation Clear Push	<p>Clears old relations (except for the downstream relations) between Applications in APM.</p> <p>This TQL query synchronization must have been run BEFORE the 'APM Application Relation Push' TQL query synchronization.</p> <p>Mapping XML: pushMappingAPMApplicationRClear.xml</p>




TQL Query	Description
APM Application Relation Push	<p>Pushes relations (except for the downstream relations) between Applications (pushed by APM Application Push) to other business elements or to nodes.</p> <p>BusinessApplication CIs must have been pushed before this TQL query synchronization in the 'APM Application Push' TQL query synchronization.</p> <p>Mapping XML: pushMappingAPMApplicationR.xml</p>
APM Application Relation Clear Down Push	<p>Clears old downstream relations between Applications in APM.</p> <p>This TQL query synchronization must have been run BEFORE the 'APM Application Relation Down Push' TQL query synchronization.</p> <p>Mapping XML: pushMappingAPMApplicationDownRClear.xml</p>
APM Application Relation Down Push	<p>Pushes downstream relations between Applications (pushed by APM Application Push) to other business elements or to nodes.</p> <p>BusinessApplication CIs must have been pushed before this TQL query synchronization in the 'APM Application Push' TQL query synchronization.</p> <p>Mapping XML: pushMappingAPMApplicationDownR.xml</p>

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

This allows deletion of synchronized data in APM when data in UCMDB are deleted.

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

Note: For scheduling configuration, see ["Schedule Data Push Jobs" on the next page.](#)

- e. Click **OK**.
- f. Save the integration point.
5. Run the job manually to see if the integration job works properly:
- To push all the relevant data for the job, click the **Full Synchronization**  button.
 - To push only the changes in the data since the job last executed, click the **Delta Synchronization**  button.
6. The job is in **Running** status. Wait for the job to complete; click the **Refresh**  button multiple times as needed until the job is completed.


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

7. When the job is completed, the job status becomes one of the following depending on the results:
 - Succeeded
 - Passed with failures
 - Failed
8. Click the **Statistics** tab to view the results; if any errors occur, click the **Query Status** tab and **Job Errors** tab for more information. For more information about errors, see ["Troubleshooting and Limitations – APM Push Integration" on page 32](#).

Schedule Data Push Jobs

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

1. Log on to UCMDB as an administrator.
2. Navigate to **Data Flow Management > Integration Studio**.

UCMDB displays a list of existing integration points.
3. Select the integration point you created for the APM - UCMDB integration.
4. Select the APM Push job.
5. Click the **Edit Integration Job**  button.

The Edit Integration Job dialog opens.

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

6. Define a schedule for Changes Synchronization or Delta Synchronization.
 - a. Click on the **Changes Synchronization** tab (or **Delta Synchronization**).
 - b. Select the **Scheduler enabled** option.
 - c. Select the scheduling options you want to use.
7. Click the **All Data Synchronization** tab (or **Full Synchronization**) and select the scheduling options you want to use.
8. Click **OK**.
9. Save the integration point.

View UCMDB Data in Micro Focus APM

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

To view UCMDB data in Micro Focus APM,

1. Log on to PPM Center.
2. On the **Open** menu, click **Application Portfolio > Search Entities**.

The Search Entities page opens.

3. In the **Entities** section, click one of the following entities:
 - Application
 - Location
 - Process
 - Server

The Search: APM - <Entity> page opens.

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

4. Click **Search**.

The Search Results page displays request search results.

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

Customize the Integration

This section includes:

- ["Overview" below](#)
- ["Customize an Existing Mapping" on the next page](#)
- ["Add a New Mapping to the Integration" on page 21](#)

Overview

This section contains details about the architecture of the integration.

Data Flow Architecture

1. The Push Engine executes the TQL query.
2. For a differential flow, the data is compared to the last synchronized data, and only the changes are forwarded.
3. Data is converted into Composite CIs (instances of data according to the TQL Root elements).
4. Data is then pushed to the Push Adapter.
5. The Push Adapter loads the correct mapping for the specific TQL query.
6. All `dynamic_mappings` are executed and saved to maps, to allow usage in the next mapping stage.

For more information, see **Developing Push Adapters** in the *Developer Reference* section of the *UCMDB Help*.

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

Integration TQL Queries

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

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

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

Customize an Existing Mapping

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

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

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

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

- a. Navigate to **Modeling > Modeling Studio > Resources** and select the **Queries Resource Type**.
- b. Navigate to **Query: Root > Integration > APM Push > APM Location Push**.
- c. Select **Root**, right-click and select **Query Node Properties**.
- d. Go to the Element Layout tab.
- e. Move the **Name** to the Specific Attributes box.
- f. Click **OK**.
- g. Save the Query.

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

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

- a. Navigate to **Data Flow Management > Adapter Management > Packages > APMPushAdapter > Configuration Files > pushMappingAPMLocation.xml**.
- b. Navigate to the **<target_ci_type name="fields">** XML tag.
- c. Below the tag, add the following XML tag to hold the value of the Description:

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

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

- d. Click **OK**.


Add a New Mapping to the Integration

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

Step 1: Create a TQL Query

1. Navigate to **Modeling > Modeling Studio > New > Query**.
2. From the **CI Types** tab, add a **Location** to the query.
3. Right-click the **Location Query Node** and select **Query Node Properties**.
4. Rename the **Element Name** to **Root**.
5. Navigate to the **Element Layout** tab.
6. Select **Select attributes for layout**.
7. In the **Attributes condition** drop down, select **Specific Attributes**, and add the **Name** attribute
8. Click **OK**.
9. Save the query to **Root > Integration > APM Push > APM Location Push**.

Step 2: Create a Push-Mapping

1. Navigate to **Data Flow Management > Adapter Management > APMPushAdapter**.
2. Click the **Create new resource**  button and select **New Configuration File**.
3. Type the following Name: **APMPushAdapter/mappings/pushMappingAPMLocation.xml**.
4. Select the **APMPushAdapter** package.
5. Click **OK**.
6. Copy the following into the newly created XML file:

```
<integration xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <info>
    <source name="UCMDB" versions="10.0" vendor="HP"/>
    <target name="APM" versions="2.0" vendor="HP"/>
  </info>
  <import>
    <scriptFile path="mappings.scripts.APMPushFunctions"/>
  </import>
  <targetcis>
    <source_instance_type query-name="APM Location Push" root-element-
name="Root" >
```

```

        <target_ci_type name="request">
            <target_mapping name="uuid" datatype="STRING" value="Root['global_
id']"/>
            <target_mapping name="requestType" datatype="STRING" value="'APM -
Location'"/>
            <target_mapping name="description" datatype="STRING" value="Root
['name']"/>
            <target_ci_type name="fields">
                <target_mapping name="REQ.DESCRPTION" datatype="STRING"
value="APMPushFunctions.subString(Root['name'],200)"/>
            </target_ci_type>
        </target_ci_type>
    </source_instance_type>
</targetcis>
</integration>

```



In the following line:

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



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


7. Click **OK**.

Step 3: Create a Job with the New TQL Query

1. Navigate to **Data Flow Management > Integration Studio**.
2. Create an Integration Point with APM.
3. In the **Integration Jobs** tab, click the **New Integration Job**  button .
4. Insert a job name in the **Name** field.
5. Click the  button, and choose the **APM Location Push query**.
6. Click **OK**.

Step 4: Run the Job

1. Click on the job created in ["Step 3: Create a Job with the New TQL Query" above](#).
2. Click the All Data Synchronization  button (or **Full Synchronization** ).

3. Wait for the job to finish. You should click the **Refresh**  button to see progress.
4. Make sure that the status is **Succeeded**.

Step 6: View the Results

1. Log on to PPM.
2. On the **Open** menu,
 - Click **Application Portfolio > Search Entities**, then in the **Entities** section, click **Location**.
Or,
 - Click **Search > Requests**, then from the **Request Type** drop-down list, select **APM - Location**.
3. Click **Search**.

The Search Results page displays request search results.

Developer References

This section includes the following:

- ["Adapter" below](#)
- ["Default Entity and Field Mappings between Micro Focus APM and UCMDB" on the next page](#)
- ["REST APIs Called in the Integration" on page 31](#)

Adapter

This integration job uses the adapter called APMAAdapter.

Input CI Type

destination_config

Triggered CI Data

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

Adapter Parameters

Name	Value
credentialsId	
domain	itg
host	
port	80
probeName	
protocolType	http

Default Entity and Field Mappings between Micro Focus APM and UCMDB

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

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

Micro Focus APM Entity	PPM Center Request Type	UCMDB CI Type	Remarks
Application	APM - Application	BusinessApplication	For detailed mappings, see "Default Field Mappings between Micro Focus APM Application and UCMDB BusinessApplication" below .
Process	APM - Process	BusinessProcess	For detailed mappings, see "Default Field Mappings between Micro Focus APM Process and UCMDB BusinessProcess" on page 28 .
Location	APM - Location	Location	For detailed field mappings, see "Default Field Mappings between Micro Focus APM Location and UCMDB Location" on page 29 .
Server	APM - Server	Node	For detailed field mappings, see "Default Field Mappings between Micro Focus APM Server and UCMDB Node" on page 30 .

Default Field Mappings between Micro Focus APM Application and UCMDB BusinessApplication

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

Micro Focus APM Field Name and Field Type	UCMDB CI Attribute and Field Type
Name KNTA_PROJECT_NAME Text Field - 300	Name name string
Updated By ^a	(N/A)
Create On ^b CREATION_DATE Date	(N/A)
Purpose	Description

Micro Focus APM Field Name and Field Type	UCMDB CI Attribute and Field Type
APM_APP_PURPOSE Text Field - 4000	description string
Business Criticality ^c APM_RATING_BUSINESS_CRIT Drop Down List	BusinessCriticality business_criticality integer
Created By ^a CREATED_BY Auto Complete List	(N/A)
Supported Processes APM_SUPPORTED_PROCESSES Auto Complete List	Name (of CI Type BusinessProcess) name (of CI Type BusinessProcess) string
Downstream Applications APM_DOWNSTREAM_APPS Auto Complete List	Name (of downstream CI Type BusinessApplication) name (of downstream CI Type BusinessApplication) string
Upstream Applications APM_UPSTREAM_APPS Auto Complete List	Name (of upstream CI Type BusinessApplication) name (of upstream CI Type BusinessApplication) string
Service Level Agreement APM_APP_SLA Text Field - 200	Name (of CI Type ServiceLevelAgreement) name (of CI Type ServiceLevelAgreement) string
Servers APM_SERVER_LIST Auto Complete List	Name (of CI Type Node) name (of CI Type Node) string
Database APM_DATABASE_LIST Text Field - 200	Name (of CI Type Database) name (of CI Type Database) string
<p>a. The APM account you provided when creating the integration point (see "Create an Integration Point between Micro Focus APM and UCMDB" on page 12).</p> <p>b. Time when the request is created for the first time in APM.</p> <p>c. The following mapping rule is used for this mapping:</p> <pre><target_mapping name="REQD.APM_RATING_BUSINESS_CRIT" datatype="STRING" value="APMPushFunctions.getPropertyValue('bc', Root['business_ criticality']).toString() , ''"/></pre> <p>where the definition of 'bc' token is defined as follows in the <code>server.properties</code> field:</p> <pre>bc.0=0 - Least critical</pre>	

Micro Focus APM Field Name and Field Type	UCMDB CI Attribute and Field Type
bc.1=1 - Slightly critical bc.2=2 - Less than average bc.3=3 - More than average bc.4=4 - Critical bc.5=5 - Highly critical When synchronizing the Business Criticality field from UCMDB to APM, if the value is '1' in UCMDB, then the field value will be set to '1 - Slightly critical' in APM.	

Default Field Mappings between Micro Focus APM

Process and UCMDB BusinessProcess

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

Micro Focus APM Field Name, Database ID, and Field Type	UCMDB CI Attribute, Database ID, and Field Type
Process Name DESCRIPTION Text Field - 200	Name name string
Parent Process APM_PARENT Auto Complete List	Name of parent BusinessProcess name of parent BusinessProcess string
Description APM_DESCRIPTION Text Area - 4000	Description description string
Created By ^a CREATED_BY Auto Complete List	(N/A)
Created On ^b CREATION_DATE Date	(N/A)
a. The APM account you provided when creating the integration point (see "Create an Integration Point between Micro Focus APM and UCMDB" on page 12) b. Time when the request is created for the first time in APM.	

Default Field Mappings between Micro Focus APM Location and UCMDB Location

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

Micro Focus APM Field Name, Database ID, and Field Type	UCMDB CI Attribute, Database ID, and Field Type
Location Name DESCRIPTION Text Field - 200	Name name string
Description APM_DESCRIPTION Text Field - 4000	Description description string
Address APM_LOC_ADDRESS Text Field - 200	StreetAddress+ExtendedStreetAddress street_address+extended_street_address string+string
Postal Code APM_LOC_ZIPCODE Text Field - 20	PostalCode postal_code string
Country APM_LOC_COUNTRY Text Field - 200	CountryOrArea country_or_area string
Region APM_LOC_REGION DDL	Region region string
Longitude APM_LONGITUDE Text Field - 40	Longitude longitude string
Latitude APM_LATITUDE Text Field - 40	Latitude latitude string
City APM_LATITUDE Text Field - 200	City ^a city string
State/Province	State ^a

Micro Focus APM Field Name, Database ID, and Field Type	UCMDB CI Attribute, Database ID, and Field Type
APM_STATE Text Field - 200	state string
a. The CI attribute is removed from UCMDB version 10.x, but exists in earlier versions of UCMDB. For UCMDB instances that upgraded from an earlier version to 10.x, this CI attribute exists but is read-only. If the CI attribute has a value, the value can be synchronized to APM; otherwise the APM field remains empty after you run the synchronization push job in UCMDB.	

Default Field Mappings between Micro Focus APM Server and UCMDB Node

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

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

REST APIs Called in the Integration

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

- Get a request

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

- Create a request

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

- Update a request

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

- Delete a request

Added in version 9.22. For more information, see ["Delete a Request" below](#).

Delete a Request

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

HTTP Method: DELETE

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

Request path variables:

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

Response entity body:

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

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

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

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

Limitation:

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

Troubleshooting and Limitations – APM Push Integration

This section includes the following:

- ["Limitations" on the next page](#)
- ["Troubleshooting Problems" on page 34](#)
- ["Logs" on page 34](#)

Limitations

- The Data Flow Probe or Integration Service must be installed on a Windows OS.
- For requests created in APM from CIs pushed from UCMDB, any changes made in APM are overwritten when you run the data push job in UCMDB.
- The APM Application request form holds a single value for Location, therefore it is designed to push only one value for Location of the Application from the UCMDB BusinessApplication.

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

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

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

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

where the definition of 'bc' token is defined as follows in the

`<APMPushAdapter.zip>/mappings/scripts/server.properties` file:

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

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

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

- This integration does not support synchronizing values in languages that are not supported by UCMDB. For example, Simplified Chinese.
- The ";" character is not supported. If a UCMDB CI name contains a ";" character, it would be treated as a separator and two entries would show up in APM after synchronization.

Troubleshooting Problems

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

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

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

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

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

Example

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

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

Logs

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

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

- On the Data Flow Probe machine:
`..\DataFlowProbe\conf\log\fcmdb.push.properties`
- If using the integration service, on the UCMDB server:
`..\UCMDB Server\Integrations\conf\log\fcmdb.push.properties`

Change the log level to DEBUG:

```
loglevel=DEBUG
```

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

- On the Data Flow Probe machine:
 `..\DataFlowProbe\runtime\log\`
- If using the integration service, on the UCMDB server:
 `..\UCMDB Server\Integrations\runtime\log\`

Chapter 2: Micro Focus Asset Manager Integration

Integration between Universal CMDB (UCMDB) and Micro Focus Asset Manager (AM) enables you to share information between UCMDB and AM. Common use cases include pulling asset from AM and pushing inventory CIs like hardware, installed software, and business services from UCMDB to Asset Manager.

The UCMDB-AM integration is based on **AM Generic Adapter**. For instructions on how to set up the integration by using **AM Generic Adapter**, see documentation for [AM Generic Adapter 1.04](#).

Chapter 3: Micro Focus Configuration Manager – Federating KPI Data

This chapter includes:

Overview	39
How to Consume Federated KPI Data from Configuration Manager	40
Troubleshooting and Limitations – Federating KPI Data	42

Overview

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

- Use UCMDb's reporting functionality to generate and schedule reports on top of Configuration Manager's data.
- Consume Configuration Manager's data in other Micro Focus applications, such as Micro Focus Business Service Management.
- Use Configuration Manager's analysis data as a basis for making decisions in other applications.

Configuration Manager exposes the following data for federation:

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

UCMDb provides the class model for the schema for the model to be shared, and uses a federation TQL query as the way to consume data in UCMDb on the fly. For details, see "Federating KPIs" in the *Configuration Manager section of the UCMDb Help*.

KPI means **Key Performance Indicator**. The UCMDb provides the **CMKpiAdapter** to federate KPI information about policy and authorization status from Configuration Manager. The data that is federated from Configuration Manager populates the **Kpi** and **KpiObjective** CITs, and can be retrieved by TQL as described in ["Create KPI Reports" on page 41](#).

How to Consume Federated KPI Data from Configuration Manager




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


This task includes the following steps:

- ["Create an Integration Point to Federate KPI Data" below](#)
- ["Create KPI Reports" on the next page](#)

Create an Integration Point to Federate KPI Data

1. In the Integration Studio, create a new integration point.
2. Set the following adapter properties:

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

3. Click **Test Connection** to make sure that you have configured the integration point correctly.
4. Click **OK** to save the integration point.
5. Select the KPI and KPIObjective CI types in the Supported and Selected CI Types tree.
6. Click the  button to save the integration point.

For further details about creating integration points, see the section about the Integration Studio in the *Data Flow Management section of the UCMDB Help*.

Create KPI Reports

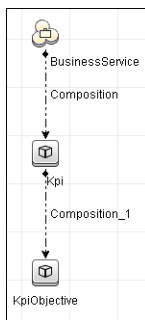
You can create KPI reports based on the CIs in a view, a custom TQL query, or business services.

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

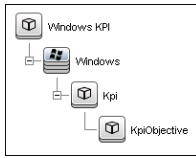
Note: When using a custom TQL query, make sure you take into account the limitations of the data capacity when using federation. You should filter the CIs in the TQL query to take this limitation into account.

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

Note: If you want to create a business services report, select the BusinessService CI type when creating the TQL query.



3. Specify the Configuration Manager integration point that you defined to be the data source that provides the policy and policy result data.
4. Set the hierarchy. An example is shown below:



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



6. If desired, you can schedule these reports to be created periodically. For details, see the *Data Flow Management* section of the *UCMDB Help*.

For details about creating reports, see the section about reports in the *Modeling* section of the *UCMDB Help*.

Troubleshooting and Limitations – Federating KPI Data

- Federation only works with CIs in the actual state. Therefore:
 - Policy compliance is federated only for CIs in the actual state.
 - The authorization status for CIs that were deleted from the actual state is not shown.
- SSL communication for KPI data federation is not supported.
- The maximum number of CIs that can be federated is configurable. For details about changing this number, edit the value of the **Max Num To Federate** setting in the Infrastructure Settings Manager in UCMDB. For details about changing settings, see the *Infrastructure Settings Manager* chapter in the *Administer* section of the *UCMDB Help*. The recommended number of CIs is no more than 2,000,000, if large views have been enabled in Configuration Manager. For details about enabling support for large views, see the section describing large capacity planning in the interactive *Universal CMDB Deployment Guide*.
- If the test connection fails, click **Details** and check the first error in the stack trace for more information.

Chapter 4: Micro Focus Configuration Manager – Federating Policy Data

This chapter includes:

Overview	44
How to Consume Federated Policy Data from Configuration Manager	45
Troubleshooting and Limitations – Federating Policy Data	49

Overview

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

- Use UCMDB's reporting functionality to generate and schedule reports on top of Configuration Manager's data.
- Consume Configuration Manager's data in other Micro Focus applications, such as Micro Focus Business Service Management.
- Use Configuration Manager's analysis data as a basis for making decisions in other applications.

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

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

UCMDB provides the **CMPolicyAdapter** to federate policy data from Configuration Manager, which populates the **Policy** and **PolicyResult** CITs, and can be retrieved by TQL as described in ["Create Policy Reports Based on CIs in a View or Custom TQL query" on page 46](#) and ["Create summary policy reports based on the CIs in a view or a custom TQL query" on page 47](#).

How to Consume Federated Policy Data from Configuration Manager




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

This task includes the following steps:


- ["Create an Integration Point to Federate Policy Compliance Data" below](#)
- ["Create Policy Reports Based on CIs in a View or Custom TQL query" on the next page](#)
- ["Create summary policy reports based on the CIs in a view or a custom TQL query" on page 47](#)

Create an Integration Point to Federate Policy Compliance Data

1. In the Integration Studio, create a new integration point.
2. Set the following adapter properties:

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

Field	Description
Integration Name	Enter a name for the new integration point.
Port	Enter the port number that is used for communication with the Configuration Manager database.

3. Click **Test Connection** to make sure that you have configured the integration point correctly.
4. Click **OK** to save the integration point.
5. Select the Policy and PolicyResults CI types in the Supported and Selected CI Types tree.
6. Click the  button to save the integration point.

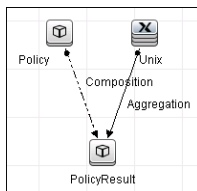
For further details about creating integration points, see the section about the Integration Studio in the *Data Flow Management section of the UCMDB Help*.

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

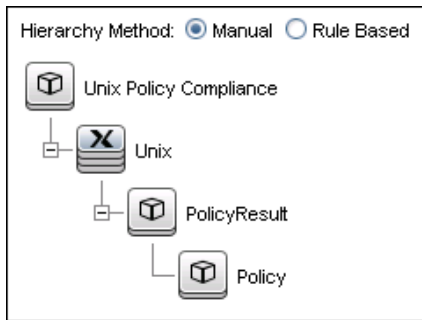
1. Create an integration point as described in , if one does not already exist.
2. In UCMDB, create a new view with a custom TQL query, or copy an existing view.

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

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



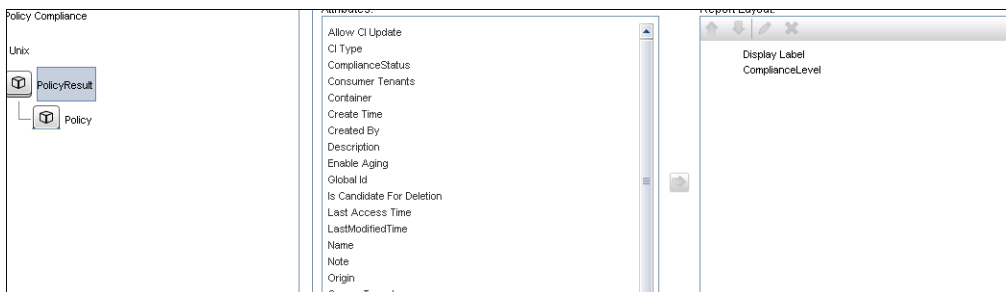
4. Specify the Configuration Manager integration point that you defined to be the data source that provides the policy and policy result data.
5. Set the hierarchy. An example is shown below:



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



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



8. If desired, you can schedule these reports to be created periodically. For details, see the *Data Flow Management section of the UCMDB Help*.

For details about creating reports, see the section about reports in the *Modeling section of the UCMDB Help*.

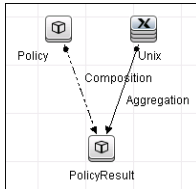
Create summary policy reports based on the CIs in a view or a custom TQL query

1. Create an integration point as described in , if one does not already exist.
2. In UCMDB, create a new view or copy an existing view.

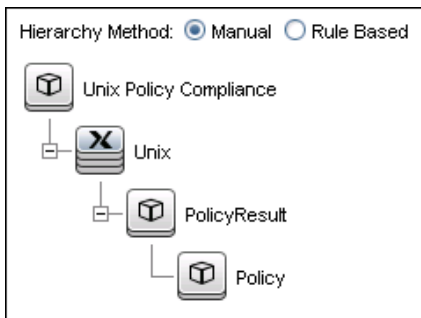
Note: When using a custom TQL query, make sure you take into account the limitations of

the data capacity when using federation. You should filter the CIs in the TQL query to take this limitation into account.

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



- Specify the Configuration Manager integration point that you defined to be the data source that provides the policy and policy result data.
- Set the hierarchy. An example is shown below:



- Create an aggregation function for the Policy CI type. An example is shown below:

Edit Function

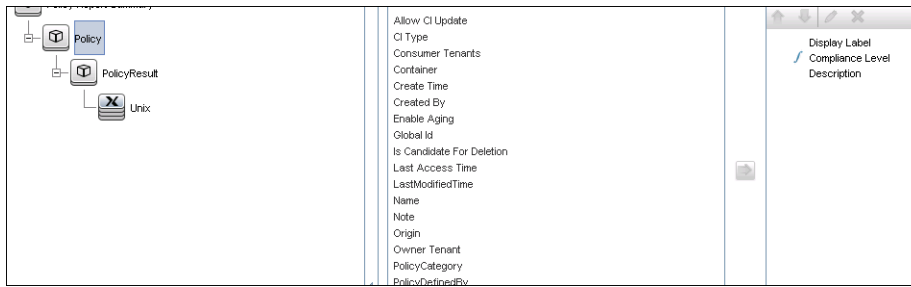
Edit function...

Related Query Node:	Function:	Attribute:
PolicyResult	Average	ComplianceLevel
Unix	Sum	RulesCompliant
	Min	RulesNonCompliant
	Max	
	Count	
	Distinct List	
	Concatenated List	

Title: Compliance Level

OK Cancel

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



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



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



10. If desired, you can schedule these reports to be created periodically. For details, see *Data Flow Management section of the UCMDB Help*.

For details about creating reports, see the section about reports in the *Modeling section of the UCMDB Help*.

Troubleshooting and Limitations – Federating Policy Data

- Federation only works with CIs in the actual state. Therefore:
 - Policy compliance is federated only for CIs in the actual state.
 - The authorization status for CIs that were deleted from the actual state is not shown.
- The maximum number of CIs that can be federated is configurable. For details about changing this number, edit the value of the Max Num To Federate setting in the Infrastructure Settings Manager in

UCMDB. For details about changing settings, see the Infrastructure Settings Manager chapter in the *Administer section of the UCMDB Help*. The recommended number of CIs is no more than 20,000, if large views have been enabled in Configuration Manager. For details about enabling support for large views, see the section describing large capacity planning in the *Universal CMDB Configuration Manager Deployment Guide*.

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

Chapter 5: Micro Focus Discovery and Dependency Mapping Inventory Integration

This chapter includes:

Overview	52
Supported Versions	52
DDMI Adapter	52
How to Populate the CMDB with Data from DDMI	54
How to Federate Data with DDMI	55
How to Customize the Integration Data Model in UCMDB	56
Predefined Queries for Population Jobs	57
DDMI Adapter Configuration Files	58
Troubleshooting and Limitations – DDMI Integration	58

Overview

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

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

Supported Versions

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

DDMI Adapter

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

The DDMI adapter supports the following features:

- Full population of all instances of the selected CI types.
- Identifying changes that have occurred in DDMI, to update them in UCMDB.
- Implementing **Remove** in DDMI. When a CI is removed in DDMI, it is not physically deleted from the database, but its status is changed to indicate that the CI is no longer valid. The DDMI adapter interprets this status as an instruction to remove the CI when needed.
- Federation of defined CI types and attributes.

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

- Node (some of the attributes are populated and some are federated)
- Layer2 connection
- Location that is connected to the node

- IP address
- Interface

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

- Asset
- CPU
- File system
- Installed software
- Printer
- Cost center

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

- **Classes**
 - Person
 - Asset
 - CPU
 - Installed software
 - Printer
 - Windows service
- **Node attributes**
 - DiscoveredOsVendor
 - DiscoveredModel
 - Description
 - DomainName
 - DiscoveredLocation
 - NetBiosName



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

How to Populate the CMDB with Data from DDMI

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

- ["Define the DDMI integration" below](#)
- ["Define a population job \(optional\)" on the next page](#)
- ["Run the population job" on the next page](#)

1. Define the DDMI integration

- a. In UCMDB, navigate to **Data Flow Management > Integration Studio**.
- b. Click the **new integration point**  button to open the new integration point Dialog Box.
 - Click , select the DDMI adapter, and click **OK**.

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

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

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

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



2. Define a population job (optional)

The DDMI adapter comes out-of-the-box with the DDMI Population job, which runs the following predefined queries: **hostDataImport**, **networkDataImport**, **printerDataImport**, and **Layer2DataImport**. For details about these queries, see ["Predefined Queries for Population Jobs" on page 57](#). This job runs according to a default schedule setting.

You can also create additional jobs. To do this, select the Population tab to define a population job that uses the integration point you defined in ["Define the DDMI integration" on the previous page](#). For details, see "New Integration Job/Edit Integration Job Dialog Box" in the *Data Flow Management section of the UCMDB Help*.


3. Run the population job

Activate the population job in one of the following ways:

- To immediately run a full population job, click . In a full population job, all appropriate data is transferred, without taking the last run of the population job into consideration.
- To immediately run a differential population job, click . In a differential population job, the previous population time stamp is sent to DDMI, and DDMI returns changes from that time stamp to the present. These changes are then entered into the UCMDB database.
- To schedule a differential population job to run at a later time or periodically, define a scheduled task. For details, see "Define Tasks that Are Activated on a Periodic Basis" in the *Administer section of the UCMDB Help*.

How to Federate Data with DDMI

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

1. In UCMDB, navigate to **Data Flow Management > Integration Studio**.
2. Select the integration point that you defined in ["Define the DDMI integration" on the previous page](#).
3. Click the **Federation** tab. The panel shows the CI types that are supported by the DDMI adapter.
4. Select the CI types and attributes that you want to federate.
5. Click **Save** .

How to Customize the Integration Data Model in UCMDB

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

To add an attribute to an existing CI type:

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

1. Go to the **orm.xml** file as follows: **Data Flow Management > Adapter Management > DDMiAdapter > Configuration Files > orm.xml**.
2. Locate the **generic_db_adapter.[CI type]** to be changed, and add the new attribute.
3. Ensure that the TQL queries that include this CI Type have the new attribute in their layouts, as follows:
 - a. In the Modeling Studio, right-click the node where you want to include the attribute.
 - b. Select **Query Node Properties**.
 - c. Click **Advanced layout settings** and select the new attribute.

For details about selecting attributes, see "Layout Settings Dialog Box" in the *Modeling section of the UCMDB Help*. For limitations on creating this TQL query, see ["Troubleshooting and Limitations – DDMI Integration" on page 58](#)

To add a new CI type to the DDMI Adapter:

1. In UCMDB, create the CI type that you want to add to the adapter, if it does not already exist. For details, see "Create a CI Type" in the *Modeling section of the UCMDB Help*.
2. Go to the **orm.xml** file as follows: **Data Flow Management > Adapter Management > DDMiAdapter > Configuration Files > orm.xml**.
3. Map the new CI type by adding a new entity called **generic_db_adapter.[CI type]**.
4. In the **orm.xml** file, ensure that the new CI type has the following mappings:
 - a. the **data_note** attribute is mapped to the **NMID_StatusInAppliance** column (this attribute is used for checking the CI's status).

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

For details, see "The orm.xml File" in the *Developer Reference section of the UCMDB Help*.

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

- a. In the Modeling Studio, right-click the node where you want to include the attribute.
- b. Select **Query Node Properties**.
- c. Click **Advanced layout settings** and select the new attribute.

For details about selecting attributes, see "Layout Settings Dialog Box" in the *Modeling section of the UCMDB Help*. For limitations on creating this TQL query, see ["Troubleshooting and Limitations – DDMI Integration" on the next page](#).

6. In UCMDB, navigate to **Data Flow Management > Integration Studio**.
7. Edit the DDMI integration point to support the new CI type by selecting it either for population or for federation.
8. If the new CI type is for population, edit the population job that you created in ["Define a population job \(optional\)" on page 55](#) to include the new TQL query.

Predefined Queries for Population Jobs

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

- **hostDataImport** - use to import nodes. Imported data includes nodes whose NodeRole attribute is either null, or contains **desktop**, **server**, or **virtualized_system**. Nodes are identified either by their interface or IP address. Information also includes the location of the nodes (building, floor and room).
- **networkDataImport** - use to import nodes that are not imported with **hostDataImport**. Similar to **hostDataImport**, except that it imports nodes whose NodeRole is not null and does not contain the following strings: **desktop**, **server**, **virtualized_system**, or **printer**.
- **printerDataImport** - use to import printers. Similar to **networkDataImport**, except that it does import nodes whose NodeRole contains the string **printer**.

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

DDMI Adapter Configuration Files

The adapter includes the following configuration files:

- **orm.xml**. The Object Relational mapping file in which you map between UCMDDB classes and database tables.
- **discriminator.properties**. Maps each supported CI type (also used as a discriminator value in **orm.xml**) to a list of possible corresponding values of the discriminator column, **DeviceCategory_ID**.
- **replication_config.txt**. Contains a comma-separated list of non-root CI and relations types that have a **Remove** status condition in the DDMI database. This status condition indicates that the device has been marked for deletion.
- **fixed_values.txt**. Includes a fixed value for the attribute **ip_domain** in the class IP (**DefaultDomain**).

For details on adapter configuration, see "Developing Generic Database Adapters" in the *Developer Reference section of the UCMDDB Help*.

Troubleshooting and Limitations – DDMI Integration

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

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

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

- The TQL graph must not have cycles.
- A query that is used to synchronize relations should have the cardinality 1...* and an OR condition

between the relations.

- The adapter does not support compound relations.
- The TQL graph should contain only CI types and relations that are supported by the DDML adapter.
- ID conditions on the integration TQL query are not supported.

Chapter 6: Micro Focus IT Executive Scorecard

This chapter includes:

Overview	61
Supported Versions	61
How to Push Data from UCMDB to Executive Scorecard	61
Executive Scorecard Adapter	64

Overview

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

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

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

Supported Versions

UCMDB supports integration with Executive Scorecard version 9.31 or later.

How to Push Data from UCMDB to Executive Scorecard

1. Prerequisite - Set up Executive Scorecard

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


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



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

- a. In UCMDB, go to **Data Flow Management > Adapter Management**.
- b. In the Resources pane, search for **XSAdapter**.

3. Create an integration point

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

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


Field	Required	Description
Credentials ID	Yes	<p>Do the following:</p> <ol style="list-style-type: none"> Click the  button. Select Generic DB Protocol (SQL) and click OK. Click the  button to add new credentials. Enter the following information: <ul style="list-style-type: none"> Database Type: Select Microsoft SQL Server. Port Number: Leave the default value (1433). Connection Timeout: Leave the default value (20000). User Name and Password: Enter the user name and password for connecting to the Executive Scorecard Data Warehouse database. When finished, click OK.
Data Flow Probe	Yes	The name of the Data Flow Probe on which the integration runs. Select a Data Flow Probe whose range includes the IP address of the Executive Scorecard database.
Additional Probes	No	Select additional probes to use when pushing to AM in order to increase redundancy.

For details about how to create an integration point, see "How to Set Up an Integration Point" in the *Data Flow Management section of the UCMDB Help*.


- Click **OK**.
- Click **Test Connection**. If the connection fails, verify that the information provided is correct.
- Click **OK** to close the dialog box and return to the Integration Studio.

4. Edit the existing integration job (optional)

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

- Select the integration point that you created in step 4 and click **XS Push job**.
- Click .
- Update the TQL queries listed or schedule as required.
- When you are finished, click **OK**.

5. Run the job

- a. In the Integration Studio, select the integration point that you created in step 4.
- b. Select **XS Push job** and click  to run a full synchronization.

Executive Scorecard Adapter

This section contains information about the **XSA**adapter.

Input CIT

destination_config

Triggered CI Data

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

Parameters

Name	Value
credentialsId	
database	
host	
port	
schema	dws

Chapter 7: Micro Focus Network Automation (NA) Integration

This chapter includes:

Overview	66
Supported Versions	66
Topology	66
How to Pull Data Topology from a Micro Focus NA Server using a Java Client	67
How to Pull LastSnapshotAttemptDate from NA	71
Pull Topology from Network Automation Adapter	72
Limitations – NA Integration	74

Overview

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

Network engineers can use NA to pinpoint the following:

- Which device configuration changed?
- What exactly was changed in the configuration?
- Who made the change?
- Why the change was made?

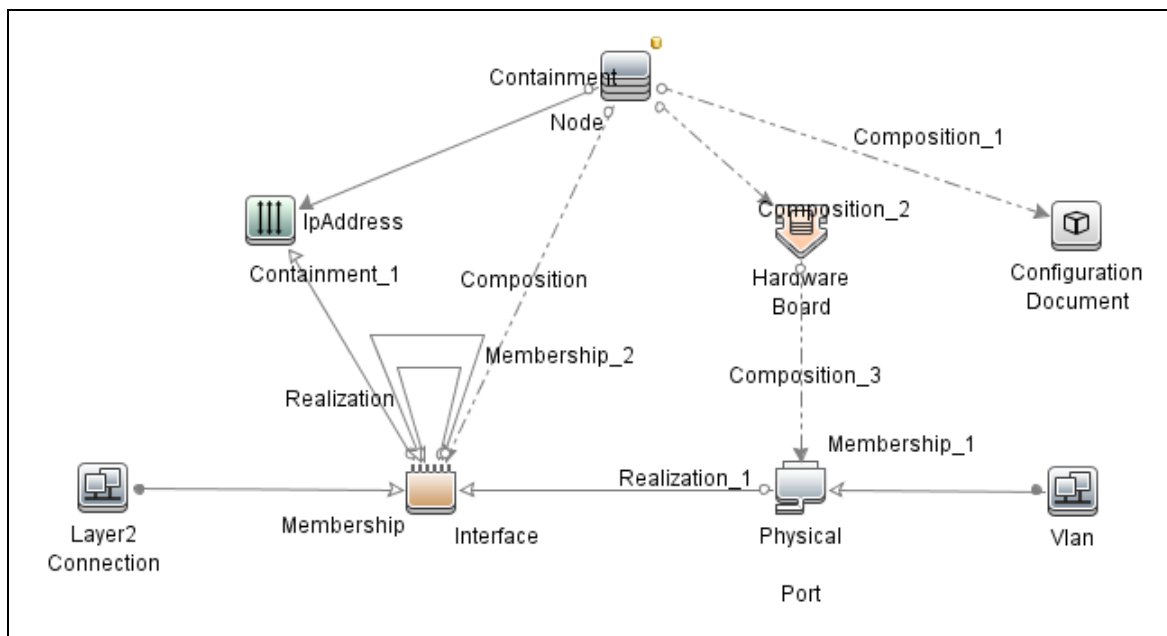
In addition, NA can enforce security and regulatory policies at the network level by making sure that configurations comply with pre-defined standards.

Supported Versions

This integration supports Micro Focus Network Automation (NA) 9.22, 10.00, 10.10, 10.11, 10.20, 10.21, 10.30, 10.40, and 10.50.

Topology

The diagram below depicts the Micro Focus NA topology.



How to Pull Data Topology from a Micro Focus NA Server using a Java Client

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

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

Prerequisites

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

Note: The ports 1098, 1099, and 4446 should be open for Data Flow Probe to communicate with the NA server.

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



Integration Flow

1. (This step is for NA 10.10, 10.11, 10.20, 10.21, and 10.30) To support NA 10.10, 10.11, 10.20, 10.21, and 10.30, do the following:
 - a. Create a new folder **naxxxx** (for example, **na1010**) under the **<DataFlowProbe_Home>\runtime\probeManager\discoveryResources** directory.


Note: **xxxx** is the NA version number.
 - b. Copy the following two JAR files from the **<DataFlowProbe_Home>\runtime\probeManager\discoveryResources\na** directory to the **<DataFlowProbe_Home>\runtime\probeManager\discoveryResources\naxxxx** directory:
 - **ddm-na-client.jar**
 - **fst.jar**
 - c. Copy the following JAR files to the **<DataFlowProbe_Home>\runtime\probeManager\discoveryResources\naxxxx** directory:
 - **bcprov-jdk150n.jar**, **cryptojce.jar**, **cryptojcommon.jar**, and **jcmFIPS.jar** in the **<NA_Installation>\jre\lib\ext** directory.
 - **truecontrol-client.jar** in the **<NA_Installation>\client** directory.
 - **json.jar** in the **<NA_Installation>\server\ext\jboss\server\default\lib** directory.
 - d. Result

The **<DataFlowProbe_Home>\runtime\probeManager\discoveryResources\naxxxx** directory includes the following files:

- **bcprov-jdk150n.jar**
- **cryptojce.jar**
- **cryptojcommon.jar**
- **ddm-na-client.jar**
- **fst.jar**
- **jcmFIPS.jar**
- **json.jar**
- **truecontrol-client.jar**

2. (This step is for NA 10.40 and 10.50) To support NA 10.40 and 10.50, do the following:
 - a. Copy **bc-fips.jar** and **bcpkix-fips.jar** from **<NA_Installation>\jre\lib\ext** to **<DataFlowProbe_Home>\runtime\probeManager\discoveryResources\NA1040**.
 - b. Copy **truecontrol-client.jar** from **<NA_Installation>\client** to **<DataFlowProbe_Home>\runtime\probeManager\discoveryResources\NA1040**.
 - c. Copy **ddm-na-client.jar** and **json.jar** from **<DataFlowProbe_Home>\runtime\probeManager\discoveryResources\na** to **<DataFlowProbe_Home>\runtime\probeManager\discoveryResources\NA1040**.
3. Create a new integration point in the Integration Studio based on the **Pull Topology from Network Automation** adapter as follows:
 - a. In UCMDB, go to **Data Flow Management > Integration Studio**.
 - b. Click the **New Integration Point**  button. The **New Integration Point** dialog box opens.
 - c. Enter a name and description for the adapter.
 - d. Check the **Is Integration Activated** check box.
 - e. In the Adapter field, click the **Select Adapter**  button. The Select Adapter dialog box opens.
 - f. Go to **HP Software Products / Micro Focus Products > Network Automation**, select **Pull Topology from Network Automation**, and click **OK**.
 - g. In the Adapter Properties section, define the following properties:

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

Parameter	Description
Credentials ID	Click the Select Credentials ID  button and in the Choose Credentials dialog box, select the HP Network Automation Java Protocol to be used for connection.
discoverDeviceGroup	This parameter indicates whether to discover Device Group CIs. The default value is true .
NAVersion10.40	This parameter is for NA 10.40 and 10.50. Set its value to true if you create the integration for NA 10.40 and 10.50.


Parameter	Description
Query Topology Per Device	This parameter affects how information about devices is queried. If false, topology entities are retrieved in single queries. If true, separate query is issued to retrieve data per device. The default value is false .
Remote JVM Arguments	<p>The JVM parameters that should be passed to the remote process.</p> <p>For NA 10.40 and 10.50 integration, if the NA server runs on the Linux operation system, you need to add the following:</p> <p>-Djava.security.egd=file:/dev/urandom</p>
Remote JVM Class Path	<p>The class path of the external JVM process.</p> <ul style="list-style-type: none"> When integrating with NA 10.10, 10.11, 10.20, 10.21, or 10.30, replace the value of the remoteJVMClasspath with the following: <code>../runtime/probeManager/discoveryResources/naxxxx/*;%minimal_classpath%</code> When integrating with NA 10.40 or 10.50, do the following: <ul style="list-style-type: none"> A. Remove all the default value. B. Add the following: <code>../runtime/probeManager/discoveryResources/NA1040/*;</code> C. Open the file <code><DataFlowProbe_Home>\conf\DataFlowProbeOverride.properties</code>, copy the value of the basic_discovery_minimal_classpath parameter to a temporary location, and remove the following: <ul style="list-style-type: none"> <code>../lib/bcprov-jdk15.jar;</code> <code>../lib/lwcm.jar;../lib/lwcrypto.jar;../lib/lwssolmpl.jar;../lib/lwssouutils.jar;</code> <p>Note: If the DataFlowProbeOverride.properties file does not exist in your environment, use the DataFlowProbe.properties file instead.</p> D. Add the rest of the value of basic_discovery_minimal_classpath that you copied to this parameter Remote JVM Class Path.
Report Device Configuration	This parameter specifies whether the latest configuration of devices should be reported as configuration_document CIs. The default value is false .
Run In Separate Process	This parameter enables the job to run in an external JVM process, separate from the main probe process. The default value is true and this value should not be changed.

Parameter	Description
Data Flow Probe	This parameter defines the Probe to be used for integration. Select DataFlowProbe to explicitly choose the probe, or Auto-Select if you want the probe to be automatically selected according to where the destination IP is assigned.
Trigger CI instance	This parameter defines the IP Address of the target NA server.

- h. Click **OK** in the New Integration Point dialog box.
4. Run the full synchronization of the **Network Automation by Java** job with the integration point that you just created. This job performs the following internal actions:
 - a. Establishes a connection to the target NA server.
 - b. Performs a query to retrieve IDs of all devices in the system.
 - c. Retrieves all devices in chunks. The default chunk size is 500 devices.
 - d. Retrieves all ports on all devices. Depending on the value of parameter **Query Topology Per Device**, the ports are retrieved per device or from all devices at once.
 - e. Retrieves all device modules. Depending on the value of parameter **Query Topology Per Device**, the device modules are retrieved per device or from all devices at once.
 - f. Retrieves all VLANs.
 - g. Retrieves connections between devices. Depending on the value of parameter **Query Topology Per Device**, the connections are retrieved per device or from all devices at once.
 - h. Retrieves configuration of each device as textual data (if **Report Device Configuration=true**).
 - i. Analyzes topology. Port channels are identified with related child ports, port aliases are identified along with the parent port.
 - j. Reports the topology. The job forms result vectors with information about devices and their connections, while trying to not exceed 10000 objects in one vector. Vectors are sent to the UCMDB server.

How to Pull LastSnapshotAttemptDate from NA

In NA, the **LastSnapshotAttemptDate** attribute in the **RN_DEVICE** table shows the date and time that the data was last updated. If you want to pull this attribute from NA to UCMDB, do the following:

1. Go to **Modeling > CI Type Manager > CI Types** pane, and then search for **Node**.
2. In the right pane, click the **Attributes** tab, and then click the **Add**  button.
3. In the Add Attribute dialog box, fill the fields in the **Details** tab as follows:
 - **Attribute Name:** Type **Last_discovered_Date_NA**.
 - **Attribute Name:** Type **LastSnapshotAttemptDate**.
 - **Attribute Type:** Click **Primitive**, and then select **data number** in the list.
4. Click **OK**.
5. Go to **Data Flow Management > Adapter Management > Resources** pane, search for **network_automation.py**.
6. In the right pane, locate the following lines, and then delete the ' ' character to uncomment the code:

```
'''  
if device.lastSnapshotAttemptDate:  
    deviceOsh.setAttribute('Last_discovered_Date_NA',  
device.lastSnapshotAttemptDate)  
'''
```

7. Run the **Pull Topology from Network Automation** job.

Pull Topology from Network Automation Adapter

This section contains information on the **Pull Topology from Network Automation** adapter.

Input CIT

IpAddress

Triggered CI Data

Name	Value
ip_address	\${SOURCE.name}

Used Scripts

- network_automation.py
- na_discover.py
- na_integration_by_java.py

Discovered CITs

- Composition
- ConfigurationDocument
- Containment
- Fan
- HardwareBoard
- Interface
- IpAddress
- Layer2Connection
- Membership
- Network Automation Device Group
- Node
- PhysicalPort
- PowerSupply
- Realization
- Vlan

Adapter Parameters

Parameter	Default Value	Description
credentialsId		The Credential used for the specific integration.
queryTopologyPerDevice	false	Indicates whether to perform separate queries per device (true), or retrieve topology

Parameter	Default Value	Description
		with queries for all devices at once (false, default).
remoteJVMArgs	-Xms1024m -Xmx4096m	The JVM parameters that should be passed to the remote process.
remoteJVMClasspath	../runtime/probeManager/discoveryResources/na/*;%minimal_classpath%	The external JVM classpath.
reportDeviceConfigs	false	Indicates whether to report configuration as config files for devices.
reportDeviceModules	false	Indicates whether to report modules (Fans, Power Supplies, and other modules) for devices.
runInSeparateProcess	true	Indicates whether to run the adapter in the external JVM.

Limitations – NA Integration

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

Chapter 8: Micro Focus Network Node Manager (NNMi) Integration

This chapter includes:

Overview	76
Use Cases	76
Supported Versions	76
NNMi - UCMDB Integration Architecture	77
Topology	78
Pull Topology from NNMi Adapter	78
Pull Topology from NNMi by REST API Adapter	79
How to Run NNMi–UCMDB Integration	79
How to Manually Add the IpAddress CI of the NNMi Server	82
How to Set Up NNMi–UCMDB Integration	83
Pull Topology from NNMi Adapter	84
Pull Topology from NNMi by REST API Adapter	89
NNMi Update IDs Adapter	91
How to Customize Integration	92
Troubleshooting and Limitations – NNMi Integration	96

Overview

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

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

Use Cases

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

- **Use Case 1:** A UCMDB user wants to view the Layer 2 network topology supporting servers and applications. The requirement is to use NNMi as the authoritative source for that information with access through the UCMDB application.
- **Use Case 2:** An NNMi operator wants to view the impact of a network access switch infrastructure failure where the impact data is available in UCMDB. The NNMi operator selects an incident or a node in NNMi and then enters a request for impacted CIs.
- **Use Case 3:** A UCMDB user wants to view the Layer 2 connection between switches and servers when NNMi is used to discover switches and Layer2 network topology, and UD discovers servers.

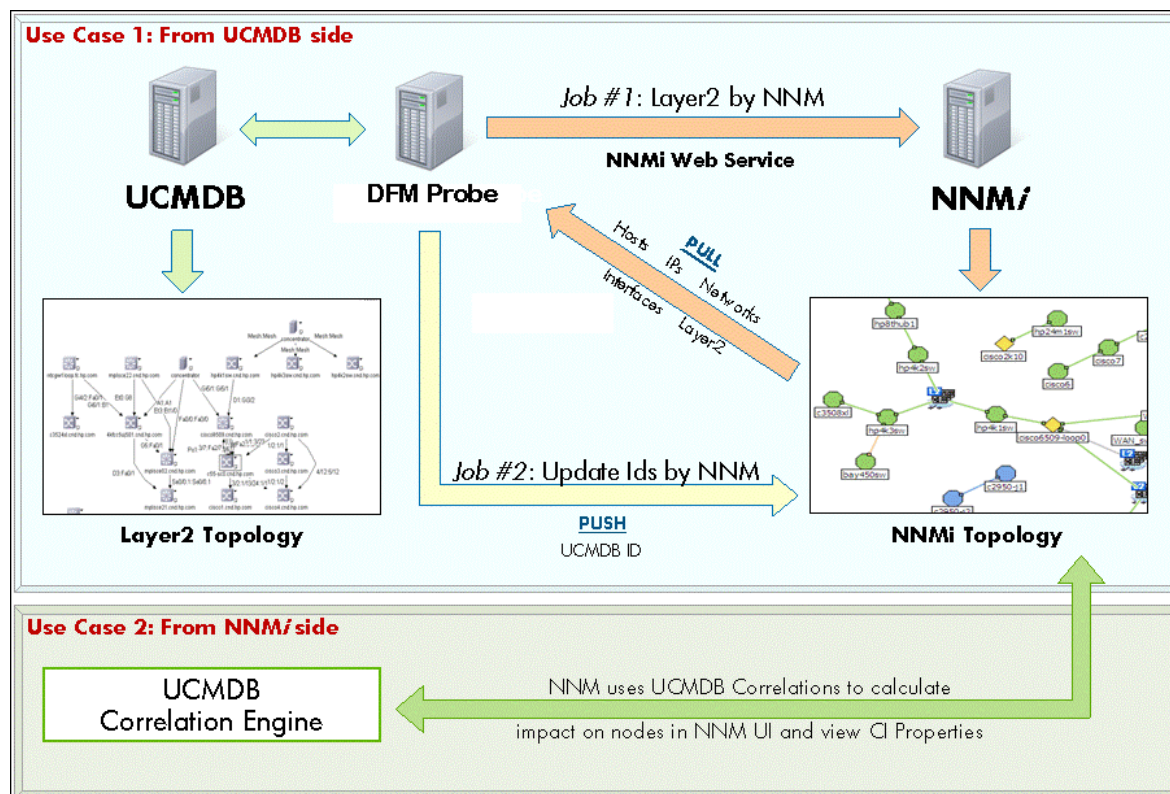
Supported Versions

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

- Data Flow Probe version 9.02 or later
- Micro Focus NNMi versions 8.1, 8.11, 9.x, 10.0x, 10.10, 10.20, 10.21, 10.30, 10.40, and 10.50.

Note: The **Pull Topology from NNMi by REST API** adapter requires NNMi 2018.05.

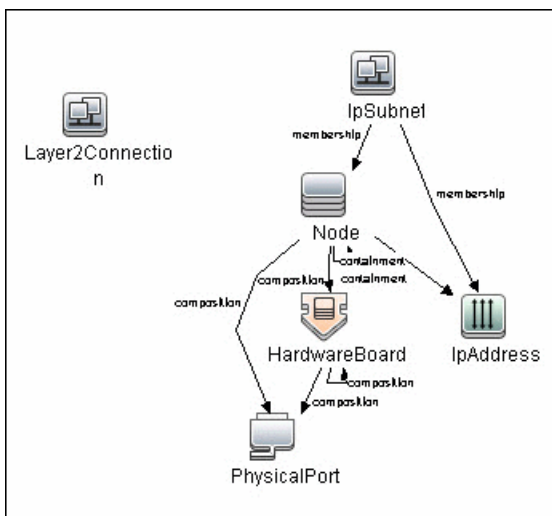
NNMi - UCMDB Integration Architecture



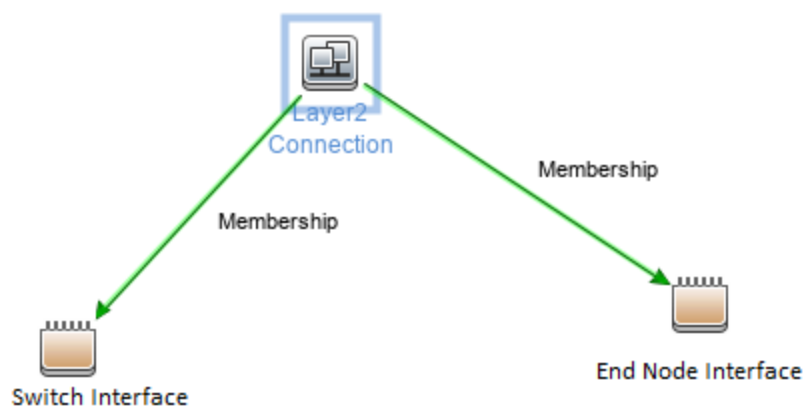
Topology

Pull Topology from NNMi Adapter

Note: For a list of discovered CITs, see ["Discovered CITs" on page 84](#).



Pull Topology from NNMi by REST API Adapter



How to Run NNMi–UCMDB Integration

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

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

This task includes the following steps:

1. Run NNMi Integration

In the Integration Studio, create a new integration point.

- a. Provide a name and description for the integration point.
- b. Under **Integration Properties > Adapter**, select the appropriate adapter:

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

This job connects to the NNMi Web service and retrieves NNMi discovered nodes, IP addresses, networks, interfaces, physical ports, VLANs, hardware boards, Layer 2 connection information to create a Layer 2 topology in UCMDB.

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

This job is needed when you can only discover switches and Layer2 connection to servers in NNMi, and the servers are discovered in Universal Discovery. This adapter can populate Layer2 connection from NNMi to Universal Discovery.

To use this job, follow these steps:

- A. Run the **Pull Topology from NNMi** job to populate switches from NNMi to Universal Discovery.
 - B. Run the **Pull Topology from NNMi by REST API** job to query server and switch interfaces from Universal Discovery and then pass them to NNMi, and query Layer2 connection and then send it back to Universal Discovery.
- **NNMi Update IDs:** Use this adapter to run push jobs. This job:
 - Retrieves the UCMDB IDs of the NNMi hosts and NNMi interfaces from the UCMDB Server using the UCMDB Web Services API.
 - Updates the **ODB_ID** and **UCMDB_ID** custom attributes on the corresponding Node and Interface CIs on the NNMi server using the NNMi Web service.
- c. Under **Adapter Properties > Data Flow Probe**, select the **Data Flow Probe**.
 - d. Under **Adapter Properties > Trigger CI instance** select:
 - **Select Existing CI** (if you have a valid, existing CI). The **Select Existing CI** pane appears. Select the CI; or
 - **Create New CI** (if you need to create a new CI). The **Topology CI Creation Wizard** appears. Complete the creation of the CI using the Wizard.
- Note:** For details on the Topology CI Creation Wizard, see "Topology CI Creation Wizard" in the *Data Flow Management section of the UCMDB Help*.
- e. Under **Adapter Properties > Credentials ID** select the appropriate credentials for connection to the NNMi server.
 - f. Save the Integration Point.
 - g. Run the job.

Note: For details on running an integration job, see "Integration Studio" in the *Data Flow Management* section of the *UCMDB Help*.

2. Validate results

Verify that data was discovered using the NNMi integration jobs.

- a. For the **Pull Topology from NNMi** and **Pull Topology from NNMi by REST API** jobs:
 - In UCMDB, navigate to **Managers > Modeling > IT Universe Manager**.
 - In the CI Selector pane, select **Browse Views**.
 - On the **View** drop-down menu, select **Layer 2 by NNMi**. Select a view. The view displays the CIs and relationships discovered by the integration job.
- b. For the **NNMi Update IDs** job:
 - In NNMi, open an NNMi Node or Interface that was discovered in UCMDB.
 - On the **Custom Attributes** tab, look for the **ODB_ID** and **UCMDB_ID** custom attributes. These attributes contain the UCMDB ID of the corresponding host or interface in UCMDB.


How to Manually Add the IpAddress CI of the NNMi Server

Note: When you installed Universal CMDB, you may have installed a bundled UCMDB that uses a Foundation license. If your UCMDB installation has a Foundation license deployed, use the steps in this section to manually add an **IpAddress** CI. If any other license (Basic or Advanced) is deployed on the UCMDB server, discover the IPAddress CI as described in ["How to Run NNMi–UCMDB Integration" on page 79](#).

To manually add the IpAddress CI of the NNMi server

1. Verify that the Data Flow Probe is correctly installed and connected to the UCMDB Server.
2. Add the IP of the NNMi server to the Data Flow Probe range:

In the **Data Flow Probe Setup** module, select the Probe that is to be used for the NNMi integration, and add the IP address of the NNMi server to its range. For details, see the section describing how to add Probe range in the Data Flow Management section of the UCMDB Help.

3. Insert the **Address** CI of the NNMi server in the CMDB:
 - a. In **Modeling > IT Universe Manager**, in the CI Selector pane, click the **Browse Views** tab and select **Network Topology** from the **View** drop-down menu.
 - b. Click the **New CI**  button.
 - c. In the New CI dialog box, select the **IpAddress** CIT from the tree and enter the following values:

Field	Description
IP Address	The IP address of the NNMi server.
IP Domain Name	The UCMDB domain name (for example, DefaultDomain).
IP Probe Name	The name of the Data Flow Probe (for example, DefaultProbe).

- d. Save the **IpAddress** CI.

How to Set Up NNMi–UCMDB Integration

The following steps describe how to configure NNMi to communicate with UCMDB:

Configure the connection between NNMi and UCMDB

On the NNMi management server, do the following:

1. In the NNMi console, open the **NNMi–UCMDB Integration Configuration** form (**Integration Module Configuration > UCMDB**).
2. Select the **Enable Integration** check box to activate the remaining fields on the form.
3. Enter the information for connecting to the NNMi management server.
4. Enter the information for connecting to the UCMDB server.
5. Click **Submit** at the bottom of the form.

A new window displays a status message. If the message indicates a problem with connecting to the UCMDB server, re-open the **NNMi–UCMDB Integration Configuration** form (or press **ALT+LEFT ARROW** in the message window), and then adjust the values for connecting to the UCMDB server as suggested by the text of the error message.

Customize the integration

On the NNMi management server, do the following:

1. In the NNMi console, open the **NNMi–UCMDB Integration Configuration** form (**Integration Module Configuration > UCMDB**).
2. Enter values for the following fields:
 - HP UCMDB Correlation Rule Prefix
 - HP UCMDB Impact Severity Level (1–9)
3. Click **Submit** at the bottom of the form.

Pull Topology from NNMi Adapter

This section contains information on the **Pull Topology from NNMi** adapter.

Input CIT

IpAddress

Triggered CI Data

Name	Value
ip_address	\${SOURCE.name}

Used Scripts

- nnmi_netutils.py
- nnmi_filters.py
- nnmi_api.py
- nnmi.py
- NNM_Integration.py

Discovered CITs

- Composition
- Containment
- HardwareBoard
- Interface
- IpAddress
- IpSubnet
- Layer2Connection
- Membership

- Node
- PhysicalPort
- Realization
- Vlan

Note: To view the topology, see ["Topology" on page 78](#).

Adapter Parameters

Parameter	Description
NNMiHost	The host name to be used with the integration. You must fill in this parameter if FIPS is enabled. NNMiHost has higher priority over IP address, which means instead of IP address, NNMiHost will be used if it is provided.
credentialsId	The credential used for the integration.
discoverDisabledIps	<p>Defines whether the integration should discover disabled IP addresses.</p> <p>When this parameter is set to false, the integration does not discover disabled IP addresses.</p> <p>Default: false.</p>
discoverLayer2	<p>Defines whether the integration should discover the Layer2Connection CIs from NNMi.</p> <p>When this parameter is set to true, the integration fetches all the Layer2Connections-related data, iteratively querying for a specified number of Layer2Connections from NNMi (based on value of the pageSizeLayer2 parameter), then querying for Network Interfaces on the ends of Layer2Connection and Nodes hosting these interfaces with instant push of collected topology to UCMDB.</p> <p>Default: true</p>

Parameter	Description
discoverNodes	<p>Defines whether the integration should discover all the Nodes that are registered in NNMi, regardless of their inclusion into Layer2 Topology or VLANs.</p> <p>When this parameter is set to true, integration fetches all the Nodes with connected IpAddresses, Interfaces, HardwareBoards, Physical Ports and IpSubnets, iteratively querying for a specified number of Nodes with related data from NNMi (based on value of the pageSizeNodes parameter) and instantly pushing collected topology into UCMDB.</p> <p>Default: true</p>
discoverNonManagedInterfaces	<p>Defines whether the integration should discover non-managed interfaces.</p> <p>When this parameter is set to false, the integration does not discover non-managed interfaces.</p> <p>Default: false.</p>
discoverNonManagedNodes	<p>Defines whether the integration should discover non-managed nodes.</p> <p>When this parameter is set to false, the integration does not discover non-managed nodes.</p> <p>Default: false.</p> <p>When this parameter is set to true, discoverDisabledIps and discoverNonManagedInterfaces are ignored for non-managed nodes; the integration will discover everything, including disabled ips and non-managed interfaces.</p>
discoverPhysicalPorts	<p>Defines whether the integration should discover physical ports.</p> <p>When this parameter is set to false, the integration does not discover physical ports.</p> <p>Default: false.</p> <p>When this parameter is set to false, the integration does not discover VLANs and HardwareBoards.</p>

Parameter	Description
discoverVlans	<p>Defines whether the integration should discover all the VLANs that are registered in NNMi.</p> <p>When this parameter is set to true, integration fetches all the VLANs with member Physical Ports, Hardware Boards and Nodes hosting those Physical Ports and Node-related topology, iteratively querying for a specified number of VLANs (based on the value of <code>pageSizeVlans</code> parameter), getting all the necessary related topology and instantly reporting it back to UCMDB.</p> <p>Default: false</p> <p>This parameter is ignored if discoverPhysicalPorts is set to false.</p>

Parameter	Description
discoveryMode	<p>Specifies the adapter's mode of execution, either related or full.</p> <p>Default: related</p> <ul style="list-style-type: none"> In related read mode, only a small subset of the topology is read, starting from the root entity. For example, the following is possible: <ul style="list-style-type: none"> read several Nodes read all entities related to these Nodes, including Interfaces, Ip Addresses, Ports, and so on. send data to UCMDB continue with other Nodes In full read mode, the entire topology is read into memory before the data is processed and result vectors are formed. This approach requires more memory, but reduces the number of requests to the NNMi server. The job sends the topology in chunks after more than 10,000 objects are accumulated for sending. The Data Flow Probe then processes these chunks and sends them to the UCMDB server. <p>Use the remoteJVMArgs parameter of the integration adapter to increase the memory limits for the remote Java virtual machine that executes the job, in case there is a lot of data in NNMi. For example, setting the value of the remoteJVMArgs parameter to Xmx4096m enables the remote JVM to consume up to 4 GB of memory. As a general rule, reading 2 million objects from NNMi consumes approximately 10-12 GB of memory.</p> <p>For full read mode, it is recommended to set all page sizes to 1000 (using the pageSizeLayer2, pageSizeNodes, and pageSizeVlans parameters).</p>
pageSizeLayer2	<p>Defines the number of Layer2Connection CIs to fetch from NNMi per one query.</p> <p>Default: 100</p>
pageSizeNodes	<p>Defines the number of Nodes to fetch from NNMi per one query.</p> <p>Default: 100</p>

Parameter	Description
pageSizeVlans	Defines the number of VLANs to be queries from NNMi per one query. Default: 1
remoteJVMArgs	The JVM parameters that should be passed to the remote process.
remoteJVMClasspath	The external JVM classpath.
requestCustomAttributes	Enables global querying for custom attributes of entities in the script. The script must still be modified to read the values and report them (this code modification is expected to be made by the customer). Without modification to the script, setting this parameter to true has no effect. For performance reason, by default, custom attributes are not read from the NNMi server . Default: false
runInSeparateProcess	Defines whether to run the adapter in the external JVM. Default: true

Pull Topology from NNMi by REST API Adapter

This section contains information on the **Pull Topology from NNMi by REST API** adapter.

Input CIT

IpAddress

Triggered CI Data

Name	Value
interface_cmdbid	\${Interface.global_id}
interface_nnm_uid	\${Interface.interface_nnm_internal_key:NA}
ip_address	\${SOURCE.name}
mac_address	\${Interface.mac_address:NA}

Used Scripts

nnmi_restapi.py

Discovered CITs

Layer2Connection

Note: To view the topology, see [Topology](#).

Adapter Parameters

Parameter	Description
NNMi_REST_APIport	The port number for NNMi. Default: 443
credentialsId	The credential used for the integration.

NNMI Update IDs Adapter

This section contains information on the **NNMI Update IDs** adapter.

Input CIT

IpAddress

Triggered CI Data

Name	Value
ip_address	\${SOURCE.name}

Used Scripts

- nnmi_filters.py
- NNM_Update_Ids.py

Adapter Parameters

Parameter	Description
NNMiHost	<p>The host name to be used with the integration. NNMiHost has higher priority over IP address, which means instead of IP address, NNMiHost will be used if it is provided.</p> <p>If using NNMiHost instead of IP address in the population from NNMi integration, you must configure the same NNMiHost here as well for the update.</p>
credentialsId	The credential used for the integration.
remoteJVMClasspath	The external JVM classpath.
runInSeparateProcess	<p>Defines whether to run the adapter in the external JVM.</p> <p>Default: true</p>

How to Customize Integration

This section describes how to customize the NNM Integration package in order to report additional attributes or entities, or to perform other changes.

Included Scripts

The NNM Integration package includes the scripts detailed in the following table:

Name	Description
nnmi_api.py	Implements API for accessing and retrieving data in NNM. Includes definition of base entities, such as Node or Interface , collections, topology and fetcher classes.
nnmi_filters.py	Support module used by nnmi_api.py . Includes definition of filters, allowing creation of advanced expressions when querying data in NNM. In general, you should not modify this script.
nnmi.py	Uses API provided by nnm_api.py to retrieve data, form topology into ObjectStateHolder objects and send result to UCMDB.
nnmi_restapi.py	Uses REST API to discover Layer2Connection CIs.
NNM_Integration.py	Main entry point of the integration. Contains DiscoveryMain method which is called by probe, and uses nnmi.py .
NNM_Integration_Utils.py	Previous version of NNM integration implementation, used by CheckCred.py script which validates credentials.
NNM_Integration_Utils_9.py	Previous version of NNM integration implementation for UCMDB 9.0
NNM_Update_Ids.py	Support module which updates custom attribute in NNM with UCMDB ID of the CI.

Customization Step by Step

The following steps illustrate customization of integration with an example, showing how you discover and report an additional attribute for an Interface. In this example, the attribute is the **managementMode** property in the NNM interface.

1. Prerequisites

- a. You need to know the **name** of the attribute in NNM. Here it is **managementMode**.
- b. You need to know the **name of a method** which should be called on the stub to read the corresponding property of entity, in this case Interface. This information is in the corresponding WSDL file or API documentation.

2. Property Retrieval

For the new property to be reported, it should be retrieved first. To do that, you must modify the **nnmi_api.py** script.

- a. Locate Entity class

Open **nnm_api.py** and find the corresponding Entity class. Entity class names follow the pattern **Nms*Entity**. For this example, you want **NmsInterfaceEntity**.

```
nnmi_api.py

class NmsInterfaceEntity(BaseNmsEntity):
    ...
```

- b. Add new entry to **field_names** tuple

Add a new field to the **field_names** tuple. This tuple is used for real-time entity introspection, mainly debugging.

Note: For use in Python classes, translate a camel cased field name into an underscore separated field name. For example: **management_mode**.

```
nnmi_api.py

field_names = (
    'id',
    'name',
    'hosted_on_id',
    'connection_id',
    'if_index',
    'if_alias',
    'if_descr',
    'if_name',
    'if_speed',
    'physical_address',
    'if_type',
    'uuid',
    'management_mode',
)
```

- c. Modify constructor of the Entity to read the new property

Modify **__init__** method of the entity to read the new property. Modification depends on whether you need property post-processing.

i. Simple read

Call the method on the stub.

```
nnmi_api.py

class NmsInterfaceEntity(BaseNmsEntity):
    ...
    def __init__(self, item, fetcher):
        ...
        self.if_type = item.getIfType()
        self.uuid = item.getUuid()

        self.management_mode = item.getManagementMode()
```

ii. Read with post processing

Delegate reading of the property to a new method which performs post processing for the retrieved value. In this case the new method is added to an **NmsInterfaceEntity** class called **_get_management_mode** which just strips the value of white spaces and makes it lowercase.

```
nnmi_api.py

class NmsInterfaceEntity(BaseNmsEntity):
    ...
    def __init__(self, item, fetcher):
        ...
        self.if_type = item.getIfType()
        self.uuid = item.getUuid()

        self.management_mode = self._get_management_mode(item)
        ...

    def _get_management_mode(self, item):
        management_mode = item.getManagementMode()

        if management_mode:
            return management_mode.strip().lower()
        else:
            return ''
```

3. Property reporting

Once the modifications to **nnmi_api.py** are done, the new property is available, but it is not reported yet. To add reporting for this property, you need to modify the **nnmi.py** script.

a. Locate **Builder** class

Open the **nnmi.py** script and find the corresponding **Builder** class for the entity being reported. All **Builder** names match the ***Builder** pattern. In your case, you need the **InterfaceBuilder** class.

```
nnmi.py
class InterfaceBuilder:
    ...
```

b. Locate a place where **ObjectStateHolder** is created

Each **Builder** has a build method and several other methods which assist in the creation of the **ObjectStateHolder**. Find where this **ObjectStateHolder** is created either directly or indirectly by calling the method in the **modeling.py** module. In your example, **ObjectStateHolder** is created in the method **_createInterfaceOsh**.

```
nnmi.py
class InterfaceBuilder:
    ...
    def _createInterfaceOsh(self, interface):
        interfaceOsh = modeling.createInterfaceOSH( ...

        self._setIsPseudoAttribute(interface, interfaceOsh)

        return interfaceOsh
```

c. Add new attribute reporting

Now you add reporting of the new property. Here, you report the new value to the attribute **interface_management_mode** of the **Interface** CIT which you previously created.

```
nnmi.py
class InterfaceBuilder:
    ...
    def _createInterfaceOsh(self, interface):
        interfaceOsh = modeling.createInterfaceOSH( ...

        self._setIsPseudoAttribute(interface, interfaceOsh)

        interfaceOsh.setAttribute('interface_management_mode',
interface.management_mode)

        return interfaceOsh
```

Troubleshooting and Limitations – NNMi Integration

This section describes troubleshooting and limitations for NNMi Integration.

- **Problem:** The NNMi Web service responds with a **cannot interrogate model** message.
Solution: This message usually indicates that the Web services request made to the NNMi server is incorrect or too complex to process. Check the NNMi jbossServer.log file for details.
- **Problem:** If an excessive number of nodes are to be updated with the same UCMDB ID, it may take a while for the update adapter to complete.
Solution: The volume of data retrieved from the NNMi server might be large. The recommended memory requirements for the Data Probe process is 1024 MB. Since the NNMi Web service enables updating the individual nodes one at a time, the time to update the nodes may take a while.
- **Problem:** The **Layer 2 by NNM** job finishes with the following warning: Failed to get any Layer 2 links from NNM.
Solution: Refer to technical article [KM629927](#).
- **Problem:** Either of the NNMi integration jobs fails with the following error in the DFM log files:
`com.hp.ov.nms.sdk.node.NmsNodeFault: Cannot interrogate model.`
Solution: This error typically means that the NNMi server failed to process the Web services call. Check the following two logs on the NNMi server for exceptions when the integration was activated:
 - jbossServer.log
 - sdk.0.0.log
- **Problem:** Either of the NNMi integration jobs fail with the following error: Could not find Discovery Probe 'DefaultProbe'. Task for TriggerCI will not be created.
Solution:
 - a. Right-click the job and select **Go To Adapter**.
 - b. Click the **Adapter Management** tab.
 - c. Select the **Override default Probe selection** check box, and enter the name of the Probe used for the NNMi integration in the **Probe** field.

- d. Click **Save** to save the adapter, then reactivate the job against the **IpAddress** CI of the NNMi server.

- **Problem:** NNMi integration fails when using integration probe.

Solution: In case of using integration service, for the NNM ID Update integration, the JDBC driver should be specified in the classpath as well.

If your UCMDB server uses Oracle, you need to add **../lib/mcoracle.jar** to the **remoteJVMClasspath** field as Adapter Properties when configuring the integration point; If your UCMDB server uses MS SQL, add **../lib/mcsqserver.jar** to the **remoteJVMClasspath** field.

- **Problem:** NNMi integration job fails with the `EOFException` error when using the integration service or using the Data Flow Probe that is upgraded from a previous version.

Solution:

- a. When using the integration service, locate and open the **<UCMDB_Server_Home>\integrations\conf\DataFlowProbe.properties** file using a text editor.

When using an upgraded Data Flow Probe, locate and open the **<DataFlowProbe_Home>\conf\DataFlowProbe.properties** file using a text editor.

For details about this file, see "DataFlowProbe.properties File" in the *Data Flow Management section of the UCMDB Help*.

- b. Search for the **basic_discovery_minimal_classpath** parameter.
- c. Change the following two values of this parameter as illustrated below:

From	To
../lib/commons-dbc.jar	../lib/commons-dbc2.jar
../lib/commons-pool.jar	../lib/commons-pool2.jar

- d. Save the changes.
- e. If the original Data Flow Probe version is 10.31 or later,
 - When using the integration service, open the **<UCMDB_Server_Home>\integrations\conf\DataFlowProbeOverride.properties** file and follow steps b through d.
 - When using the Data Flow Probe, open the **<DataFlowProbe_Home>\conf\DataFlowProbeOverride.properties** file and follow steps b through d.

For details about this file, see "DataFlowProbeOverride.properties File" in the *Data Flow*

Management section of the UCMDB Help.

f. Restart the integration service or Data Flow Probe service.

- **Problem:** Population from NNMi integration returns success status when NNMi is offline.

Solution: Actually this is the designed behavior. In some cases, the NNMi job may process for nearly 10 hours, if NNMi disconnects after 9 hours of processing, you may want to save the result and would not like to re-process the job.

If you want to delete the result instead, you can change a parameter in the **nnmi_api.py** file. To do so,

- a. In UCMDB UI, go to **Data Flow Management > Adapter Management**.
- b. Search for the **nnmi_api.py** file and open it using a text editor.
- c. Locate line 1150 and find **rethrow_exception=False**.
- d. Change the value from **False** to **True**.
- e. Save the file.

Limitation

When integrating with multiple NNMi servers, each node (host) may only be managed by one NNMi server at the same time.

Chapter 9: Micro Focus SMA-X Push Integration

This section includes:

Overview	100
How Data is Synchronized Between UCMDB and SMA-X	100
Supported Versions	101
How to Integrate UCMDB and SMA-X	101
Set up UCMDB	101
Push CI Data from UCMDB to SMA-X	104
Schedule Data Push Jobs	107
Tailor the Integration	108
Integration Architecture	108
Change Adapter Settings	111
Customize an Existing Mapping	112
Add a New Mapping to the Integration	113
Troubleshooting and Limitations – SMA-X Push Integration	116

Overview

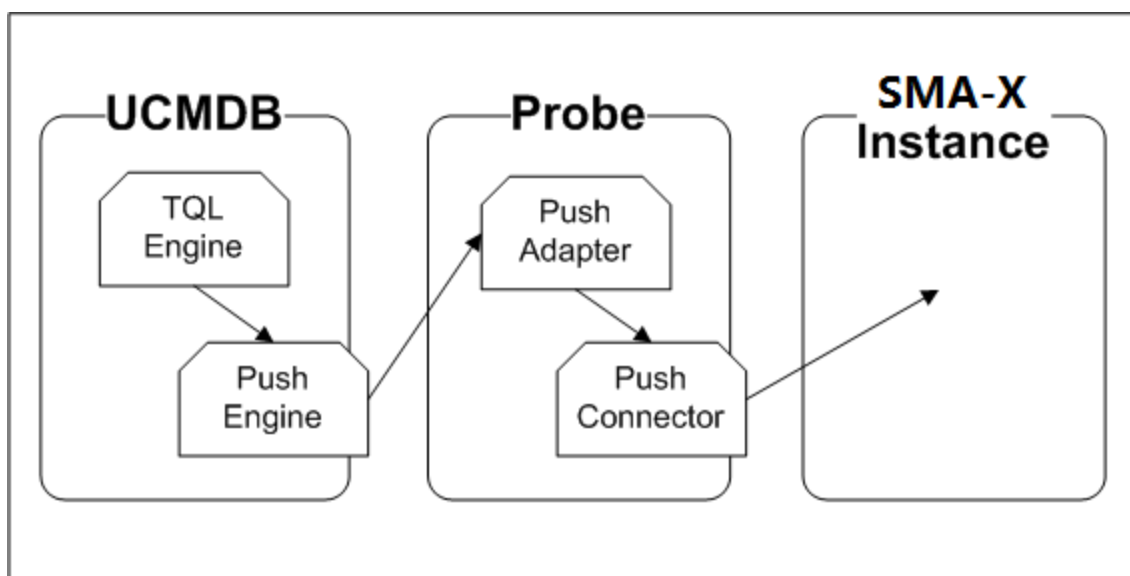
Note: **Service Anywhere Push Adapter** is renamed to **SMA-X Push Adapter**.

Integration between Universal CMDB (UCMDB) and Micro Focus Service Management Automation X (SMA-X) suite enables you to share information from UCMDB with SMA-X. Common use cases include Hardware, Installed Software, Running Software, and Business Services.

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

How Data is Synchronized Between UCMDB and SMA-X

The following graphic shows the high-level components of the integration:



Note: The Push Adapter and Push Connector are executed in the Data Flow Probe/Integration Service process.

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

The **UCMDB Push Engine**:

- Retrieves the required data from UCMDB, using the given TQL query.
- If set for differential synchronization, filters the data to include only the data that has changed since the last execution of this synchronization.
- Splits the data into multiple chunks without breaking consistency.
- Sends the information to the Probe/Adapter

The **Push Adapter** is a generic framework for easily configuring push adapters, using only XML and Groovy¹. It allows easy mapping of the data from the UCMDB data model into the SMA-X data model, and the transfer of this converted data into the Push Connector. For more information, see **Developing Push Adapters** in the *Developer Reference section of the UCMDB Help*.

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

Supported Versions

This integration supports Micro Focus SMA-X version 1.00 and later.

How to Integrate UCMDB and SMA-X

To set up integration between UCMDB and SMA-X, you must complete the following steps:


1. ["Set up UCMDB" below](#)
2. ["Push CI Data from UCMDB to SMA-X" on page 104](#)

Set up UCMDB


To set up UCMDB, complete the following steps:

Create an Integration Point in UCMDB

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

1. Log on to UCMDB as an administrator.
2. Go to **Data Flow Management > Integration Studio**. UCMDB displays a list of existing integration points.
3. Click **New Integration Point** . The New Integration Point dialog box is displayed.
4. Complete the Integration and Adapter Properties fields as shown in the following table:

Field	Required	Description
Integration Name	Yes	Type the name (unique key) of the integration point.
Integration Description	No	Type a description of the current integration point.
Adapter	Yes	Select HP Software Products / Micro Focus Products > SMA-X >SMA-X Push Adapter .
Is Integration Activated	Yes	Select this option to indicate the integration point is active.
Hostname/IP	Yes	Type the external hostname or IP Address of the SMA-X instance.
Port	Yes	The external communication port of the SMA-X instance. Note: The Adapter automatically populates this field with the default SSL port 443.
Web application Root context	Yes	The Root context of SMA-X. Note: The Adapter automatically populates this field.
Protocol	Yes	The protocol used to connect with SMA-X. The only valid protocols are HTTPS and HTTP. Note: <ul style="list-style-type: none"> ◦ The Adapter automatically populates this field with the HTTPS protocol. ◦ For SSL connection to a specific SMA-X instance, you must first import the certificate from the SMA-X instance to the probe keystore. The keystore path is: \\<UCMDBHOST>\c:\UCMDB\DataFlowProbe\bin\jre\lib\security\cacerts

Field	Required	Description
Credentials ID	Yes	Click  and select Generic protocol. Do one of the following: <ul style="list-style-type: none"> Select the appropriate credential Create a new credential. For more information, see "How to Create a New Credential" on the next page
Data Flow Probe	Yes	Select the name of the Data Flow Probe/Integration service used to execute the synchronization from. Note: Do not use Auto-Select for this field.
Additional Probes	No	Select additional probes to use when pushing to SMA-X in order to increase redundancy.

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


6. Click **OK**.

The integration point is created and its details are displayed.

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

How to Create a New Credential

If you need to create a new credential, you must complete the following steps:

1. Select Generic Protocol
2. Click **Create new connection details for selected protocol type** .
3. Populate the fields on the Generic Protocol Parameters page, as follows:

Field	Value
Network Scope	Use the default value.
User Label	Type a label for the credential.
User Name	The name of the user used to connect to SMA-X. Note: The user must have integration level permission in SMA-X.
Password	The password of the user needed to connect to SMA-X.

Push CI Data from UCMDB to SMA-X

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

1. Log on to UCMDB as an administrator.
2. Go to **Data Flow Management > Integration Studio**. UCMDB displays a list of existing integration points.
3. Select the integration point you created for SMA-X.
4. Select the default **SAW Push** job, or any other job you have created for this integration.




UCMDB Creates a default data push job when creating an integration point. The following table lists the Topology Query Language (TQL) queries in the default data push job. If required, you may create, update, or remove TQL queries for the push job. You may also need to update the mapping. See ["Customize an Existing Mapping" on page 112](#).

To access the out-of-the-box TQL queries for push, go to **Modeling > Modeling Studio > Resources**, select **Queries** for Resource Type, and then go to **Root > Integration > Service Anywhere Push**.

Note: The order in which the TQL queries are resolved is important.

TQL Query	Description
SAW_BusinessElement	<p>Pushes Business Activity, Business Application, Business Function, Business Process, Business Service, Business Infrastructure, Business Transaction, CI Collection, Functional Groups, and Organization CIs.</p> <p>Note: Persons are not synchronized in this integration. You must import Persons into SMA-X using LDAP.</p> <p>Mapping XMLs:</p> <ul style="list-style-type: none"> businessElementCisMappings.xml businessElementrelationsMappings.xml
SAW_Computer	<p>Pushes nodes (Computers, Network Devices, etc.) and Location CIs.</p> <p>Mapping XMLs:</p> <ul style="list-style-type: none"> computerCisMappings.xml computerRelationsMappings.xml
SAW_ComputerElement	<p>Pushes Buffer, File System, File System Export, Installed Software, and Logical Volume CIs.</p> <p>Mapping XMLs:</p> <ul style="list-style-type: none"> computerCisMappings.xml computerRelationsMappings.xml
SAW_ComputerPhysicalElements	<p>Pushes CPU, Disk Device, Environmental Sensor, Fan, Hardware Board, Memory Unit, Physical Ports, and Power Supply CIs.</p> <p>Mapping XMLs:</p> <ul style="list-style-type: none"> computerPhysicalElementCisMappings.xml computerPhysicalElementRelationsMappings.xml

TQL Query	Description
SAW_ComputerRelations	<p>Pushes only relationships.</p> <p>Note: The relationships that are pushed depend on the CIs pushed in the preceding TQL queries.</p> <p>Mapping XMLs:</p> <ul style="list-style-type: none"> computerRelationsMappings.xml
SAW_NetworkEntity	<p>Pushes Interface, IpAddress, IpSubnet, Layer2 Connection, and VLAN CIs.</p> <p>Mapping XMLs:</p> <ul style="list-style-type: none"> computerCisMappings.xml computerRelationsMappings.xml
SAW_RunningSoftware	<p>Pushes Application Server, Cluster Resource Group, Cluster Software, Failover Cluster, IpService Endpoint, Running Software, and URI Endpoint CIs.</p> <p>Mapping XMLs:</p> <ul style="list-style-type: none"> runningSoftwareCisMappings.xml runningSoftwareRelationsMappings.xml

5. Run the job manually to see if the integration job works properly:
 - a. To push all the relevant data for the job, click **Full Synchronization** .
 - b. To push only the changes in the data since the job last executed, click **Delta Synchronization** .
6. Wait for the job to complete; click **Refresh**  multiple times as needed until the job is completed.
7. When the job is completed, the job status becomes one of the following depending on the results:
 - Succeeded
 - Completed
 - Failed
8. Click the **Statistics** tab to view the results; if any errors occur, click the **Query Status** tab and **Job Errors** tab for more information.


Note: For details about these tabs and managing the integration, see **Integration Jobs Pane**

in the *Data Flow Management* section of the *UCMDB Help*.

If the job completes successfully, you can view the UCMDB CI data in SMA-X.

Schedule Data Push Jobs

To schedule job executions directly from a data push job:

1. Log in to UCMDB as an administrator.
2. Go to **Data Flow Management > Integration Studio**. UCMDB displays a list of existing integration points.
3. Select the integration point you created for the UCMDB - SMA-X.
4. Select the push job.
5. Click **Edit Integration Job** .

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

6. Define a schedule for Changes Sync.
 - a. Click on the **Changes Synchronization** tab.
 - b. Select the **Scheduler enabled** option.
 - c. Select the scheduling options you want to use.
7. Click the **All Data Synchronization** tab and select the scheduling options you want to use.
8. Click **OK**.
9. Save the integration point.

Note: The out-of-the-box default job schedule is for a Changes Synchronization data push every 2 hours.

Tailor the Integration

This section includes:

Integration Architecture	108
Change Adapter Settings	111
Customize an Existing Mapping	112
Add a New Mapping to the Integration	113

Integration Architecture

This section includes:

- ["Data Flow Architecture" below](#)
- ["Integration TQL Queries" on the next page](#)
- ["SMA-X Rules and Flows" on the next page](#)
- ["Data Mapping" on the next page](#)
- ["Push Mapping" on the next page](#)

Data Flow Architecture

1. The Push Engine executes the TQL query.
2. For a differential flow, the data is compared to the last synchronized data, and only the changes are forwarded.
3. Data is then pushed to the Push Adapter.
4. The Push Adapter loads the correct mapping for the specific TQL query.
5. Data is mapped from the UCMDB data Model into the SMA-X Data model according to the mapping XML.
6. Data is sent to the Push Connector.
7. The Push Connector orders all the data in a set of dependency trees, starting with the records that

do not depend on any other record.

8. The Push Connector starts reconciling and pushing any record without any dependencies, or a record whose dependencies have already been reconciled/pushed to SMA-X.
 - a. The Push Connector first tries to reconcile with existing records
 - b. If it finds a match, it attempts to update that record,
 - c. If it does not find a match, it attempts to create a new record.

Integration TQL Queries

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

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

SMA-X Rules and Flows

For details of SMA-X rules and flows, refer to the SMA-X documentation.

Data Mapping

For details, see Developing Push Adapters in the *Developer Reference section of the UCMDB Help*.

Push Mapping

The following table describes variables in the Generic Push Adapter XML mapping file that are specifically required for integration with SMA-X.

Variable	Description
externalId	The external ID of the CI in UCMDB.
serviceName	The Web service name used in the SOAP request.
attributesHierarchy	The location of the CI attributes in the SOAP request.
keysHierarchy	The location of the CI keys in the SOAP request.
keyTag	The key in the SOAP request that passes the value of the ID.
keyVal	The ID value from UCMDB.
retrieveRequestName	The name of the SOAP request for a retrieval operation.
createRequestName	The name of the SOAP request for a creation operation.
deleteRequestName	The name of the SOAP request for a delete operation.
updateRequestName	The name of the SOAP request for an update operation.

Change Adapter Settings

To change adapter settings:

1. Go to **Data Flow Management > Adapter Management > SAWPushAdapter > Adapters**.
2. Right-click **SAWPushAdapter** and click **Edit Adapter Source**.
3. Scroll down to the **<adapter-settings>** tag.
4. Edit the settings as required and click the Save button.

The following table shows the Settings relevant to the SMA-X Push Adapter:

Setting	Default	Description
replication.chunk.size	4,000	Defines the requested number of CIs and Relationships sent in each chunk. It is possible to adjust this setting to try and improve performance of the server and the Probe.
is.new.generic.push.adapter	true	Enables use of the correct generic Push Adapter XSD.
PushConnector.class.name	com.hp.ucmdb.adapters.sawpush.SAWPushAdapter	The name of the Java class used to load the Push Connector. Note: Only change this setting if you are not using the out-of-the-box connector.
support.ci.type.mapping	true	Specifies the mapping of single CIs to single CIs.
parallel.thread.pool.size	5	The amount of threads used in Parallel Push Mode. The more threads, the more CPU used. Increasing the pool size may lower performance.

Customize an Existing Mapping

This example shows you how to add a new attribute to the integration, including the TQL query, and Push Adapter Mapping. It enables the integration to push the attribute to SMA-X.

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

1. Add the new attribute to an existing SMA-X TQL query layout.

In this step you add the attribute of a specific CI to the integration TQL query, so that you can use the attribute and value in the mapping.

- a. Go to **Modeling > Modeling Studio > Resources** and select the **Queries** Resource Type.
- b. Go to **Queries > Root > Integration > Service Anywhere Push**.
- c. Select the TQL query containing the CI to which you want to add a new attribute, and double-click. The query is displayed in the main pane.

For example, if you are adding a new attribute **bt_example** to the Business Transaction CI, double-click the SAW_BusinessElement query.

- d. Select the specific CI for which you want to add an attribute, right-click and select **Query Node Properties**. The **Query Node Properties** dialog box is displayed.
- e. Go to the **Element Layout** tab.
- f. Move the new attribute to the **Specific Attributes** box.
- g. Click **OK**.
- h. Save the query.

2. Add the new attribute to the corresponding push adapter mapping.

In this step, you take the value from the TQL result, and remodel it to the SMA-X data model.

- a. Go to **Data Flow Management > Adapter Management > Packages > SAWPushAdapter > Configuration Files**, and select the XML file corresponding to the CI to which you have added a new attribute.

For example, if you are adding a new attribute **bt_example** to the **Business Transaction** CI, select the **businessElementCisMappings.xml** file.

- b. Go to the **source_ci_type** XML tag for the element that matches the name of the element in the

TQL you edited.

Continuing the previous example, you go to:

```
<source_ci_type query-element-name="BusinessTransaction*"
  extends="AbstractBusinessNode">
```

- c. Add the target mapping to hold the value of the new attribute.

Continuing the previous example:

```
<target_mapping name="bt_example" datatype="STRING"
  value ="BusinessTransaction.attributeExists('bt_example') ?
BusinessTransaction['bt_example']: 'MISSING'"/>
```

- d. Click **OK**.


Add a New Mapping to the Integration

This example shows how to add a new TQL query and push-mapping to the integration for a CIT called **Example**. It consists of the following steps:

Step 1: Create a TQL Query

1. Go to **Modeling > Modeling Studio > New > Query**.
2. From the **CI Types** tab, add **Example CIT** to the query.
3. Right-click the **Example Query Node** and select **Query Node Properties**.
4. Go to the **Element Layout** tab.
5. Select **Select attributes for layout**.
6. In the **Attributes condition** drop down, select **Specific Attributes**, and add the new attributes that you want to map and push to Service Anywhere.
7. Click **OK**.
8. Save the query to **Modeling > Modeling Studio > Resources > Queries > Root > Integration > Service Anywhere Push**.

Step 2: Create a Push-Mapping

1. Go to **Data Flow Management > Adapter Management > SAWPushAdapter**.
2. Click **Create new resource**  and select **New Configuration File**.
3. Type the a name for the file.
4. Select the **SAWPushAdapter** package.
5. Click **OK**.
6. Copy the following into the newly created XML file, replacing **ExampleCI** with the name of the relevant CI:

```
<integration
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="../generic-push-adapter.xsd">
  <info>
    <source name="UCMDB" versions="10.0" vendor="HP"/>
    <target name="ServiceAnywhere" versions="1.0" vendor="HP"/>
  </info>
  <import>
    <scriptFile path="mappings.scripts.SAWPushFunctions"/>
  </import>
  <targetcis>
    <source_ci_type query-element-name="ExampleCI*">
      <target_ci_type name="ExampleCI">
        <variable name="externalId" datatype="STRING"
          value="ExampleCI['external_id_obj']"/>
        <variable name="serviceName" datatype="STRING"
          value="'CMDB_UDM_1_6'"/>
        <variable name="attributesHierarchy" datatype="STRING"
          value="'model.instance'"/>
        <variable name="keysHierarchy" datatype="STRING"
          value="'model'"/>
        <variable name="keyTag" datatype="STRING" value="'ID'"/>
        <variable name="keyVal" datatype="STRING"
          value="ExampleCI['cldb_id']"/>
        <variable name="retrieveRequestName" datatype="STRING"
          value="'RetrieveExampleCI'"/>
        <variable name="createRequestName" datatype="STRING"
          value="'CreateExampleCI'"/>
        <variable name="deleteRequestName" datatype="STRING"
          value="'DeleteExampleCI'"/>
        <variable name="updateRequestName" datatype="STRING"
          value="'UpdateExampleCI'"/>
        <target_mapping name="Name" datatype="STRING"
          value="ExampleCI.attributeExists('name') ?
```



```

      ((ExampleCI['name']==null)
      ?ExampleCI['display_label']:ExampleCI['name']) :
      'MISSING'"/>
    <target_mapping name="Description" datatype="STRING"
    value="ExampleCI.attributeExists('description')
    ? ExampleCI['description'] : 'MISSING'"/>
    <target_mapping name="DisplayLabel" datatype="STRING"
    value="ExampleCI.attributeExists('display_label')
    ? ExampleCI['display_label'] : 'MISSING'"/>
    <target_mapping name="GlobalId" datatype="STRING"
    value="ExampleCI.attributeExists('global_id')
    ? ExampleCI['global_id'] : 'MISSING'"/>
  </target_ci_type>
</source_ci_type>
</targetcis>
</integration>



```

7. Click **OK**.

Step 3: Create a Job with the New TQL Query

1. Go to **Data Flow Management > Integration Studio**.
2. Create an Integration Point with Service Anywhere.
3. In the **Integration Jobs** tab, click **New Integration Job** .
4. Insert a job name in the **Name** field.
5. Click **Add Query** , and choose the newly created query.
6. Click **OK**.

Step 4: Run the Job

1. Click on the job created in "[Step 3: Create a Job with the New TQL Query](#)".
2. Click **Full Synchronization** .
3. Wait for the job to complete; click **Refresh**  multiple times as needed until the job is completed.
4. Make sure that the status is **Succeeded**.

Step 5: View the Results

1. Go to Service Anywhere.
2. Validate that the new attribute, pushed in the previous steps, is displayed.

Troubleshooting and Limitations – SMA-X Push Integration

- The **SMA-X Push Adapter** assumes that relationships comply with the Universal Data Model specification. The adapter pushes CIs that comply with the specification, and ignores those that do not comply.
- **PROBLEM:** On the first synchronization from SMA-X to UCMDB, you may see an error message displayed similar to the following:

Integration Point doesn't exist. No adapter for given target.

Workaround: To remedy this, do the following:

- a. Log in to the UCMDB instance.
- b. Go to **Data Flow Management > Integration Studio**.
- c. Right-click the integration point: **<endpoint name>_<tenant id>**, and click **Edit**. The Edit Integration Point dialog box is displayed.
- d. Deselect **Is Integration Activated**.
- e. Click **OK**.
- f. Reselect **Is Integration Activated**.
- g. Click **OK**.
- h. Go to the job in UCMDB and run a full synchronization.

Chapter 10: Micro Focus Service Manager Integration

You can establish and maintain a connection between Universal CMDB (UCMDB) and Service Manager (SM) to synchronize configuration items (CIs) and CI relationships between the two systems and trigger appropriate actions in Service Manager.

The UCMDB-SM integration is based on an integration adapter. The following Service Manager integration adapters are available with the current Content Pack for the supported product versions as listed in the table below.

UCMDB	SM9.40 or earlier	SM9.41 or later
10.1x	ServiceManagerAdapter9.x	ServiceManagerAdapter9.41
10.20 or later		ServiceManagerAdapter9.41
		ServiceManagerEnhancedAdapter9.41

For instructions on how to set up the integration by using a specific Service Manager adapter, see the integration guide for the corresponding adapter as listed below.

Service Manager Adapter	Adapter User Guide	Supported Service Manager Version
ServiceManagerAdapter9.41	See CP_HP_Integrations_SM941_xslt.pdf	9.41
ServiceManagerEnhancedAdapter9.41	See CP_HP_Integrations_SM941_Generic.pdf	9.41
ServiceManagerAdapter9.x	See CP_HP_Integrations_SM940_xslt.pdf	9.40

Chapter 11: Micro Focus Storage Essentials (SE) Integration

This chapter includes:

Overview	120
Supported Versions	120
How to Perform the SE Integration	120
Storage Essentials Integration Packages	122
Adapter Parameters	123
Discovered CITs and Relationships	123
Node Details	125
SAN Topology	126
Storage Topology	127
Views	127
Storage Array Details	128
FC Switch Details	128
FC Switch Virtualization	129
Storage Pool Details	129
Host Storage Details	130
SAN External Storage	130
SAN Topology	131
Storage Topology	132
FC Port to FC Port	132
Impact Analysis Rules	133
FC Port to FC Port	133
FC Switch Devices to FC Switch	133
Host Devices to Host	134
Logical Volume to Logical Volume	134
Storage Array Devices to Storage Array	135
Reports	135

Storage Array Configuration135

Host Configuration136

Storage Array Dependency 136

Host Storage Dependency 136

Storage Pool Configuration137

Troubleshooting and Limitations – SE Integration137

Overview

Integration involves synchronizing devices, topology, and the hierarchy of a customer storage infrastructure in the Universal CMDB database (CMDB). This enables Change Management and Impact Analysis across all business services mapped in UCMDB from a storage point of view.

You integrate SE with UCMDB using Data Flow Management (DFM).

When you activate the Storage Essentials integration, DFM retrieves data from the SE Oracle database and saves CIs to the Universal CMDB database. Users can then view SE storage infrastructure in UCMDB.

The data includes information on storage arrays, fibre channel switches, hosts (servers), storage fabrics, logical volumes, host bus adapters, storage controllers, and fibre channel ports. Integration also synchronizes physical relationships between the hardware, and logical relationships between logical volumes, storage zones, storage fabrics, and hardware devices.

Supported Versions

The integration procedure supports SE versions 6.x, 9.4, 9.41, 9.5, 9.6, 9.6.1, and 9.70.

How to Perform the SE Integration

This task includes the steps to perform SE-UCMDB integration.

1. Prerequisites.

- a. Configure database connection for the integration (PostgreSQL only)
 - i. Download the latest JDBC driver for PostgreSQL from <http://jdbc.postgresql.org/download.html> (the package name is **postgresql-9.3-1102.jdbc41.jar**, for jdbc41) and place it on the UCMDB server in the following directory:

```
<DataFlowProbe_  
Home>\runtime\probeManager\discoveryResources\db\postgresql
```
 - ii. Add the following line to the **C:\HP\data\HPSEData\pg_hba.conf** file on the SE PostgreSQL database server:

host	all	report_user	all	trust
------	-----	-------------	-----	-------

- b. Add the Storage Essentials server IP address to the Data Flow Probe IP ranges.

In UCMDB, go to **Data Flow Management > Data Flow Probe Setup**. Choose the probe corresponding to the UCMDB server and add the IP address or the IP range of the SE database server (whether Oracle or PostgreSQL).

- c. Define credentials for the SE database (Oracle and/or PostgreSQL) using the Generic DB Protocol.
- d. Run the following jobs for Oracle and/or PostgreSQL:
- **For Oracle database**, run one of the following sets of jobs to trigger SE discovery:
 - Set 1:
 - Network Infrastructure > Basic > **Range IPs by ICMP** (discovers the IP address of the SE server.)
 - Network Infrastructure > Host Connection > **Host Connection by Shell/WMI/SNMP** (discovers operating system information on the SE server)
 - Hosts and Resources > Basic Applications > **Host Applications by Shell/SNMP/WMI** (discovers the Oracle database instance used by SE)
 - Database > Oracle > **Oracle Database Connection by SQL** (discovers Oracle databases using the Generic DB Protocol (SQL))
 - Set 2:
 - Network Infrastructure > Basic > **Range IPs by ICMP** (discovers the IP address of the SE server)
 - Database > Oracle > **Databases TCP Ports**
 - Database > Oracle > **Oracle Database Connection by SQL** (discovers Oracle databases using the Generic DB Protocol (SQL))
 - **For PostgreSQL database**
 - Network Discovery > Basic > **Class C IPs by ICMP** (if we have provided single IP for the PostgreSQL server)
 - Network Infrastructure > Basic > **Range IPs by ICMP** (if we have provided range of IPs for the PostgreSQL server)
 - Database > PostgreSQL > **Databases TCP Ports**
 - Database > PostgreSQL > **PostgreSQL Connection by SQL**

2. Create a new integration point.

For details on running integration jobs, see "Integration Studio" in the *Data Flow Management section of the UCMDB Help*.

- a. Provide a name and description for the integration point.
- b. Under **Integration Properties > Adapter**, select the **Storage Essentials** adapter.
- c. Under **Adapter Properties > Data Flow Probe**, select the Data Flow Probe.
- d. Under **Adapter Properties > Trigger CI instance** select:
 - i. **Select Existing CI** (if you have a valid, existing CI). The **Select Existing CI** pane appears. Select the CI or
 - ii. **Create New CI** (if you need to create a new CI). The **Topology CI Creation Wizard** appears. Complete the creation of the CI using the Wizard.

Note: For details on the Topology CI Creation Wizard, see "Topology CI Creation Wizard" in the *Data Flow Management section of the UCMDB Help*.

- e. Verify the credentials for the chosen CI instance. Right-click **Trigger CI instance** and select **Actions > Edit Credentials Information**.
- f. Save the integration point.

3. Run the Storage Essentials job.

Storage Essentials Integration Packages

The integration includes two UCMDB packages:

- **SE_Integration.zip**. Contains the trigger query for SE discovery, discovery script, adapter, and job.
- **Storage_Basic.zip**. Contains the new CI Type definitions, views, reports, and impact analysis rules. This package is common to all Storage Management integration solutions.

Tip: You can include the SE job in the DFM schedule.

For details, see "New Integration Job/Edit Integration Job Dialog Box" in the *Data Flow Management section of the UCMDB Help*.

Adapter Parameters

This job runs queries against Oracle materialized views that are installed and maintained by Storage Essentials in the Oracle database. The job uses a database CI as the trigger.

A switch or server in SE inherits from a Node CIT in UCMDB based on the following adapter parameters:

Parameter	Description
allowDNSLookp	<p>If a node in the SE database does not have an IP address but has a DNS name, it is possible to resolve the IP address by the DNS name.</p> <p>True: If a node does not have an IP address, an attempt is made to resolve the IP address by DNS name (if a DNS name is available).</p> <p>Default: False</p>
ignoreNodesWithoutIP	<p>Defines whether or not nodes in SE without IP addresses should be pulled into UCMDB.</p> <ul style="list-style-type: none">• True. Nodes without IPs are ignored.• False. A Node CI is created with an SE ID as the node key attribute. <p>Note: Setting this parameter to False may result in duplicate CIs in the CMDB.</p> <p>Default: True</p>
ignorePortsWithoutWWN	<p>If set to true, the integration ignores Fiber Channel Ports that do not have a WWN.</p> <p>Default: True</p>

Discovered CITs and Relationships

This section describes SE storage entities in UCMDB:

- **Fibre Channel Connect.** Represents a fibre channel connection between fibre channel ports.
- **Fibre Channel HBA.** Has change monitoring enabled on parameters such as state, status, version, firmware version, driver version, WWN, and serial number. A Fibre Channel HBA inherits

from the Node Resource CIT.

- **Fibre Channel Port.** Has change monitoring enabled on parameters such as state, status, WWN, and trunked state. Since a Fibre Channel Port is a physical port on a switch, it inherits from the Physical Port CIT under the NodeElement Resource CIT.
- **Fibre Channel Switch.** Falls under the Node CIT because SE maintains an IP address for each switch. Parameters such as status, state, total/free/available ports, and version are change monitored.

This package retrieves Fibre Channel Switch details from the **mvc_switchsummaryvw** and **mvc_switchconfigvw** views. The job retrieves detailed information about Fibre Channel Ports on each switch from the **mvc_portsummaryvw** view.

- **Logical Volume.** Represents volumes on storage arrays and hosts with change monitoring on availability, total/free/available space, and storage capabilities.
- **Tape:** Represents a tape library.
- **Storage Array.** Represents a storage array with change monitoring on details such as serial number, version, and status. Since a storage array may not have a discoverable IP address, it inherits from the Network Device CIT.
- **NetApp Filer.** This is a specialized storage array from the NetApp application.

Both Storage Array and NetApp Filer CITs retrieve details from the **mvc_storagesystemssummaryvw** view. DFM retrieves detailed information on Storage Processors and HBAs from the **mvc_storageprocessorssummaryvw** and **mvc_cardsummaryvw** tables respectively.

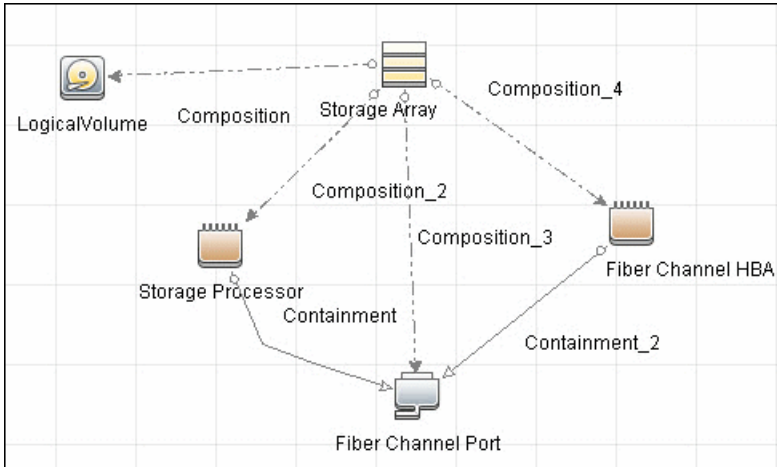
The SE database may possibly not be able to obtain IP address information on storage arrays for a variety of technical and policy related reasons.

Since Fibre Channel Ports may be present on a storage array, Storage Processor, or HBA, DFM uses three separate queries to retrieve Fibre Channel Ports for each storage array. Detailed information about Fibre Channel Ports on each array is retrieved from the **mvc_portsummaryvw** view. Since this view uses a container ID as the key, DFM queries the view by container ID for each storage array, each Storage Processor on a storage array, and each HBA on a storage array.

DFM retrieves detailed information about Logical Volumes on each storage Array from the **mvc_storagevolumessummaryvw** view.

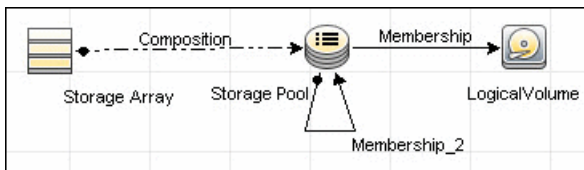
DFM retrieves detailed information about Tape Library from the **mvc_tapelibrarysummaryvw** view.

Results from these queries populate a map as shown below:



- **Storage Fabric.** Inherits from the Network Resource CIT and represents a storage fabric. This CIT has no change monitoring enabled.
- **Storage Processor.** Represents other storage devices such as SCSI controllers, and inherits from the Host Resource CIT. A Storage Processor CIT monitors change on parameters such as state, status, version, WWN, roles, power management, and serial number.
- **Storage Pool.** Storage Pool information is also collected from each storage array using the query below.

Results from this query populate a map as shown below:



Node Details

DFM retrieves Host details from the **mvc_hostsummaryvw** view and detailed information on HBAs from the **mvc_cardsummaryvw** view.

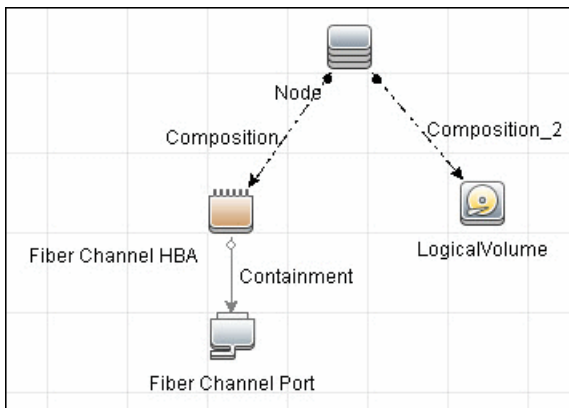
SE maintains information on Operating Systems, IP address, and DNS name on each host. DFM uses this information to create Node CIs (UNIX or Windows) and IPAddress CIs.

Since UCMDB uses the IP address of a node as part of its primary key, DFM attempts to use the IP address from SE for this purpose. If an IP address is not available, DFM then attempts to resolve the hosts IP address using a DNS name. If neither an IP address nor a DNS name is available, DFM ignores the host (see ["Adapter Parameters" on page 123](#)).

Similar to Storage Arrays, a node may have Fibre Channel Ports directly associated with itself or on HBAs on the host. The DFM job uses three separate queries to retrieve Fibre Channel Ports for each host. The job retrieves detailed information about Fibre Channel Ports on each host from the **mvc_portsummaryvw** view. Since this view uses a ContainerID attribute as the key, the job queries the view by containerID for each host, and each HBA on a host.

Finally, DFM retrieves detailed information about Logical Volumes on each host from the **mvc_hostvolumesummaryvw** and **mvc_hostcapacityvw** views. The **mvc_hostcapacityvw** view maintains capacity information for each volume over multiple instances in time, and the job uses only the latest available information.

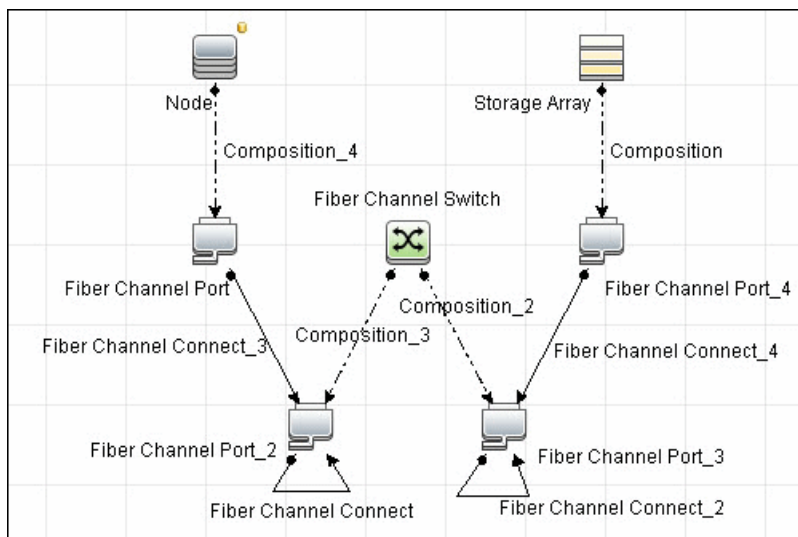
Results from these queries populate a map as shown below:



SAN Topology

SAN Topology consists of the Fibre Channel network topology and includes (fibre channel) connections between Fibre Channel Switches, Hosts, and Storage Arrays. SE maintains a list of WWNs that each Fibre Channel Port connects to, and this package uses this list of WWNs to establish Fibre Channel Connection links.

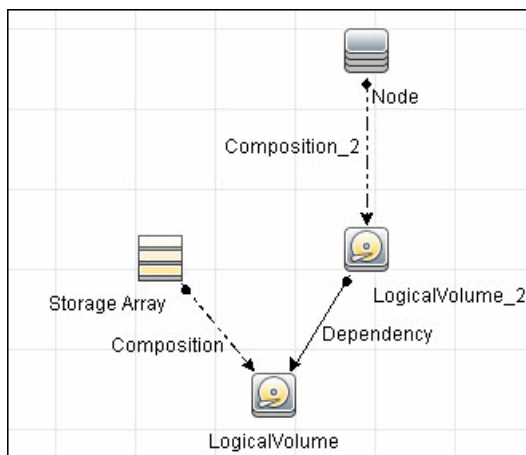
Results from these queries populate a map as shown below:



Storage Topology

Storage topology consists of relationships between Logical Volumes on a host and Logical Volumes on a Storage Array. DFM uses multiple tables to identify this relationship as shown in the query below. This view is a summary of all of the above information.

Results from these queries populate a map as shown below:



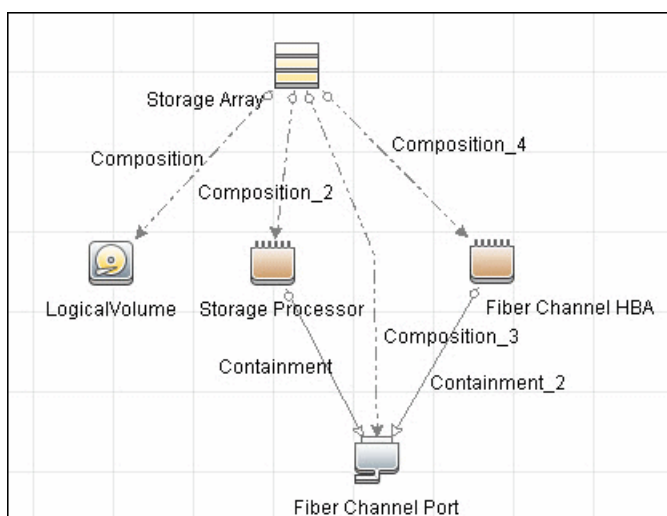
Views

The SE package contains views that display common storage topologies. These are basic views that can be customized to suit the integrated SE applications.

Storage Array Details

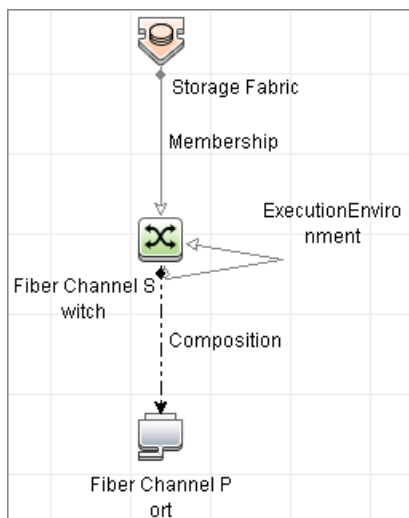
This view shows a Storage Array and its components including Logical Volumes, HBAs, Storage Processors, and Fibre Channel Ports. The view shows each component under its container Storage Array and groups Logical Volumes by CI Type.

Storage Array does not require all components in this view to be functional. Composition links stemming from the Storage Array have a cardinality of zero-to-many. The view may show Storage Arrays even when there are no Logical Volumes or Storage Processors.



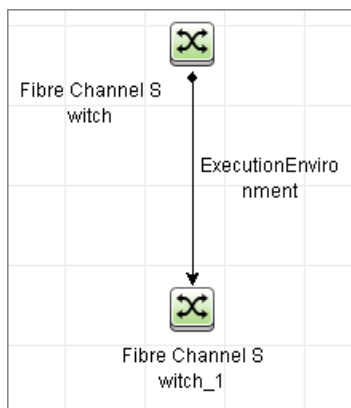
FC Switch Details

This view shows a Fibre Channel Switch and all connected Fibre Channel Ports.



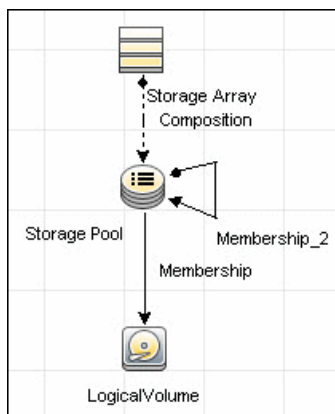
FC Switch Virtualization

FC Switch Virtualization consists of a physical switch or chassis, partitioned into multiple logical switches. Unlike Ethernet virtualization, physical ports are not shared among multiple virtual switches. Rather, each virtual switch is assigned one or more dedicated physical ports that are managed independently by the logical switches.



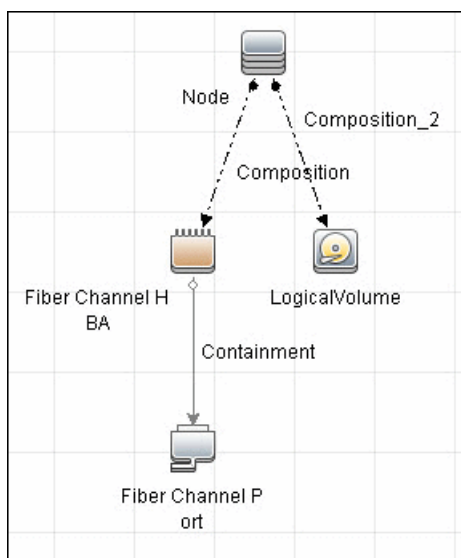
Storage Pool Details

This view shows Storage Pools with associated Storage Arrays and Logical Volumes.



Host Storage Details

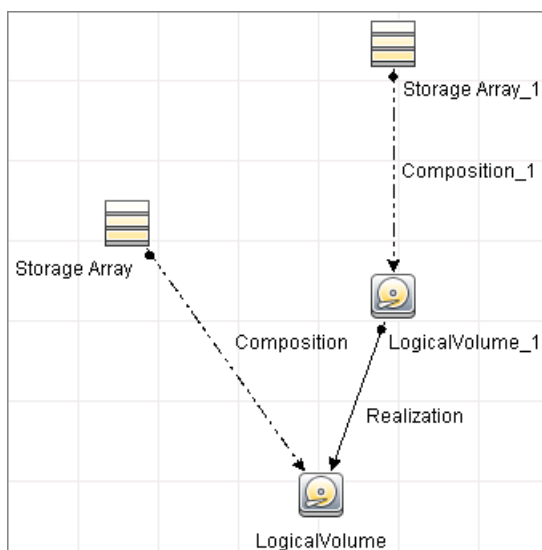
This view shows only Hosts that contain a Fibre Channel HBA or a Logical Volume. This keeps the view storage-specific and prevents hosts discovered by other DFM jobs from being included in the view.



SAN External Storage

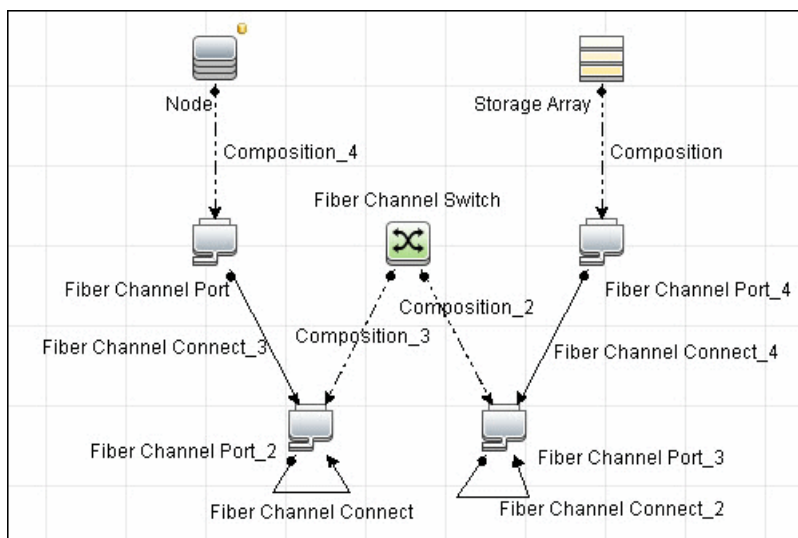
External storage configuration consists of a storage array presenting a logical volume that, in reality, belongs to another storage array. This is typically used in configurations where high-end, more expensive, front-end arrays present volumes from back-end, cheaper, storage to servers. The goal of

this type of virtualization is to virtualize multiple disk arrays from different vendors, scattered over the network, into a single monolithic storage device that can be managed uniformly.



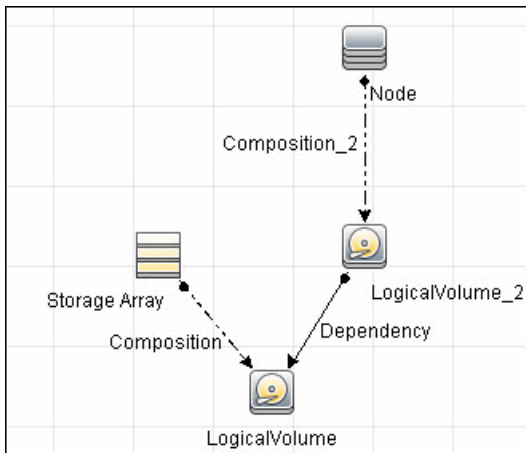
SAN Topology

This view maps physical connections between Storage Arrays, Fibre Channel Switches, and Hosts. The view shows Fibre Channel Ports below their containers. The view groups the Fibre Channel Connect relationship CIT to prevent multiple relationships between the same nodes from appearing in the top layer.



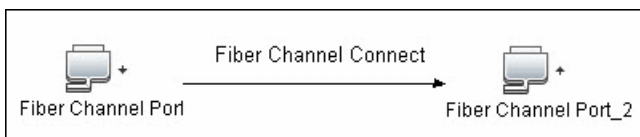
Storage Topology

This view maps logical dependencies between Logical Volumes on Hosts and Logical Volumes on Storage Arrays. There is no folding in this view.



FC Port to FC Port

This rule propagates events on a Fibre Channel Port to another connected Channel Port.



Example of HBA crashing on a Storage Array:

- The event propagates from the HBA to the Storage Array and the Logical Volumes on the Array because of the Storage Devices to Storage Array rule.
- The impact analysis event on the Logical Volume then propagates to other dependent Logical Volumes through the Logical Volume to Logical Volume rule.
- Hosts using those dependent Logical volumes see the event next because of the Host Devices to Host rule.
- Depending on business needs, you define impact analysis rules to propagate events from these hosts to applications, business services, lines of business, and so on. This enables end-to-end mapping and impact analysis using UCMDB.

Impact Analysis Rules

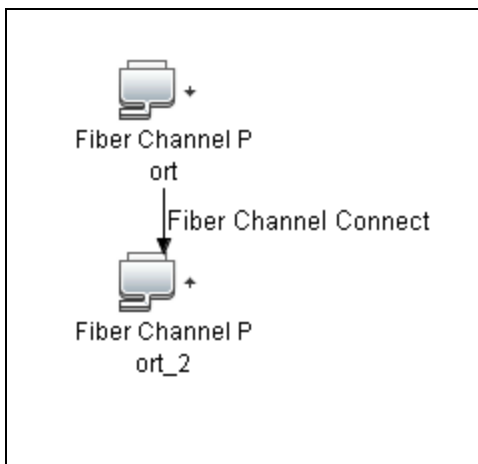
This package contains basic impact analysis rules to enable impact analysis and root cause analysis in UCMDB. These impact analysis rules are templates for more complex rules that you can define based on business needs.

All impact analysis rules fully propagate both Change and Operation events. For details on impact analysis, see "Impact Analysis Manager Page" and "Impact Analysis Manager Overview" in the *Modeling section of the UCMDB Help*.

Note: Impact analysis events are not propagated to Fibre Channel Ports for performance reasons.

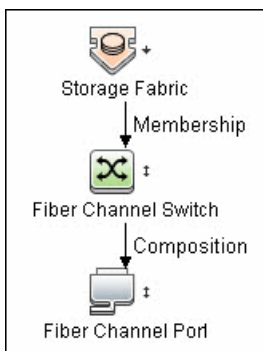
FC Port to FC Port

This impact analysis rule propagates events between related fiber channel ports.



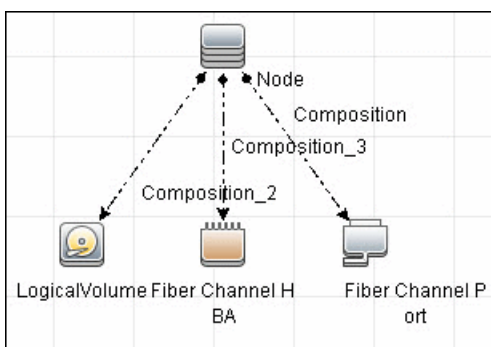
FC Switch Devices to FC Switch

This impact analysis rule propagates events from a Fibre Channel Port to and from a Switch. The event is also propagated to the associated Storage Fabric.



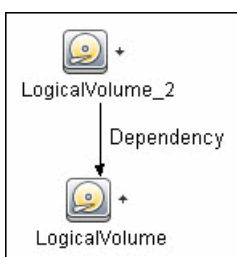
Host Devices to Host

This impact analysis rule propagates events between Fibre Channel HBAs and Hosts, and Logical Volumes on the Host.



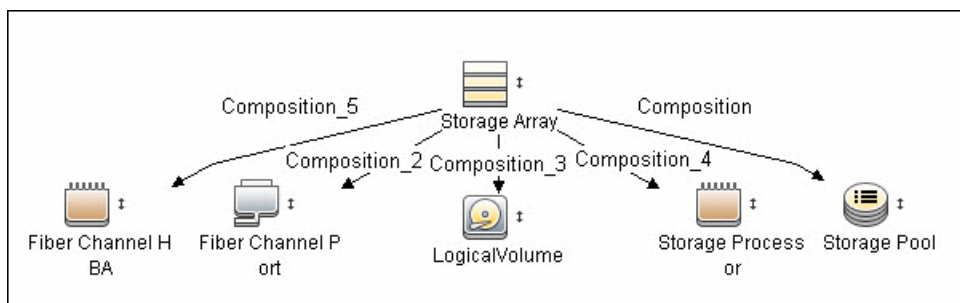
Logical Volume to Logical Volume

This impact analysis rule propagates events on a Logical Volume contained in a Storage Array to the dependent Logical Volume on the Host.



Storage Array Devices to Storage Array

This impact analysis rule propagates events between Logical Volumes, Storage Processors, Fibre Channel HBAs, and Storage Arrays.



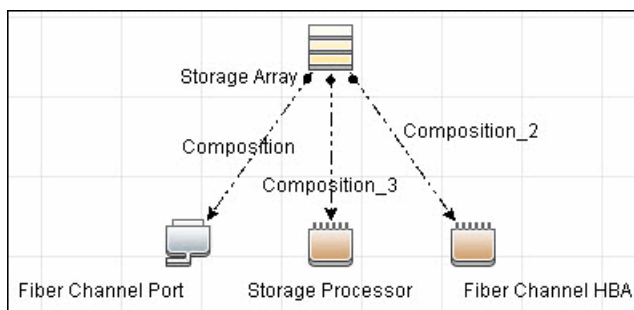
Reports

The SE package contains basic reports that can be customized to suit the integrated SE applications.

In addition to the system reports, Change Monitoring and Asset Data parameters are set on each CIT in this package, to enable Change and Asset Reports in Universal CMDB. For details see "[Storage Array Configuration](#)" below, "[Host Configuration](#)" on the next page, "[Storage Array Dependency](#)" on the next page, and "[Host Storage Dependency](#)" on the next page.

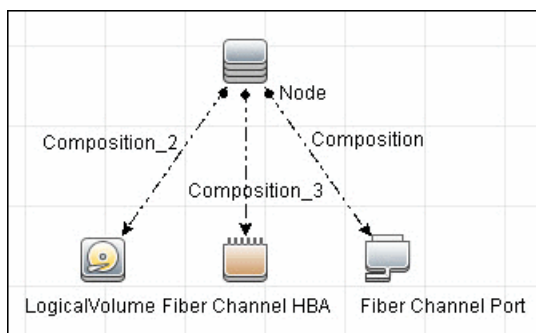
Storage Array Configuration

This report shows detailed information on Storage Arrays and its sub-components including Fibre Channel Ports, Fibre Channel Arrays, and Storage Processors. The report lists Storage Arrays with sub-components as children of the Array.



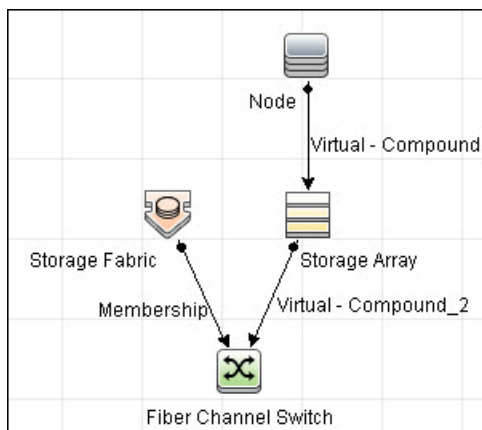
Host Configuration

This report shows detailed information on hosts that contain one or more Fibre Channel HBAs, Fibre Channel Ports, or Logical volumes. The report lists hosts with sub-components as children of the host.



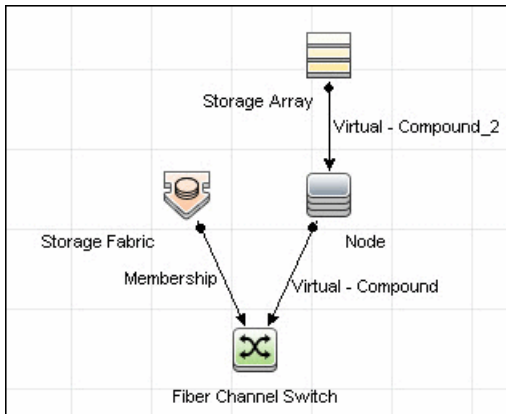
Storage Array Dependency

This report maps dependencies on a Storage Array. The report also displays information on switches connected to it.



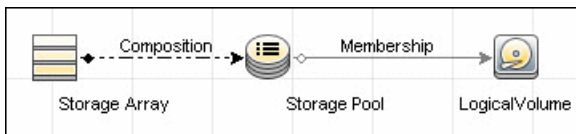
Host Storage Dependency

This report shows detailed information on storage infrastructure dependencies of a Host. The report lists hosts and dependent components.



Storage Pool Configuration

This report shows detailed information on Storage Pool configuration.



Troubleshooting and Limitations – SE Integration

This section describes troubleshooting and limitations of Storage Essentials Integration.

- **Problem:** If the SE system has duplicate entries for nodes, switches or arrays, the job produces the following error message: "**Process validator error: multiple updates in bulk...**".

Solution: This is expected behavior and does not affect population of valid CIs into UCMDB. To prevent this error message, duplicates must be removed from the SE system.

- **Error message:** "Please use the SE database with SID 'REPORT' for this integration."

Solution: For integration with SE 9.4 or higher, you should configure the integration point for a REPORT database instance (and not for an APPIQ instance as applied for SE versions up to and including 6.3).

- **Limitation:** The credentials used for integration with SE should have access to the relevant materialized view status table as detailed below:

Storage Essentials	Materialized View Status Table
Version 6.03 and earlier	appiq_system.mview_status
Later than Version 6.03 and earlier than Version 6.1	appiq_system.mviewcore_status and appiq_system.mview_status
Later than Version 6.1 and earlier than Version 9.4	appiq_system.mview_module_status
Version 9.4 and later	appiq_system.mv_report_user_status

- **Limitation:** If the discovery does not find a valid IP address and serial number for a VMWare ESX server in the Storage Essentials database, that VMware ESX server is not reported to UCMDB.

Chapter 12: Micro Focus Storage Operations Manager (SOM) Integration

Deprecated: Starting from Content Pack 26, the SOM integration is deprecated and will be removed in a future Content Pack.

This chapter includes:

Overview	140
Supported Versions	140
How to Perform the SOM Integration	140
Install the Integration	141
Enable the Integration	141
Adapter Properties	144
Discovered CITs and Relationships	145
Node Details	146
SAN Topology	147
Storage Topology	148
Views	148
Storage Array Details	149
FC Switch Details	150
FC Switch Virtualization	150
Storage Pool Details	151
Host Storage Details	151
SAN External Storage	152
SAN Topology	152
Storage Topology	153
FC Port to FC Port	153
Impact Analysis Rules	154
FC Port to FC Port	155
FC Switch Devices to FC Switch	155

Host Devices to Host	156
Logical Volume to Logical Volume	156
Storage Array Devices to Storage Array	156
Reports	157
Storage Array Configuration	157
Host Configuration	158
Storage Array Dependency	158
Host Storage Dependency	159
Storage Pool Configuration	159
Troubleshooting the SOM Integration	159
Known Issues	160

Overview

Integration involves synchronizing devices, topology, and the hierarchy of a customer storage infrastructure in the Universal CMDB (UCMDB). This enables Change Management and Impact Analysis across all business services mapped in UCMDB from a storage point of view.

Storage Operations Manager (SOM) communicates with UCMDB using Rest APIs via HTTPS protocols.

When you activate the SOM integration, HTTPS protocols retrieve data and saves CIs to UCMDB. Users can then view SOM storage infrastructure in UCMDB.

The data includes information on storage arrays, fibre channel switches, hosts (servers), storage fabrics, logical volumes, host bus adapters, storage controllers, and fibre channel ports. Integration also synchronizes physical relationships between the hardware, and logical relationships between logical volumes, storage pools, storage fabrics, and hardware devices.

Supported Versions

The integration procedure supports SOM version 10.10.

How to Perform the SOM Integration

This section includes the steps to perform SOM-UCMDB integration.

Install the Integration	141
Enable the Integration	141

Install the Integration

The SOM-UCMDB integration is delivered as the **HP_SOM_Integration.zip** file. The package must be on your local directory in order to deploy its resources.

1. Log in to UCMDB as an administrator.
2. Go to **Administration > Package Manager**.
3. Click the **Deploy packages to server (from local disk)** button.
4. In the Deploy Packages to Server dialog box, click the **Add** button.
5. Select the **HP_SOM_Integration.zip** package and click **Open**.
6. Select the resources from the package that you want to deploy. All the resources are selected by default.
7. Click **Deploy**.

Enable the Integration

This task includes the steps to perform the SOM-UCMDB integration.

This integration uses an external JAR and runs in a separate Java virtual machine (JVM) than other UCMDB integrations.

If the new data flow probe will be on the UCMDB server, ensure that UCMDB services have started completely before beginning the integration enablement procedures. To determine the status of the UCMDB services, open the URL **https://localhost:8443/status** in a web browser on the UCMDB server system.

Basic Setup

1. Go to **Data Flow Management > Data Flow Probe Setup**.

In the UCMDB system services, the UCMDB_Probe service must be up and the UCMDB_Integration_Service must be down.

2. In the **Domains and Probes** pane > **Data Flow Probes** tree, click the probe you want to use.
3. Under **Ranges**, click the ***New** icon to create a new range, and then enter the SOM management server details.
4. Click **OK** to create the range.

You can create more than one range on this page.

5. In the **Domains and Probes** pane > **Credentials** tree,
 - a. Choose **WMI Protocol** for SOM servers running Windows.
 - b. Choose **SSH Protocol** for SOM servers running Linux.
6. Click the ***New** icon in the protocol pane to set protocol parameters.
 - a. Click the **Edit** button to edit the network scope.
 - b. In the Scope Definition dialog box, click the **Selected Range** option button, and then click the ***New** icon to create a new range. Fill in the SOM server information and then click **OK** to add the network scope.
 - c. Enter the SOM server log in credentials.
 - d. Click **OK** to save the protocol.
7. Select the **HTTP Protocol**, then the ***New** icon to set protocol parameters.
 - Follow the same setup procedure as WMI and SSH above.
 - The user name must be part of the web services client role.

To view what role the user belongs to, in the SOM console go to Configurations > Security > User Account Mappings.
 - The timeout must be set to a minimum of 60,000 msec. (60 seconds)
 - Trust store information is optional.
8. Go to **Universal Discovery > Discovery Modules/Jobs** tab > **Discovery Modules** tree > **Network Infrastructure > Basic**.
 - a. Right-click one of the ICMP discovery jobs and select **Activate**.
 - b. Expand **Host Connection** in the file tree
 - For SOM servers running the Windows operating system, right-click **Host Connection by WMI** and select **Activate**.

- For SOM servers running Linux operating system, right-click **Host Connection by Shell** and select **Activate**.
9. In Integration Jobs, right-click the job that was created and select run all-data sync to perform the job.

Create the Integration

1. Add the SOM management server IP address to the Data Flow Probe IP ranges.

Go to **Data Flow Management > Data Flow Probe Setup**. Choose the probe corresponding to the UCMDB server.
2. Enter user name and password credentials to access SOM from UCMDB.
Certificate based authentication is optional.
3. Create a new Integration Point.
 - a. In the Integration Studio, click the ***New** icon to create a new integration.
 - b. Set Integration Properties.
 - i. **Integration Name**. Type the name (unique key) of the integration point.
 - ii. **Integration Description**. Type a description of the current integration point.
 - iii. **Adapter**. Click the **Select Adapter** button to select the **HP SOM Integration** adapter.
 - iv. **Is Integration Activated**. Select this option to indicate the integration point is active.
 - c. Set the adapter properties under the Integration Properties.

For more information, see ["Adapter Properties" on the next page](#).
 - i. Set the credentials ID to the HTTPS protocol credential. This is the only option supported for SOM.
 - ii. Select the Data Flow Probe from the drop-down menu.
 - iii. Click the cube next to the trigger CI instance field and choose a CI.
The CI should be set to the IP of the SOM server.
4. Run the integration created in step 3.

Note: For information about running integration jobs, see "Integration Studio" in the *Data Flow Management section of the UCMDB Help*.

Optional Configuration

For large SOM data sets, adjust the `appilog.agent.local.max.worker.runtime` value in the `<DataFlowProbeInstallDir>/conf/DataFlowProbe.properties` file to match your environment. For example, with a SOM data set of approximately 15,000 elements, setting the argument to 3600000 resulted in acceptable behavior.

After modifying the `DataFlowProbe.properties` file, restart the DataFlowProbe service.

Adapter Properties

This job uses a database CI as the trigger.

A switch or server in SOM inherits from a Node CIT in UCMDB based on the following adapter properties:

Parameter	Description
Allow DNS Lookup	<p>If a node in the SOM database does not have an IP address but has a DNS name, it is possible to resolve the IP address by the DNS name.</p> <p>True: If a node does not have an IP address, an attempt is made to resolve the IP address by DNS name (if a DNS name is available).</p> <p>Default: False</p>
Credentials ID	<p>The HTTP Protocol Credentials of the SOM management server configured in UCMDB.</p>
Ignore Nodes Without IP	<p>Defines whether or not nodes in SOM without IP addresses should be pulled into UCMDB.</p> <ul style="list-style-type: none">• True. Nodes without IPs are ignored.• False. A Node CI is created with a SOM ID as the node key attribute. <p>Note: Setting this parameter to False may result in duplicate CIs in the CMDB.</p> <p>Default: True</p>
Ignore Ports Without WWN	<p>If set to true, the integration ignores Fiber Channel Ports that do not have a WWN.</p>

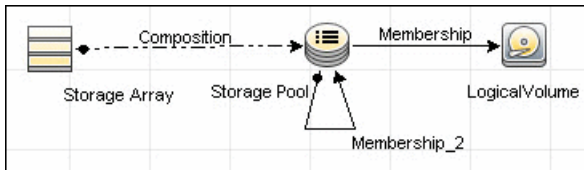
Parameter	Description
	Default: True
Remote JVM Arguments	<p>JVM parameters that are passed to the external JVM.</p> <p>Default: -Xms1024m -Xmx2560m</p> <p>Adjust the value of this argument to match your environment. For example, with a SOM data set of approximately 15,000 elements, setting the argument to -Xms25G -Xmx25G -XX:-UseGCOverheadLimit resulted in acceptable behavior.</p>
Remote JVM Class Path	<p>Set the external JVM class path.</p> <p>Note: Do not change %minimal_classpath%.</p>
Remote Process Timeout	<p>Timeout for remote process in external JVM.</p> <p>Set to a minimum of 60,000 m. sec. (60 seconds).</p> <p>Adjust the value of this argument to match your environment. For example, with a SOM data set of approximately 15,000 elements, setting the argument to 5400000 (m. sec.) resulted in acceptable behavior.</p>
Run In Separate Process	<p>If set to true, the adapter will run in the external JVM.</p> <p>Default: True</p> <p>Note: Currently, this value must remain set to True.</p>
Data Flow Probe	Name of the data flow probe for collecting SOM data.
Trigger CI Instance	IP address of the SOM management server.

Discovered CITs and Relationships

This section describes SOM storage entities in UCMDB:

- **Fibre Channel Connect.** Represents a fibre channel connection between fibre channel ports.
- **Fibre Channel Port.** Has change monitoring enabled on parameters such as state, status, WWN, and trunked state. Since a Fibre Channel Port is a physical port on a switch, it inherits from the Physical Port CIT under the NodeElement Resource CIT.
- **Fibre Channel Switch.** Falls under the Node CIT because SOM maintains an IP address for each switch. Parameters such as status, state, total/free/available ports, and version are change monitored

- **Logical Volume.** Represents volumes on storage arrays with change monitoring on availability, total/free/available space, and storage capabilities.
- **Storage Array.** Represents a storage array with change monitoring on details such as serial number, version, and status. Since a storage array may not have a discoverable IP address, it inherits from the Network Device CIT.
- **Storage Fabric.** Inherits from the Network Resource CIT and represents a storage fabric. This CIT has no change monitoring enabled.
- **Storage Processor.** Represents other storage devices such as SCSI controllers, and inherits from the Host Resource CIT. A Storage Processor CIT monitors change on parameters such as state, status, version, WWN, roles, power management, and serial number.
- **Storage Pool.** Storage Pool information is also collected from each storage array. This data collection populates a map as shown here:

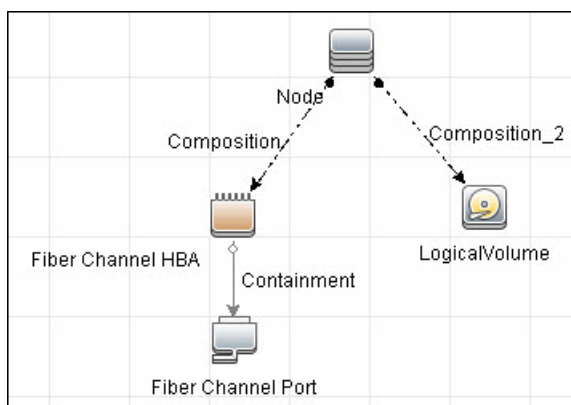


Node Details

SOM maintains information on Operating Systems, IP address, and DNS name on each host. DFM uses this information to create Node CIs (UNIX or Windows) and IpAddress CIs.

Because UCMDB uses the IP address of a node as part of its primary key, DFM attempts to use the IP address from SOM for this purpose. If an IP address is not available, DFM then attempts to resolve the host's IP address using a DNS name. If neither an IP address nor a DNS name is available, DFM ignores the host (see ["Adapter Properties" on page 144](#)).

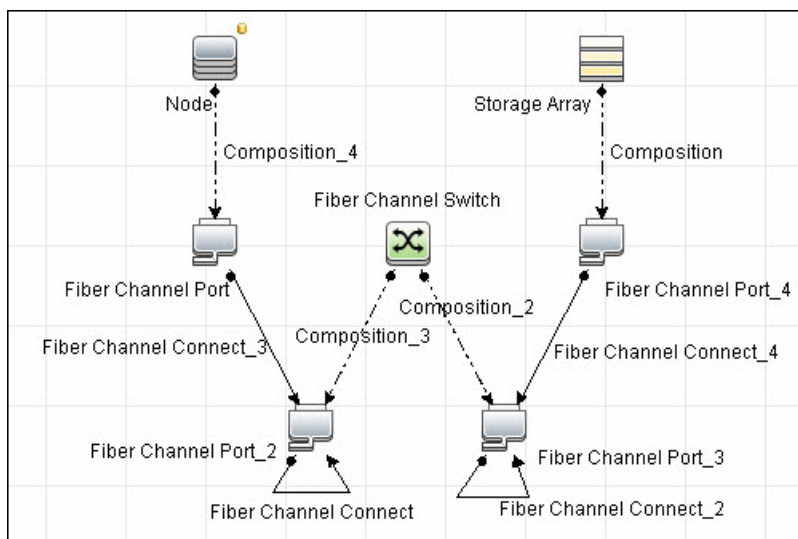
Node details populate a map as shown here:



SAN Topology

SAN Topology consists of the Fibre Channel network topology and includes (fibre channel) connections between Fibre Channel Switches, Hosts, and Storage Arrays. SOM maintains a list of WWNs that each Fibre Channel Port connects to, and this package uses this list of WWNs to establish Fibre Channel Connection links.

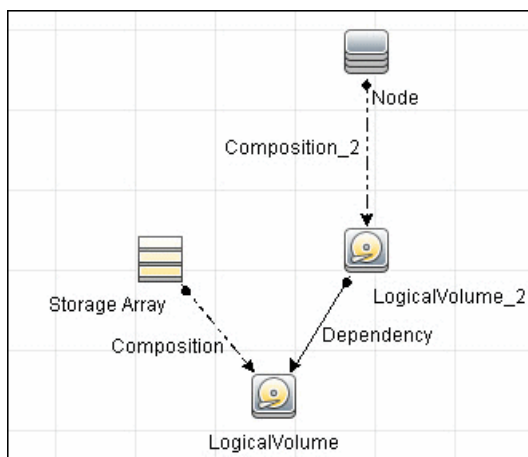
SAN topology data populate a map as shown here:



Storage Topology

Storage topology consists of relationships between Logical Volumes on a host and Logical Volumes on a Storage Array. DFM uses multiple tables to identify this relationship as shown in the query below. This view is a summary of all of the above information.

Storage topology data populate a map as shown here:



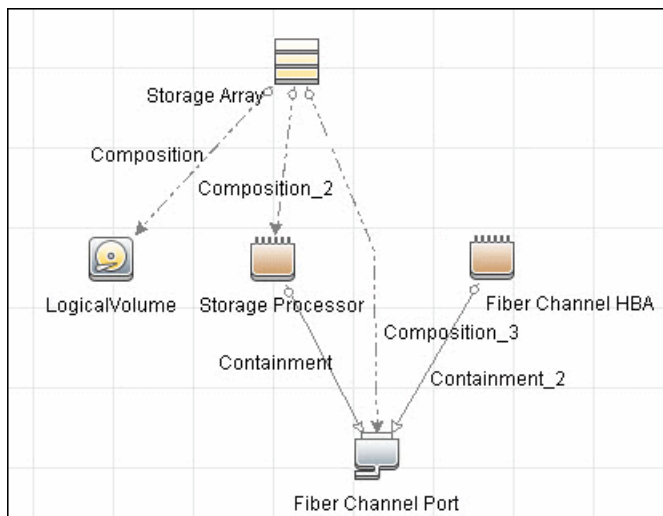
Views

The SOM package contains views that display common storage topologies. These are basic views that can be customized to suit the integrated SOM applications.

Storage Array Details

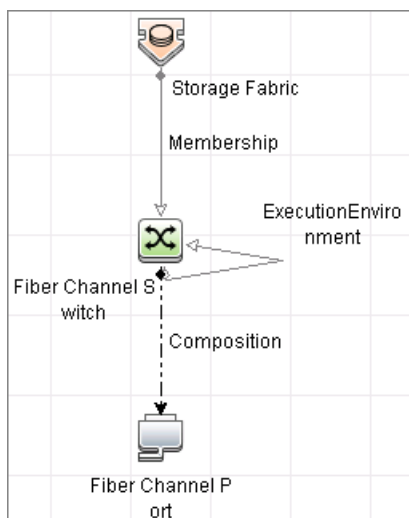
This view shows a Storage Array and its components including Logical Volumes, HBAs, Storage Processors, and Fibre Channel Ports. The view shows each component under its container Storage Array and groups Logical Volumes by CI Type.

Storage Array does not require all components in this view to be functional. Composition links stemming from the Storage Array have a cardinality of zero-to-many. The view may show Storage Arrays even when there are no Logical Volumes or Storage Processors.



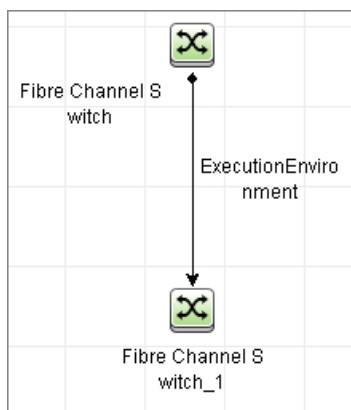
FC Switch Details

This view shows a Fibre Channel Switch and all connected Fibre Channel Ports.



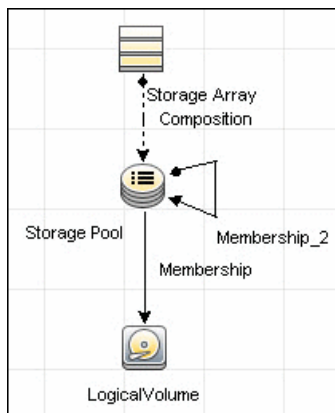
FC Switch Virtualization

FC Switch Virtualization consists of a physical switch or chassis, partitioned into multiple logical switches. Unlike Ethernet virtualization, physical ports are not shared among multiple virtual switches. Rather, each virtual switch is assigned one or more dedicated physical ports that are managed independently by the logical switches.



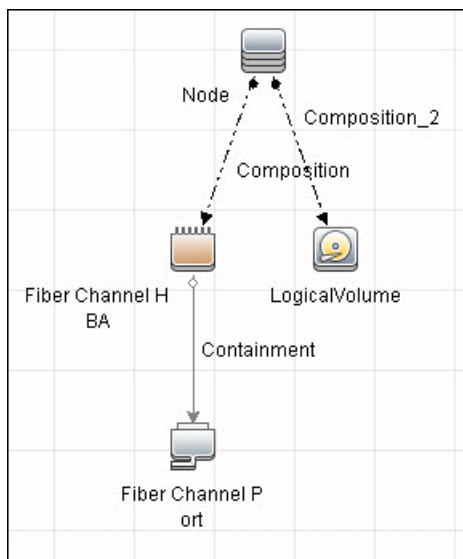
Storage Pool Details

This view shows Storage Pools with associated Storage Arrays and Logical Volumes.



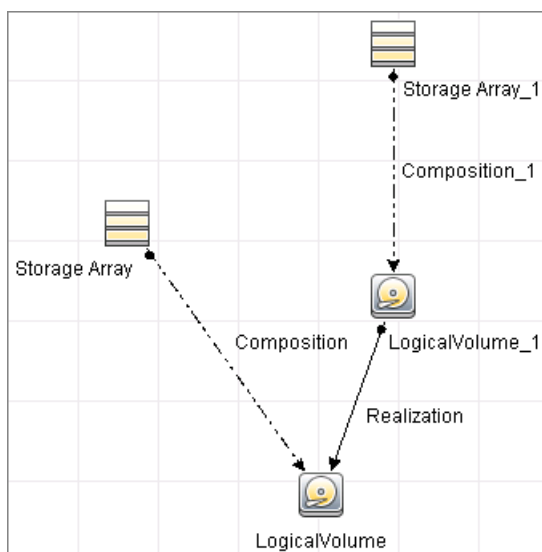
Host Storage Details

This view shows only Hosts that contain a Fibre Channel HBA or a Logical Volume. This keeps the view storage-specific and prevents hosts discovered by other DFM jobs from being included in the view.



SAN External Storage

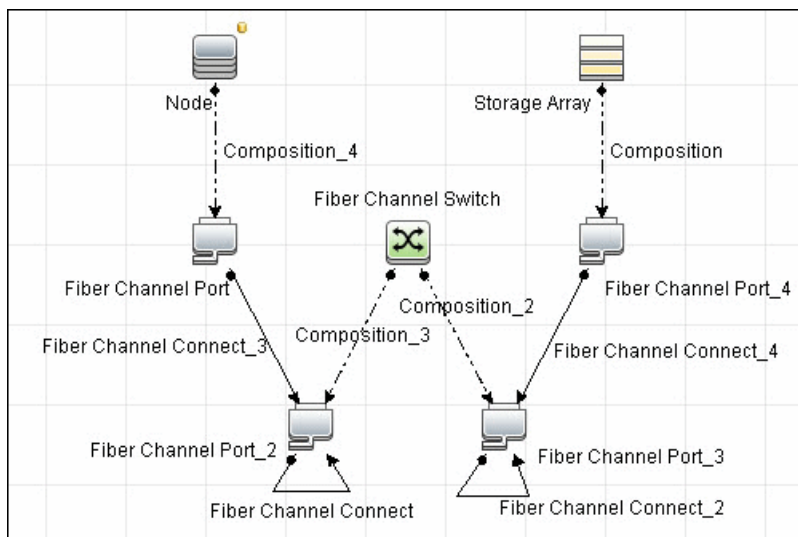
External storage configuration consists of a storage array presenting a logical volume that, in reality, belongs to another storage array. This is typically used in configurations where high-end, more expensive, front-end arrays present volumes from back-end, cheaper, storage to servers. The goal of this type of virtualization is to virtualize multiple disk arrays from different vendors, scattered over the network, into a single monolithic storage device that can be managed uniformly.



SAN Topology

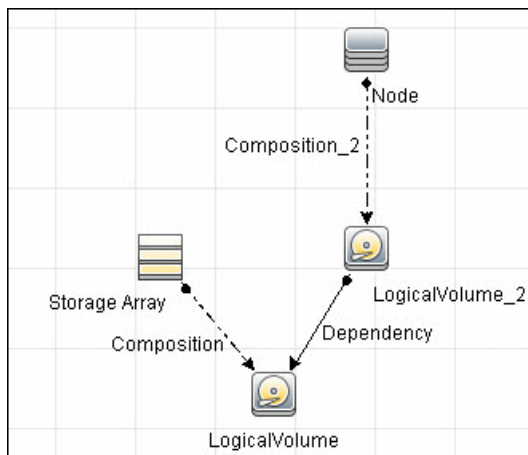
This view maps physical connections between Storage Arrays, Fibre Channel Switches, and Hosts. The view shows Fibre Channel Ports below their containers. The view groups the Fibre Channel Connect relationship CIT to prevent multiple relationships between the same nodes from appearing in the top layer.

Note: A node can be a physical host or an inferred host.



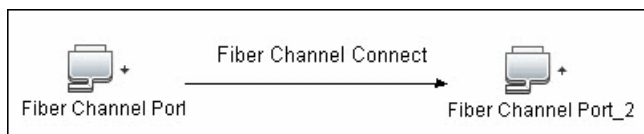
Storage Topology

This view maps logical dependencies between Logical Volumes on Hosts and Logical Volumes on Storage Arrays. There is no folding in this view.



FC Port to FC Port

This rule propagates events on a Fibre Channel Port to another connected Channel Port.



Example of HBA crashing on a Storage Array:

- The event propagates from the HBA to the Storage Array and the Logical Volumes on the Array because of the Storage Devices to Storage Array rule.
- The impact analysis event on the Logical Volume then propagates to other dependent Logical Volumes through the Logical Volume to Logical Volume rule.
- Hosts using those dependent Logical volumes see the event next because of the Host Devices to Host rule.
- Depending on business needs, you define impact analysis rules to propagate events from these hosts to applications, business services, lines of business, and so on. This enables end-to-end mapping and impact analysis using UCMDB.

Impact Analysis Rules

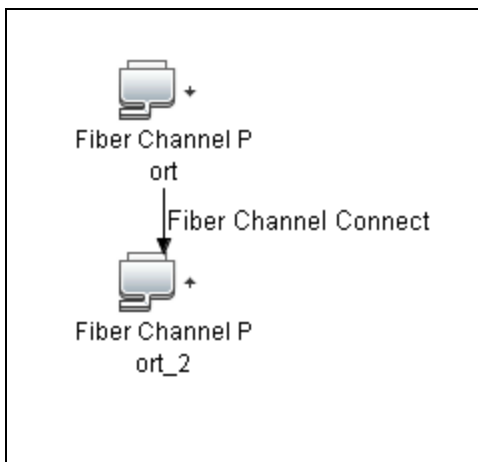
This package contains basic impact analysis rules to enable impact analysis and root cause analysis in UCMDB. These impact analysis rules are templates for more complex rules that you can define based on business needs.

All impact analysis rules fully propagate both Change and Operation events. For details on impact analysis, see "Impact Analysis Manager Page" and "Impact Analysis Manager Overview" in the *Modeling section of the UCMDB Help*.

Note: Impact analysis events are not propagated to Fibre Channel Ports for performance reasons.

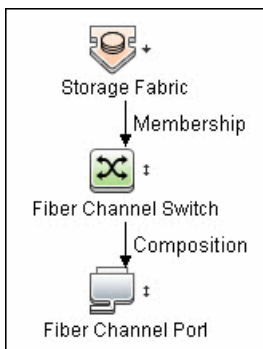
FC Port to FC Port

This impact analysis rule propagates events between related fiber channel ports.



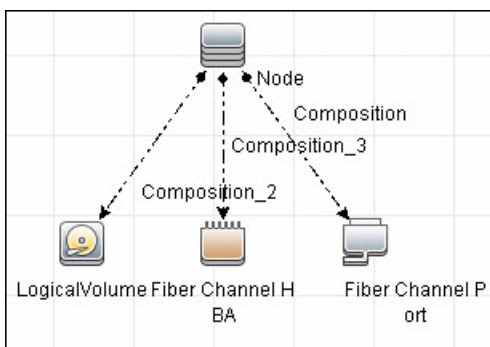
FC Switch Devices to FC Switch

This impact analysis rule propagates events from a Fibre Channel Port to and from a Switch. The event is also propagated to the associated Storage Fabric.



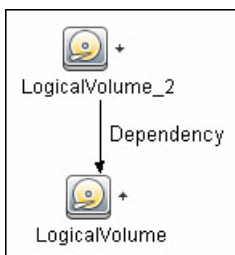
Host Devices to Host

This impact analysis rule propagates events between Fibre Channel HBAs and Hosts, and Logical Volumes on the Host.



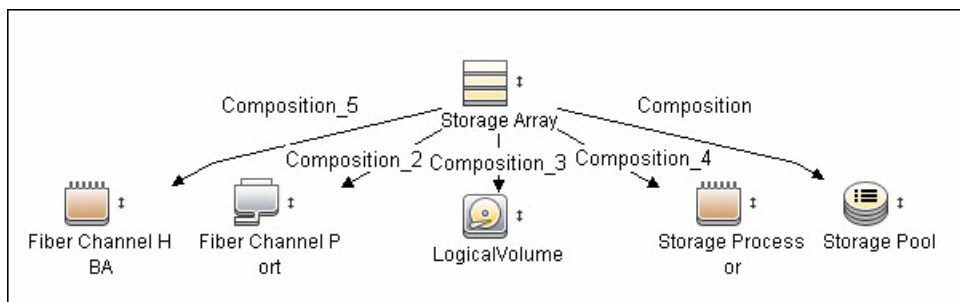
Logical Volume to Logical Volume

This impact analysis rule propagates events on a Logical Volume contained in a Storage Array to the dependent Logical Volume on the Host.



Storage Array Devices to Storage Array

This impact analysis rule propagates events between Logical Volumes, Storage Processors, Fibre Channel HBAs, and Storage Arrays.



Reports

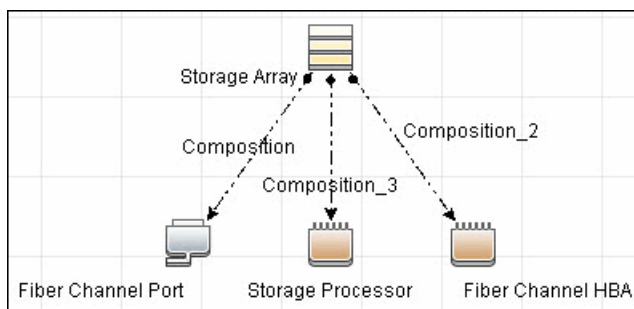
The SOM package contains basic reports that can be customized to suit the integrated storage content.

The Change Monitoring and Asset Data parameters are set on each CIT in this package to enable Change and Asset Reports in Universal CMDB. The following reports are available:

- ["Storage Array Configuration" below](#)
- ["Host Configuration" on the next page](#)
- ["Storage Array Dependency" on the next page](#)
- ["Host Storage Dependency" on page 159](#)
- ["Storage Pool Configuration" on page 159](#)

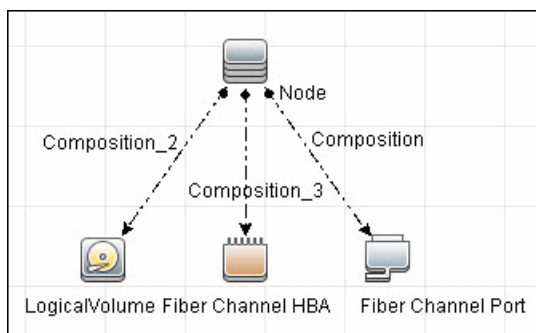
Storage Array Configuration

This report shows detailed information on Storage Arrays and its sub-components including Fibre Channel Ports, Fibre Channel Arrays, and Storage Processors. The report lists Storage Arrays with sub-components as children of the Array.



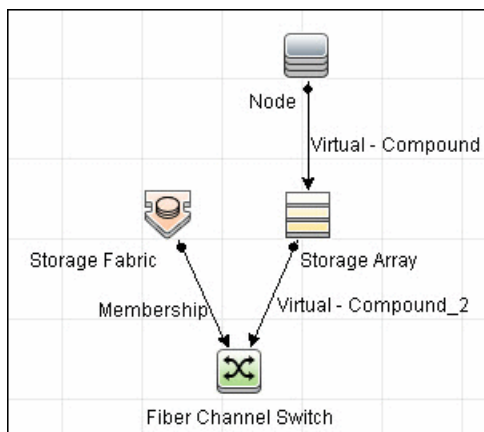
Host Configuration

This report shows detailed information on hosts that contain one or more Fibre Channel HBAs, Fibre Channel Ports, or Logical volumes. The report lists hosts with sub-components as children of the host.



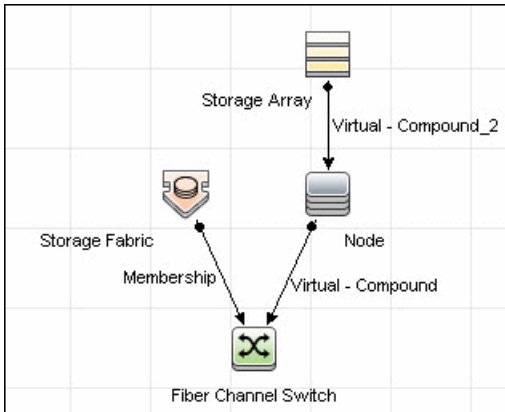
Storage Array Dependency

This report maps dependencies on a Storage Array. The report also displays information on switches connected to it.



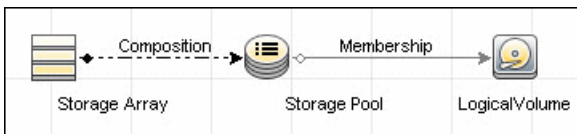
Host Storage Dependency

This report shows detailed information on storage infrastructure dependencies of a Host. The report lists hosts and dependent components.



Storage Pool Configuration

This report shows detailed information on Storage Pool configuration.



Troubleshooting the SOM Integration

This section contains useful information for troubleshooting the SOM-UCMDB Integration.

- Log files
 - Integration errors and exceptions are logged in the `RemoteProcesses.log` file, which is located in the following directory:

`<DataFlowProbeInstallDir>\runtime\log`

The default `<DataFlowProbeInstallDir>` is `<Drive>:\UCMDB\DataFlowProbe`.

- You can also view the log information from the UCMDB console. To view the log for an integration point, in the Integration Studio, select an integration point. In the integration jobs list, right click a job, and then click **View Communication Log**.
- Differences in identifiers

The data type of the FC port identifiers and the storage pool identifiers differs between SOM and UCMDB. For this reason, the SOM IDs cannot be stored directly in the UCMDB database.

The integration assigns a new identifier to each FC port and each storage pool. The integration maintains a mapping file of the new UCMDB IDs to the existing SOM IDs. This mapping file must be present on the UCMDB server for CI reconciliation to occur during the SOM data collection cycle.

The mapping file is maintained in the following location:

```
<DataFlowProbeInstallDir>\runtime\probeManager\  
discoveryResources\som\cache
```

Known Issues

This section describes the known issues of the SOM-UCMDB Integration.

- The SAN host configuration report page shows all switch, host, and storage data.
- The integration does not currently support NAS devices.
- For Switch CITs, UCMDB applies the following identification rules:

Nodes are identified using OS identifiers (for example, IP address or Net Bios name) or hardware identifiers (for example, MAC address or serial number). Two nodes that match the value for at least one OS identifier and one hardware identifier are considered to be the same node. Nodes that do not have such a match and have a conflicting value in either an OS identifier or a hardware identifier are considered to be different.

- OS identifiers include:
 - Name
 - 66% of their IP addresses
 - 66% of their IP addresses' Authoritative DNS Name
 - Net Bios name
- Hardware identifiers include:

- 66% of their interfaces' MAC addresses
- SNMP system name
- BIOS serial number
- BIOS UUID
- Serial number
- BIOS asset tag
- Two similarly identified nodes are always considered to be different entities for a mismatch of at least one of the following:
 - Operation System Family
 - Cluster Resource Group
- UCMDB displays only one port for an NPIV switch.

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 Discovery and Integrations Content Guide - Micro Focus Integrations
(Configuration Management System (CMS) Content Pack 28.00 (CP28))**

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 cms-doc@microfocus.com.

We appreciate your feedback!