



Asset Manager Generic Adapter

Software Version: 1.04

Windows® operating system

User Guide

Document Release Date: March 2018

Software Release Date: March 2018



Legal Notices

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

© 1994 - 2018 Micro Focus or one of its affiliates.

Trademark Notices

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.

This product includes an interface of the 'zlib' general purpose compression library, which is Copyright © 1995-2002 Jean-loup Gailly and Mark Adler.

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.

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 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

Overview	7
Supported Versions	8
Architecture	9
How to Integrate UCMDB and Asset Manager	11
Asset Manager Setup	11
Validate Pre-Loaded Data in Asset Manager	11
Create an Account with Administrative Rights	12
Update Asset Manager Schema	13
Activate Workflows for Population	13
Prepare Asset Manager for Parallel Push	14
Prepare Asset Manager API Zip Package	15
UCMDB Setup	16
Deploy Asset Manager Zip Package	16
Install a Database Client	21
Create an Integration Point in UCMDB	22
Out-of-Box Integration Jobs	25
Asset Manager Population Jobs	25
Asset Manager Push Jobs	26
Asset Manager Federation Configuration	33
Verify Out-of-Box Population and Push Jobs	34
Synchronize Data between UCMDB and Asset Manager	34
How to View UCMDB Data in Asset Manager	36
Nodes	36
Business Elements	36
How to View Asset Manager Data in UCMDB	37
How to Federate Asset Manager Data in UCMDB	37
Integration Jobs Configuration	39
How to Schedule Data Integration Jobs	40
Edit Data Integration Jobs	40
Standards and Concepts	43

Asset Manager Entity	43
Asset Manager Entity Definition Steps	44
Import Tables and Entity Definition	45
Out-of-Box Entity Definition	46
UCMDB TQL	47
Groovy Functions	48
Basic Functions	49
AM Population Groovy	49
AM Push Groovy	50
Utility Functions	53
Reconciliation Functions	53
Data Mapping Schema	54
Population and Federation	56
Criteria for Asset Manager Records to be Populated	56
Transformation for Asset Manager Records to be Populated	57
Reconciliation	59
Population Condition and Push Back Definition	60
Built-in attributes from AM	61
Federation Tags	61
Population Tags	62
Push and Reconciliation	62
Data Flow Architecture	62
Integration TQL Queries	63
Reconciliation Proposals	63
Asset Manager Rules and Flows	65
Mapping Attributes	65
Reconciliation	66
Target CI Validation	67
Reference Attribute	68
Attribute Reconciliation	69
Action on Delete	70
Action on Update	70
Enum Attribute	71
Ignored Attributes	71
Deletion	72

Population Deletion Configuration	72
Push Deletion Configuration	73
Installed Software / Software Utilization	74
Mapping UI	77
How to Tailor the Integration	79
How to Change Adapter Settings	79
How to Customize an Existing Mapping	82
How to Add a New Mapping to the Integration	85
How to Populate Asset Manager Contract	87
How to Set up Federation	94
How to set up integration in a multi-tenant environment	95
Quick start configuration	95
Solution design	96
Tailoring	98
Performance Tuning and Best Practice	100
Chunk Size	100
Time out	100
TQL Tailoring	101
Push Mapping Simplification	101
Global ID	102
Population Mapping Optimization	103
Push Mode	104
Push Mode Comparison	104
How to Set Push Mode	105
Non-parallel Push Mode	106
Architecture	106
Parameters	106
Multi-threading Push Mode	107
Architecture	107
Parameters	107
Multi-processing Push Mode	108
Architecture	108
Parameters	109
Tuning	109

Troubleshooting	110
Differences between Push Adapter and Generic Adapter	113
Differences between Old Push Adapter and Push in Generic Adapter	113
Mapping Schema Changes	113
File Renaming and reorganizing	113
Asset Manager Entity Reference in Mapping Script and Reconciliation	114
Differences between Old Population Adapter and Population in AM Generic Adapter	114
No need to import views for AM database	114
Population mapping needs to specify TQL	115
Attributes conversion and discrimination	115
Frequently Asked Questions	117
Troubleshooting and Limitations	121
Logs	123
AM API Logs	124

Overview

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

You can use the Asset Manager Generic Adapter to automate the creation and update of Asset and Portfolio information in Asset Manager through data push. This ensures that Asset Manager is kept up-to-date with real, accurate, and discovered data in your environment. On the other hand, it automates the creation and update of Node and Asset information in UCMDB by populating asset data from Asset Manager to UCMDB. It takes advantage of the built-in procurement and retirement processes of Asset Manager to enable UCMDB to manage IT assets which are not in operation and are undiscoverable.

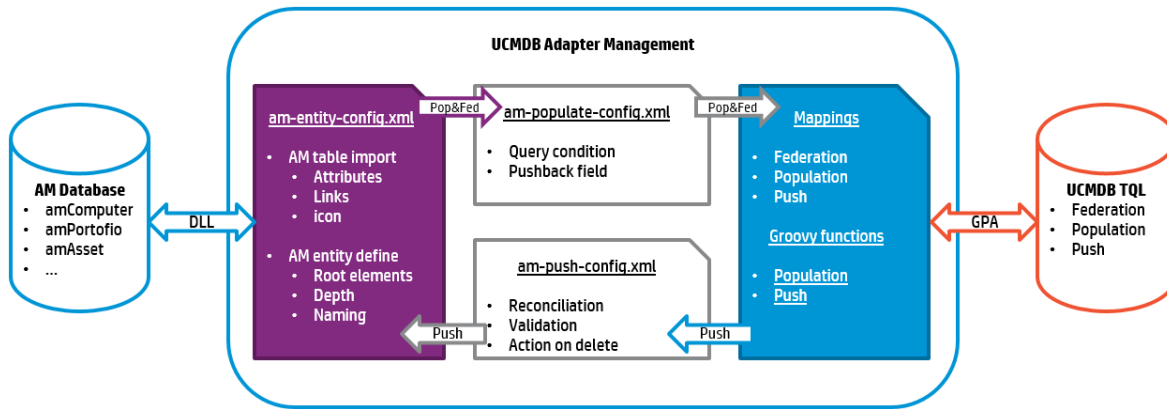
Note: This Integration replaces the Connect-It Scenarios used for synchronizing hardware and software information from DDML 9.3x (and earlier versions) to Asset Manager. Also, this integration replaces the Connect-It Scenarios used for synchronizing Business Services and Business Applications from UCMDB to Asset Manager.

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

Supported Versions

Refer to the *Asset Manager Push Adapter and Generic Adapter Support Matrix* on [Micro Focus Marketplace](#).

Architecture



Asset Manager Generic Adapter is a new version of the AM adapter for exchanging data between Asset Manager and UCMDB. It bundles 3 core features of the AM and UCMDB integration in a single adapter: population, federation, and push. The AM Generic Adapter is built on top of the Generic Adapter (GA), which is a unified framework for external systems to integrate with UCMDB. The GA framework provides a graphic user interface for integration developers and administrators to ease the creation and customization of data mapping files. In the graphic mapping UI, you can easily map attributes and relationships between AM Entity and UCMDB CI.

AM Entity is a new layer introduced in the AM Generic Adapter. For more information, see ["Asset Manager Entity" on page 43](#).

Population and Federation

The AM Generic Adapter retrieves data from the AM database through the AM native APIs and creates AM entity objects with regards to the AM entity structure defined in `am-entity-config.xml`. Conditions specified in the file `am-populate-config.xml` filter the data retrieved from AM for a given entity.

The AM entity objects are converted to UCMDB CI structure according to the mapping script defined in mapping files. The Generic Adapter framework transmits the mapped CIs to the UCMDB Data In engine for processing.

Push

UCMDB stores its information using CIs. The integration chooses which data to be pulled 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 Generic Adapter framework allows easy mapping of the data from the UCMDB data model into the Asset Manager Data Model. It also allows transfer of this converted data into the AM Generic Adapter.

The AM Generic Adapter connects to the AM database through the AM native APIs to reconcile, push, and handle the complex logic needed to synchronize data into Asset Manager.

How to Integrate UCMDB and Asset Manager

This section describes how to integrate UCMDB and Asset Manager with Asset Manager Generic Adapter.

Asset Manager Setup

To set up Asset Manager for the integration, following the steps described in this section.

Validate Pre-Loaded Data in Asset Manager

For the integration to succeed, some basic data is required to exist in the Asset Manager database.

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

For hardware synchronization the required data is:

- Shared Data
- UNSPSC Product Classification
- Portfolio – Line-of-business data
- Virtualization – Line-of-business data
- Business services management – Line-of-business data

For software synchronization the required data is:

- Software Asset Management – Line-of-business data

Create an Account with Administrative Rights

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

The details of the default Administrator user are:

- User: Admin
- Password: <empty>

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

1. Log on to Asset Manager as the Administrator.
2. Go to **Organization Management > User actions > Add a user**.
 - a. In **ID #**, type: Integration-Admin.
 - b. In **Name**, type: Integration-Admin.
 - c. In **First**, type: Integration-Admin.
 - d. Click **Next**.
 - e. Click **Next**.
 - f. Click **Finish**.
3. Go to **Organization Management > Organization > Employees**.
4. Select the newly created user and click the **Profile** tab.
5. In **User name**, type: Integration-Admin.
6. In **Password**, type: <A password you would like to use>.
7. In the **Password Administration** pane, ensure that **Never Expires** is selected.
8. In the **Profile** pane, ensure **Administration rights** is selected.
9. Select the **Can create or modify shared data** check box.
10. Click **Modify**.

Update Asset Manager Schema

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

To fix this issue, we recommend that you change the Asset Manager Column Sizes to the values shown in the following table of Asset Manager attributes.

Table	Name	Default Max Length	New Value
amAsset	SerialNo	128	250
amModel	Name	128	250
amSoftInstall	Field1	26	255
amBrand	Name	128	250
amEmplDept	UserName	128	200
amEmplDept	UserDomain	128	200

Note: The new values in the table are only a suggestion, and you may need to change them according to actual data per customer use case.

Note: For DB2, the default table space page size of 4K may be too small in some cases. We recommend that you use 8K or higher larger.

Activate Workflows for Population

In Asset Manager, the amComputer is an overflow table to the amPortfolio table. Part of its attributes are stored in the amPortfolio table. For instance, seAssignment. Some other attributes are distributed to other tables, such as amAsset. By default, if you make changes to any other tables than amComputer, its Last Modified Time does not change. As a result, when you run a delta sync on a job to populate Node, the changes will not be captured. To solve this problem, you need to activate two workflows in Asset Manager. To do this, follow these steps.

1. Log on to the Asset Manager client.
2. On the Tools menu, point to **Workflow**, click **Workflow schemes**.
3. Locate the **Update dtRecCreation** (SQL name: sysCoreUpCrTime) workflow and the **Update last modify time** (SQL name: sysCoreUpMdifTime) workflow.
4. On the **General** tab, empty the **End field of the Validity** pane.
5. Save the changes.

Prepare Asset Manager for Parallel Push

Enabling Parallel Push significantly improves the performance of the push. However, some advanced preparations are necessary. Different actions are needed for different database types, as shown in the following table.

Database	Action
DB2	Mandatory: Follow Eliminating locks and deadlocks in the Asset Manager Tuning Guide.
Oracle	Optional: Follow Eliminating locks and deadlocks in the Asset Manager Tuning Guide.
SQL Server	Mandatory: Follow the steps below.

Mandatory steps for SQL Server

1. Alter the SQLServer Schema isolation level. To do this, execute the following command on the database, replacing <AMSchema> with the real schema name.

```
ALTER DATABASE <AMSchema> SET READ_COMMITTED_SNAPSHOT ON  
ALTER DATABASE <AMSchema> SET ALLOW_SNAPSHOT_ISOLATION ON  
GO
```

Note: If the execution takes too long, you may need to disconnect all connections to the database. One possible way is to restart the database service, then execute the command. If you restart SQL Server and an Integration Point has already been created in UCMDB, you should restart the UCMDB Probe as well to avoid the issue of dead connections.

2. Alter Asset Manager database options:

- a. Open the Asset Manager client and connect to the appropriate database schema.
 - b. Navigate on the top menus to **Administration > Database options**.
 - c. For the option **Sql Server specifics'** **Isolation command before starting a write transaction**, change the current value to **set transaction isolation level snapshot**.
3. Create a table for each of the following counters.

To do this, follow the instructions in the *Eliminating locks and deadlocks* chapter of the *Asset Manager Tuning Guide*.

- amAsset_AssetTag
- amBrand_BarCode
- amModel_BarCode
- amModel_ModelRef
- amAssignment_Code
- amEmplDept_BarCode
- amComputer_Group
- amComputer_Domain
- amComputer_Vm
- amComputer_MD
- amComputer_Name
- amSoftInstall_Code
- amMonitor_Serial

Prepare Asset Manager API Zip Package

In order for the adapter to connect to the appropriate Asset Manager version, you must supply the Data Flow Probe with the appropriate Asset Manager API DLLs and Jars.

For Asset Manager 9.60 and later versions, you can obtain the installed Asset Manager APIs from the %Asset Manager installation%\ integrations\ucmdb\ folder.

For versions before 9.60, follow these steps:

1. Copy the files below:
 - <Asset Manager Installation folder>\x64*.dll
 - <Asset Manager Installation folder>\websvc\lib*.jar
2. Create a package named AMGenericAdapterAPI_<AM Version Number>.zip. For example, for

version 9.41 the package name is AMGenericAdapterAPI_9.41.zip.

3. Paste the copied files to:

```
AMGenericAdapterAPI_<AM Version  
Number>.zip\discoveryResources\AMGenericAdapter\amVersion\<AM Version Number>
```

For example, for version 9.41, the path is:

```
AMGenericAdapterAPI_9.41.zip\discoveryResources\AMGenericAdapter\amVersion\9.41
```

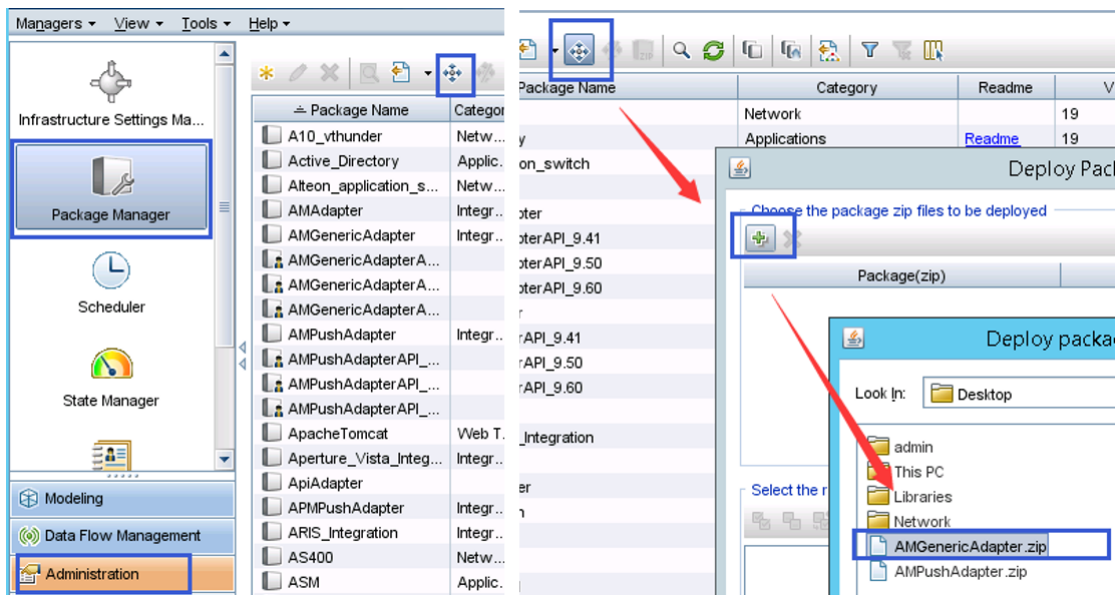
UCMDB Setup

To set up UCMDB for the integration, follow the steps detailed in this section.

Deploy Asset Manager Zip Package

This section describes how to deploy the AMGenericAdapter.zip package.

1. Open the UCMDB Applet UI.
2. Click **Administration** and then click **Package Manager**.
3. Click **Deploy packages to server (from local disk)**.



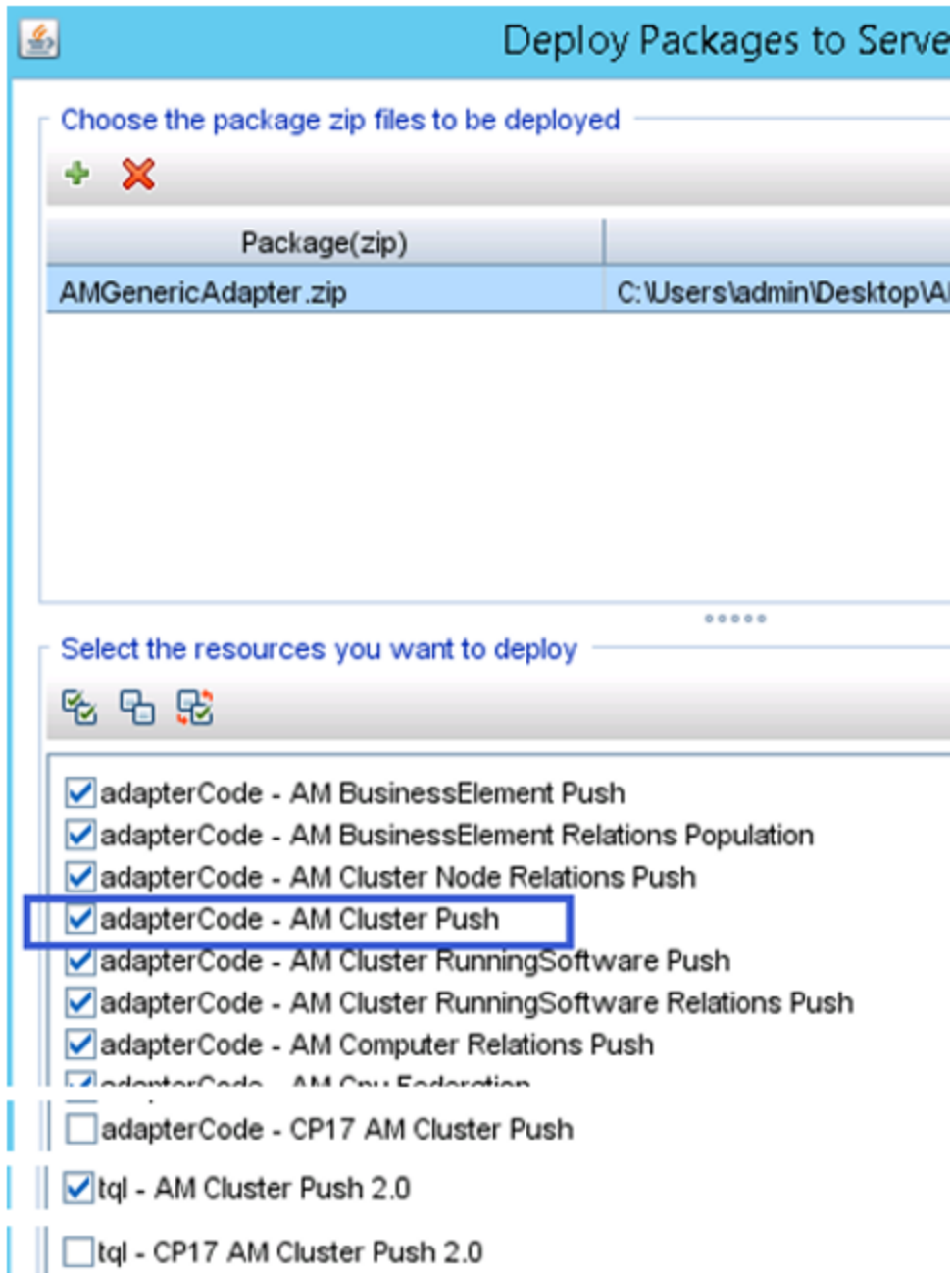
4. If you use UCMDB **CP 18 or a later version**, you must uncheck the mappings and TQL files that start with "CP17". By default, all files are checked.

Note: "AM Cluster Push" contains the new type "oracle_data_guard" and it is supported from CP18. The "CP17" TQL files and mappings do not contain "oracle_data_guard".

The tables list the unchecked files.

TQL name	TQL path
CP17 AM Cluster Push 2.0.xml	AMGenericAdapter\1.00\package\tql\Integration\AMGenericAdapter\push\CP17 AM Cluster Push 2.0.xml

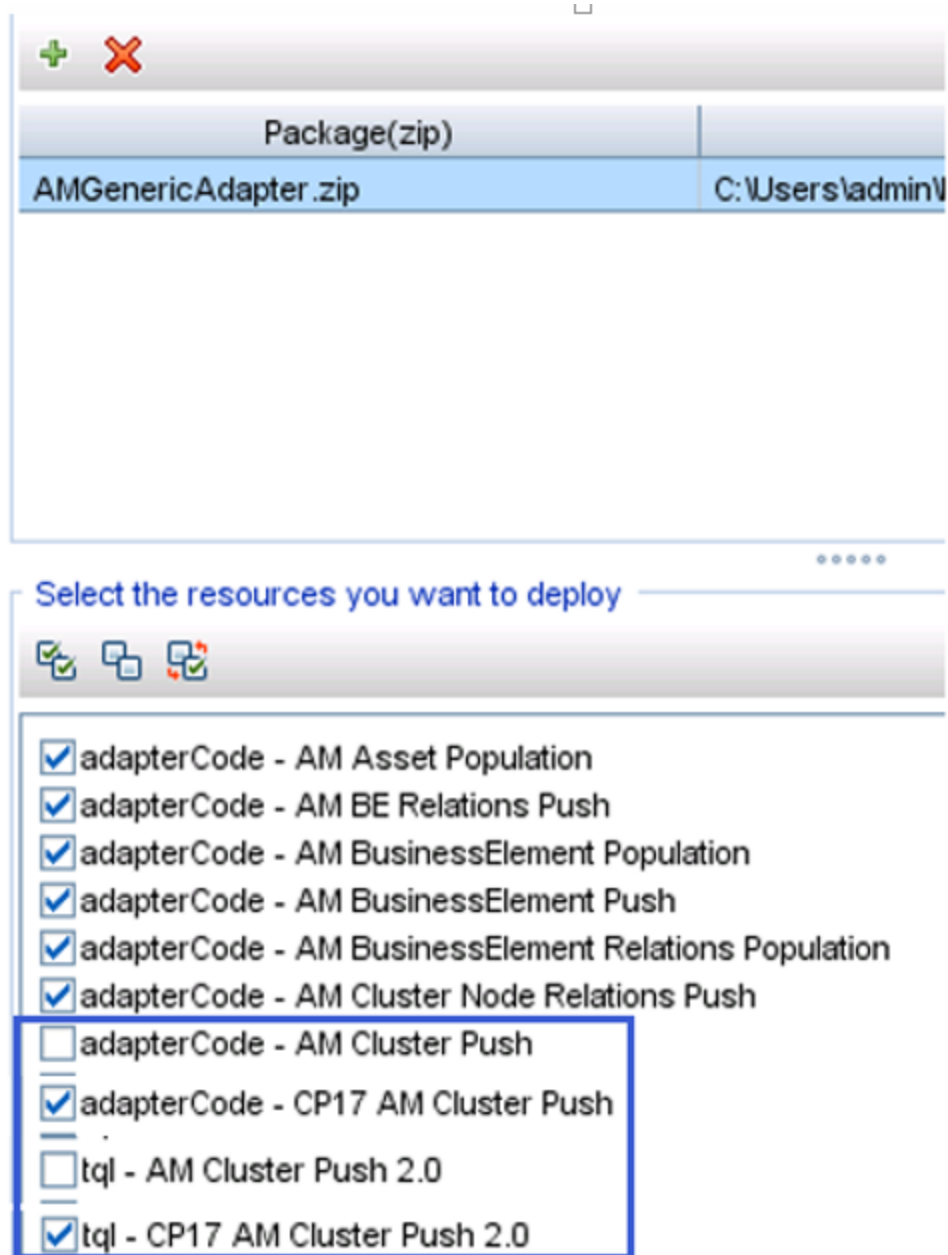
Mapping name	Mapping path
CP17 AM Cluster Push.xml	AMGenericAdapter\1.00\package\adapterCode\AMGenericAdapter\mappings\push\hw\CP17 AM Cluster Push.xml



5. If you use UCMDB **CP 17 or an earlier version**, you must uncheck the mappings and TQL files that contain the unsupported type "oracle_data_guard".

TQL name	TQL path
AM Cluster Push 2.0.xml	AMGenericAdapter\1.00\package\tql\Integration\AMGenericAdapter\push\AM Cluster Push 2.0.xml

Mapping name	Mapping path
AM Cluster Push.xml	AMGenericAdapter\1.00\package\adapterCode\AMGenericAdapter\mappings\push\hw\AM Cluster Push.xml



Package(zip)

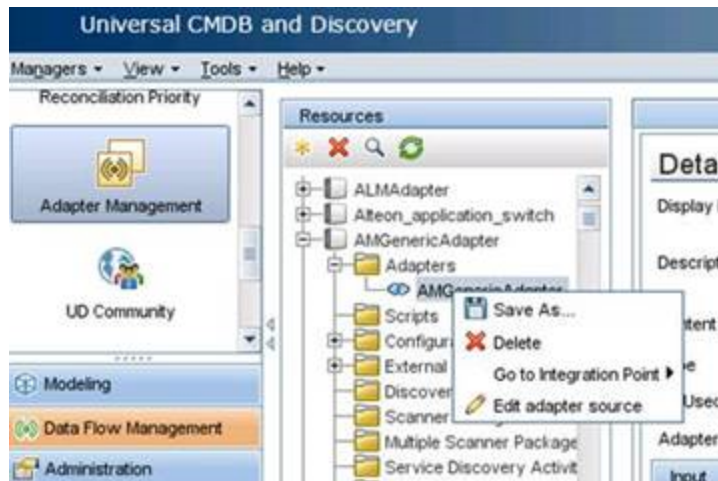
AMGenericAdapter.zip	C:\Users\admin\...
----------------------	--------------------

Select the resources you want to deploy

- ☒ adapterCode - AM Asset Population
- ☒ adapterCode - AM BE Relations Push
- ☒ adapterCode - AM BusinessElement Population
- ☒ adapterCode - AM BusinessElement Push
- ☒ adapterCode - AM BusinessElement Relations Population
- ☒ adapterCode - AM Cluster Node Relations Push
- ☐ adapterCode - AM Cluster Push
- ☒ adapterCode - CP17 AM Cluster Push
- ☐ tql - AM Cluster Push 2.0
- ☒ tql - CP17 AM Cluster Push 2.0

6. Click **Deploy**.

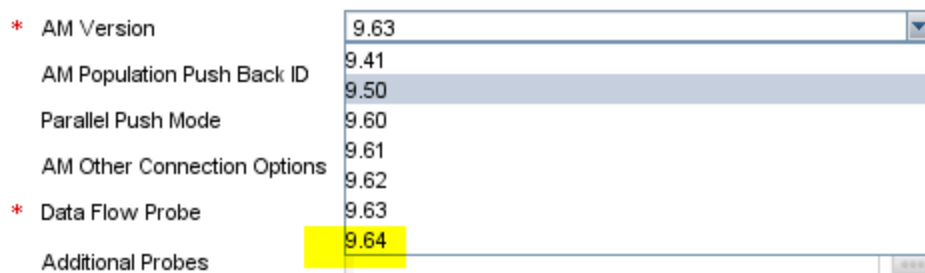
7. If you want the Generic Adapter to support a new Asset Manager version which is not on the version list, edit the adapter source to support the new Asset Manager version. To do this, follow these steps.
 - a. Go to **Data Flow management > Adapter Management > AMGenericAdapter > Adapters**, right-click **AMGenericAdapter**, and then click **Edit adapter source**.



- b. Open the adapter source file and find the parameter for AM version, add the new Asset Manager version in the **valid-values** list and save, for example, 9.64.

```
<parameter name="amVersion" description="Remote AM Version" type="string"
display-name="AM Version" valid-values="9.41;9.50;9.60;9.61;9.62;9.63;<new
version here>" mandatory="true" order-index="16">9.41</parameter>
```

- c. You can now configure the new version using the drop-down list.



Note: When you upgrade GA Adapter from CP15, you need to remove all federation TQLs of CP15 from UCMDB manually. That is because the Federation TQLs of CP15 do not follow the new naming convention for file name and TQL name, the new package cannot replace the old federation TQLs, thus causing adapter error.

CP15 Federation TQL Name Style	New Federation TQL Name Style
AM Federation cpu_12 2.0	AM Federation Cpu 2.0
AM Federation cpu_node 2.0	AM Federation Cpu_Node 2.0
AM Federation disk_device_12 2.0	AM Federation DiskDevice 2.0
AM Federation disk_device_node 2.0	AM Federation DiskDevice_Node 2.0
AM Federation file_system_12 2.0	AM Federation FileSystem 2.0
AM Federation file_system_node 2.0	AM Federation FileSystem_Node 2.0
AM Federation installed_software_12 2.0	AM Federation InstalledSoftware 2.0
AM Federation installed_software_node 2.0	AM Federation InstalledSoftware_Node 2.0
AM Federation printer_12 2.0	AM Federation Printer 2.0
AM Federation printer_node 2.0	AM Federation Printer_Node 2.0
AM Federation logical_volume_12 2.0	AM Federation LogicalVolume 2.0
AM Federation logical_volume_node 2.0	AM Federation LogicalVolume_Node 2.0


Install a Database Client


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

Database	Client Software
SQL Server	None required.

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

Create an Integration Point in UCMDB

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

Field	Required	Description
Integration Name	Yes	Type the name (unique key) of the integration point.
Integration Description	No	Type a description of the current integration point.
Adapter	Yes	Select Micro Focus Software Products > Asset Manager > Asset Manager Generic Adapter
Is Integration Activated	Yes	Select this option to indicate the integration point is active.
Hostname/IP	Yes	Type the hostname or IP Address of the Asset Manager database.
DB Type	Yes	Select the database type your Asset Manager schema is located on.
DB Port	Yes	Type the communication port of the Asset Manager Data Base.
DB Name/SID	Yes	<ul style="list-style-type: none"> ◦ DB2: type in the name of the Database Alias you defined in the database connection. ◦ Oracle: type the service name. ◦ SQL Server: type the name of the schema.
DB Owner Name	Yes	Enter the owner of the AM database schema.
Credentials ID	Yes	<p>Select Asset Manager Protocol. To create a new protocol, click the  button. Under Asset Manager Protocol complete:</p> <ul style="list-style-type: none"> ◦ Asset Manager User Name: An AM administrator's user name. ◦ Asset Manager Password: An AM administrator's password. ◦ DB User Name: The AM database user's name. ◦ DB Password: The AM database user's password.
AM Version	Yes	Select the version of Asset Manager this integration point is to connect to.
AM Population Push Back ID	No	Select True to allow the adapter to push the Global Id of UCMDB CI back to the corresponding AM entity. For more information, see "Population Condition and Push Back Definition" on page 60.

Field	Required	Description
AM Other Connection Options	No	Options can be used in the amdb.ini file. See Asset Manager Installation Guide, Chapter .ini, .cfg, and .res files > Amdb.ini File Entries.
Parallel Push Mode	No	Select to allow parallel (multi-threaded) data push to Asset Manager. This improves performance. Note that you must configure SQL Server & DB2 to support parallel push. See "Prepare Asset Manager for Parallel Push" on page 14 .
Data Flow Probe	Yes	Select the name of the Data Flow Probe used to execute the synchronization from.
Additional Probes	No	Select additional probes to use when pushing to AM in order to increase redundancy.
Default owner name	No	<p>The name of the owner tenant that should be assigned to the federated or populated CIs and relationships.</p> <p>This field does not take effect for push operations.</p> <p>Note:</p> <ul style="list-style-type: none"> ◦ This field is displayed when creating a federation or population type integration point in a multi-tenancy environment only. ◦ If no owner tenant is specified, but the Data Flow Probe selected for the integration point has an owner tenant, then the Data Flow Probe owner tenant is assigned to all discovered CIs. ◦ The System Owner Tenant is assigned when: <ul style="list-style-type: none"> • No owner tenant is specified, and no owner tenant is defined on the Data Flow Probe. • The data source is not a multi-tenancy environment. ◦ This field only appears in a Multi-Tenant Enabled UCMDb.

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

6. Click **OK**.

The integration point is created and its 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 *Universal CMDB Data Flow Management Guide*.

Out-of-Box Integration Jobs

AM Generic Adapter is shipped with out-of-box jobs for population and push between Asset Manager and UCMDB. Integration job is driven by UCMDB TQLs.

To access these TQLs, follow these steps.

1. Log on to UCMDB as an administrator.
2. Go to **Modeling > Modeling Studio > Resources**, and then select **Queries** from the **Resource Type** drop-down list.
3. Expand **Root/Integration/AMGenericAdapter**. There are 3 sub folders: federation, population and push.
 - All queries under the federation folder federate CI data from AM to UCMDB.
 - All queries under the population folder manage CI data synchronization from AM to UCMDB.
 - All queries under the push folder manage CI data synchronization from UCMDB to AM.

Asset Manager Population Jobs

In the following table, all TQLs which are included in the default population job are listed. For each TQL query, it describes the source AM entity type and the target CI types converted by the out-of-box mapping XMLs. In the description column, it contains a summary of the TQL query, the out-of-box mapping XML to convert the data, and the other data populations it depends on, which need to be run beforehand.

TQL Query	AM Entity	UCMDB CI Type	Description
AM Location Population 2.0	Location	Location <ul style="list-style-type: none">• Location	Populates location. Mapping XML: AM Location Population.xml
AM Node Population 2.0	ITEquipment	Node <ul style="list-style-type: none">• Location• Asset	Populates Node with Asset and Location. Mapping XML: AM Node Population.xml
AM Interface	NetworkCard	Interface	Populates Interface with

TQL Query	AM Entity	UCMDB CI Type	Description
Population 2.0		<ul style="list-style-type: none"> Node IpAddress 	IpAddress and Node relations. Depends On: AM Node Population 2.0 Mapping XML: AM Interface Population.xml
AM BusinessElement Population 2.0	Asset	BusinessElement <ul style="list-style-type: none"> Asset 	Populate Business Element with Asset. Mapping XML: AM BusinessElement Population.xml
AM BusinessElement Relations Population 2.0	Client_Resource_Relationship	BusinessElement <ul style="list-style-type: none"> BusinessElement Node 	Populate Business Element relationship with Business Element and Node. Depends On: <ul style="list-style-type: none"> AM Node Population 2.0 AM BusinessElement Population 2.0 Mapping XML: AM BusinessElement Relations Population.xml
AM Printer Population 2.0	Portfolio_Printer	Printer <ul style="list-style-type: none"> Node 	Populates Printer with the relation to Node. Depends On: AM Node Population 2.0 Mapping XML: AM Printer Population.xml

Asset Manager Push Jobs

The below table lists all TQLs for data push contained in the out-of-box AM Generic Adapter. Some of them are included in the default push job. For each TQL query, it describes the source CI type and the target AM Entity type converted by the out-of-box mapping XMLs. In the description column, it contains

a summary of the TQL query, the out of the box mapping XML to convert the data, and the other data pushes it depends on, which need to be run beforehand.

TQL Query	AM Entity	UCMDB CI Type	Description
AM Node Push 2.0	ITEquipment <ul style="list-style-type: none"> PhysicalDrives NetworkCards LogicalDrives ExtensionCards AddOn (Printer, Monitor) 	Node <ul style="list-style-type: none"> Asset CPU FileSystem LogicalVolume HardwareBoard DisplayMonitor DiskDevice InventoryScanner IpAddress Virtual Host Resource PhysicalPort VMWare Host Resource Printer Interface 	<p>Pushes nodes (Computers, Network Devices, etc.). Also pushes IPs, Interfaces, Disk Devices, Physical Ports, Hardware Boards, Display Monitors, CPUs, Printers, Inventory Scanners, File Systems, and Assets.</p> <p>Minimal attributes for pushing a Node:</p> <ul style="list-style-type: none"> Serial Number Vendor or Discovered Vendor Model or Discovered Model Node Role <p>Note: These are required values, and depend on the capability of the data source to report them.</p> <p>Mapping XML: AM Node Push.xml</p>
AM Business Element Push 2.0	Asset	BusinessElement	<p>Pushes Business Applications, Business Services and Business Infrastructure CIs.</p> <p>Mapping XML: AM BusinessElement Push.xml</p>
AM Business Element Relations Push 2.0	Client_Resource_Relationship	BusinessElement <ul style="list-style-type: none"> BusinessElement Node 	<p>Pushes relationships between Business Elements (pushed by AM Business Element Push Query) to other business elements or to nodes.</p> <p>Depends On:</p>

TQL Query	AM Entity	UCMDB CI Type	Description
			<ul style="list-style-type: none"> AM Business Element Push 2.0 AM Node Push 2.0 <p>Note: If the Asset Manager Business Element is not related to computers, AM Business Element Relations Push does not depend on AM Computer Push.</p> <p>Mapping XML: AM BE Relations Push.xml</p>
AM Host Server And Running VM Relations Push 2.0	Client_Resource_Relationship	Node <ul style="list-style-type: none"> Node 	<p>Pushes relations between Host and Guest (Virtualized) operating systems.</p> <p>Depends On: AM Node Push 2.0</p> <p>Mapping XML: AM Host VM Relations Push.xml</p>
AM Host Server And Running Solaris VM Relations Push 2.0	Client_Resource_Relationship	Unix <ul style="list-style-type: none"> Unix 	<p>Pushes relations between Solaris Host and Guest (Virtualized) operating systems.</p> <p>Depends On: AM Node Push 2.0</p> <p>Mapping XML: AM Host VM Solaris Relations Push.xml</p>
AM Host Server And Running LPAR VM Relations Push 2.0	Client_Resource_Relationship	Unix <ul style="list-style-type: none"> Unix 	<p>Pushes relations between Host and Guest (virtualized) systems of LPAR type.</p> <p>Depends On: AM Node Push 2.0</p> <p>Mapping XML:</p>

TQL Query	AM Entity	UCMDB CI Type	Description
			AM Host VM Lpar Relations Push.xml
AM Computer Relations Push 2.0	Client_Resource_Relationship	Node <ul style="list-style-type: none"> Node 	<p>Pushes relations between Computers to any node (Computers, Network Devices, etc.).</p> <p>Depends On: AM Node Push 2.0</p> <p>Mapping XML: AM Computer Relations Push.xml</p>
AM Net Device Relations Push 2.0	Client_Resource_Relationship	Node <ul style="list-style-type: none"> Node 	<p>Pushes relations between Network Devices and Network Devices.</p> <p>Depends On: AM Node Push 2.0</p> <p>Mapping XML: AM NetDevice Relations Push.xml</p>
AM Installed Software Push 2.0	SoftwareInstallation	InstalledSoftware <ul style="list-style-type: none"> Node InventoryScanner 	<p>Pushes Installed Software CIs. These Installed Software CIs will be normalized in AM after push.</p> <p>Depends On: AM Node Push 2.0</p> <p>Mapping XML: AM InstalledSoftware Normalize Push.xml</p>
AM Software Utilization Push 2.0	SoftwareInstallation	UserSoftwareUtilization <ul style="list-style-type: none"> Node InventoryScanner InstalledSoftware 	<p>Pushes Software Utilization CIs. These Software Utilization CIs will be normalized in AM after push.</p> <p>Depends On: AM Node Push 2.0</p> <p>Mapping XML:</p>

TQL Query	AM Entity	UCMDB CI Type	Description
			AM UserSoftwareUtilization Normalize Push.xml
AM Installed Software Normalized Push 2.0	SoftwareInstallation	InstalledSoftware <ul style="list-style-type: none"> • Node • InventoryScanner 	<p>Pushes Installed Software CIs. These Installed Software CIs should already be recognized by normalization technology in UCMDB, for example, BDNA. They will not be normalized in AM after push.</p> <p>Depends On:</p> <p>AM Node Push 2.0</p> <p>Mapping XML:</p> <p>AM InstalledSoftware Non-normalize Push.xml</p>
AM Software Utilization Normalized Push 2.0	SoftwareInstallation	UserSoftwareUtilization <ul style="list-style-type: none"> • Node • InventoryScanner • InstalledSoftware 	<p>Pushes User_Software_Utilization CIs. These Installed Software CIs should already be recognized by normalization technology in UCMDB, for example, BDNA. They will not be normalized in AM after push.</p> <p>Depends On:</p> <p>AM Node Push 2.0</p> <p>Mapping XML:</p> <p>AM UserSoftwareUtilization Non-normalize Push.xml</p>
AM Software Hypervisor Push 2.0	SoftwareInstallation	Hypervisor <ul style="list-style-type: none"> • Node 	<p>Pushes Hypervisor Installed Software.</p> <p>Depends On:</p> <p>AM Node Push 2.0</p> <p>Mapping XML:</p> <p>AM Hypervisor Push.xml</p>
AM Software Solaris Push 2.0	SoftwareInstallation	Unix <ul style="list-style-type: none"> • Unix 	Pushes Solaris Installed Software.

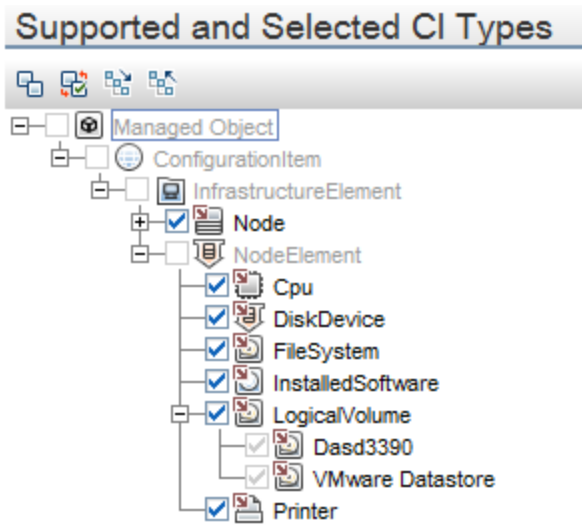
TQL Query	AM Entity	UCMDB CI Type	Description
			Depends On: AM Node Push 2.0 Mapping XML: AM Solaris Zone Push.xml
AM Oracle LMS Push 2.0	MonitoredApplication	Oracle <ul style="list-style-type: none"> Node InstalledSoftware AuditDocuement 	Pushes the Oracle Running Software and its Oracle LMS data. Depends On: AM Node Push 2.0 Mapping XML: AM Oracle LMS Push.xml
AM Cluster Push 2.0	Cluster	Cluster <ul style="list-style-type: none"> Asset 	Pushes Cluster CIs. It is not included in the default push job. Mapping XML: AM Cluster Push.xml
AM Cluster Node Relations Push 2.0	Cluster_ Component_ Relationship	Cluster <ul style="list-style-type: none"> Node 	Pushes cluster relations between Computer Elements and Cluster. Depends On: AM Node Push 2.0 AM Cluster Push 2.0 Mapping XML: AM Cluster Node Relations Push.xml
AM Cluster Runningsoftware Push 2.0	MonitoredApplication	Runningsoftware <ul style="list-style-type: none"> Node Cluster 	Pushes Cluster Runningsoftware CIs. By default, it is not included in the push job. Depends On: <ul style="list-style-type: none"> AM Node Push 2.0 AM Cluster Push 2.0 Mapping XML:

TQL Query	AM Entity	UCMDB CI Type	Description
			AM Cluster RunningSoftware Push.xml
AM Cluster Runningsoftware Relations Push 2.0	Cluster_ Component_ Relationship	Cluster <ul style="list-style-type: none"> Runningsoftware 	<p>Pushes cluster RunningSoftware relations between Computer Elements and Cluster. By default, it is not included in the push job.</p> <p>Depends On:</p> <p>AM Cluster Push 2.0</p> <p>AM Cluster Runningsoftware Push 2.0</p> <p>Mapping XML:</p> <p>AM Cluster RunningSoftware Relations Push.xml</p>
AM Resource Pool Push 2.0	ResourcePool	<ul style="list-style-type: none"> Resource Pool Node 	<p>Pushes IBM processor pool Data.</p> <p>Depends on:</p> <ul style="list-style-type: none"> AM Node Push 2.0 <p>Mapping XML:</p> <ul style="list-style-type: none"> AM Resource Pool Push.xml
AM Resource Pool Relation Push 2.0	Computer_ ResourcePool_ Relations	<ul style="list-style-type: none"> Resource Pool Node 	<p>Pushes the relations between virtual machines and IBM processor pools.</p> <p>Depends on:</p> <ul style="list-style-type: none"> AM Node Push 2.0 AM Resource Pool Push 2.0 <p>Mapping XML:</p> <ul style="list-style-type: none"> AM Resource Pool Relations Push.xml

Asset Manager Federation Configuration

By default, the following supported and selected CI types can be federated from Asset Manager:

- Node
- CPU
- DiskDevice
- FileSystem
- InstalledSoftware
- LogicalVolume
- Printer



The following table lists all TQLs for the data federation that are contained in the out-of-box AM Generic Adapter. For each TQL query, it describes the source AM Entity type and the target CI type converted by the out-of-box mapping XMLs.

TQL Query	AM Entity	UCMDB CI Type	Mapping XML
AM Federation Cpu 2.0	ITEquipment	CPU	AM Cpu Federation.xml
AM Federation Cpu_Node 2.0	ITEquipment	CPU <ul style="list-style-type: none">• Node	AM Cpu Node Relations Federation.xml

TQL Query	AM Entity	UCMDB CI Type	Mapping XML
AM Federation DiskDevice 2.0	PhysicalDrive	DiskDevice	AM DiskDevice Federtion.xml
AM Federation DiskDevice_Node 2.0	PhysicalDrive	DiskDevice <ul style="list-style-type: none"> Node 	AM DiskDevice Node Relations Federtion.xml
AM Federation FileSystem 2.0	LogicalDrive	FileSystem	AM FileSystem Federtion.xml
AM Federation FileSystem_Node 2.0	LogicalDrive	FileSystem <ul style="list-style-type: none"> Node 	AM FileSystem Node Relations Federtion.xml
AM Federation InstalledSoftware 2.0	SoftwareInstallation	InstalledSoftware	AM InstalledSoftware Federtion.xml
AM Federation InstalledSoftware_Node 2.0	SoftwareInstallation	InstalledSoftware <ul style="list-style-type: none"> Node 	AM InstalledSoftware Node Relations Federtion.xml
AM Federation Printer 2.0	LogicalDrive	LogicalVolume	AM LogicalVolume Federtion.xml
AM Federation Printer_Node 2.0	LogicalDrive	LogicalVolume <ul style="list-style-type: none"> Node 	AM LogicalVolume Node Relations Federtion.xml
AM Federation LogicalVolume 2.0	Portfolio_Printer	Printer	AM Printer Federtion.xml
AM Federation LogicalVolume_Node 2.0	Portfolio_Printer	Printer <ul style="list-style-type: none"> Node 	AM Printer Node Relations Federtion.xml

Verify Out-of-Box Population and Push Jobs



This section describes how to verify out-of-box population and push jobs.




Synchronize Data between UCMDB and Asset Manager

The AM Generic Adapter provides two types of jobs:

- Data push jobs copy CI or CI relationship records from your UCMDB system to your Asset Manager system.
- Data population jobs copy Asset or Asset relationship records from your Asset Manager system to your UCMDB system.

To run a data push/population job, follow these steps:

1. Log on to UCMDB as an administrator.
2. Go to **Data Flow Management > Integration Studio**. UCMDB displays a list of existing integration points.
3. Select the integration point you created for Asset Manager.
4. Add a new data push/population job as follows:
 - a. Click the  button on the right panel.
 - b. In the Name field, type a unique name for the job.
 - c. Click the  button to add existing TQL queries to the job.
 - d. For push jobs, select or unselect the **Allow Deletion** option for each query. (The setting determines if this TQL query is allowed to delete data from Asset Manager, though the actual action on delete is defined by CI type in the mapping xml.)

For population jobs, select the **Allow Integration Job to Delete Removed Data** according to the requirement. (The setting determines if the population job is allowed to delete CIs from UCMDB.)
 - e. Click **OK**.
 - f. Save the integration point.
5. Manually run the job to see if the integration job works properly:
 - a. To push/populate all the relevant data for the job, click the  button.
 - b. To push/populate only the changes in the data since the job last executed, click the  button.
6. Wait for the job to complete, click the  button multiple times as needed until the job is completed.
7. When the job is completed, the job status becomes one of the following depending on the results:

- Succeeded
 - Completed
 - Failed
8. Click the **Statistics** tab to view the results. If any errors occur, click the **Query Status** tab and **Job Errors** tab for more information. For more information about errors, see ["Troubleshooting and Limitations" on page 121](#) .

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

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

How to View UCMDB Data in Asset Manager

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

Nodes

This includes computers, network devices, and so on.

To view the nodes:

1. Log on to Asset Manager as an administrator.
2. Go to **Portfolio Management > Asset Configurations > IT equipment > Computers and virtual machines**.
3. In the opened dialog box, for **IP name**, type the name of the computer you are searching for.
4. You may use '<name prefix>%' for easier searching. For example, searching IP name for **mycomp%** returns computer mycomp1, mycomp2, and so on.
5. Browse the computer for the different data.

Business Elements

This includes Business Applications, Business Services and Business Infrastructures.

1. Log on to Asset Manager as an administrator.
2. Go to **Asset Lifecycle > IT Services and virtualization > Business services > Business services**.
3. Browse the different Services.

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

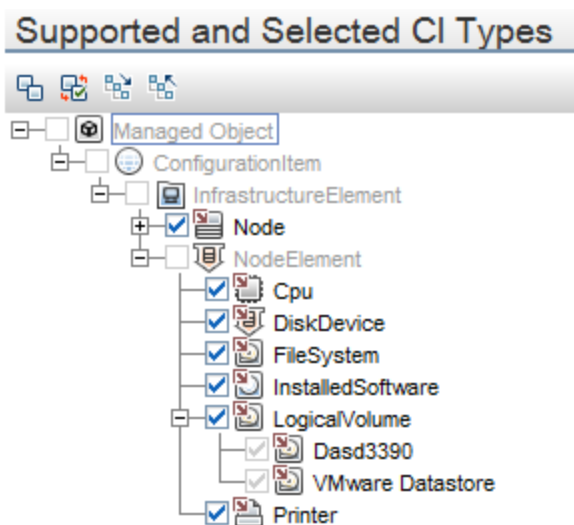
How to View Asset Manager Data in UCMDB

After a population job is successfully completed, you can search for and verify that the populated Asset data is in UCMDB. To do this, follow these steps.

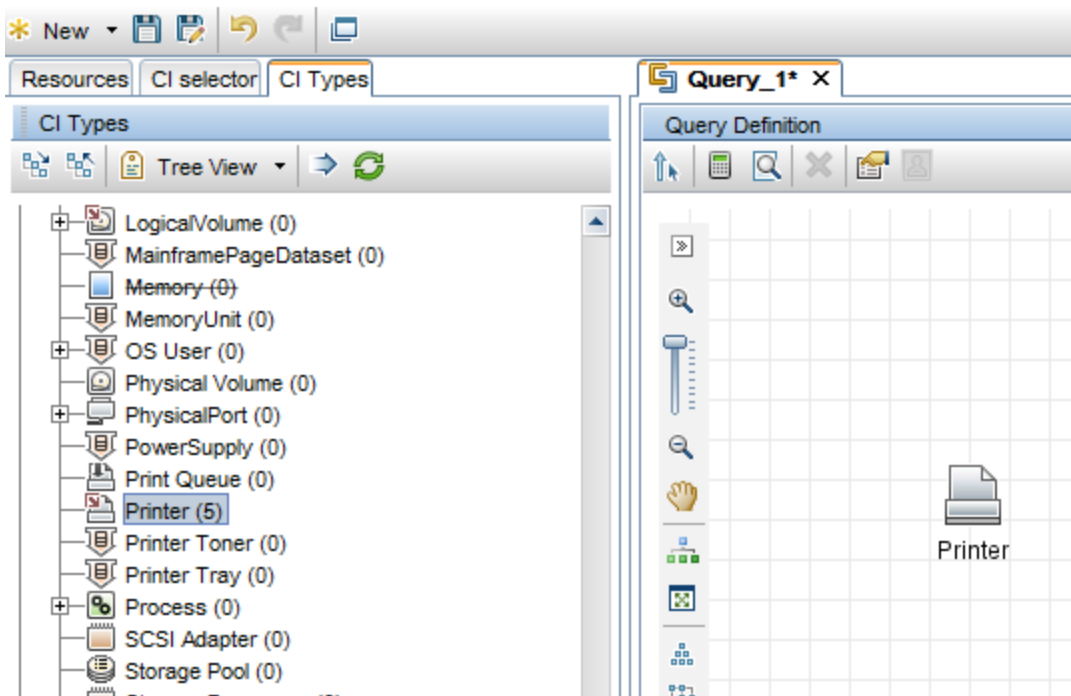
1. Log on to UCMDB.
2. Go to **Modeling > IT Universe Manager**.
3. Click the **Search CIs** tab and search for the name of the Asset populated from Asset Manager.

How to Federate Asset Manager Data in UCMDB

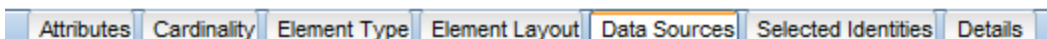
After Integration point of AM Generic Adapter is created and activated, you will find the supported CI types in the Federation page.



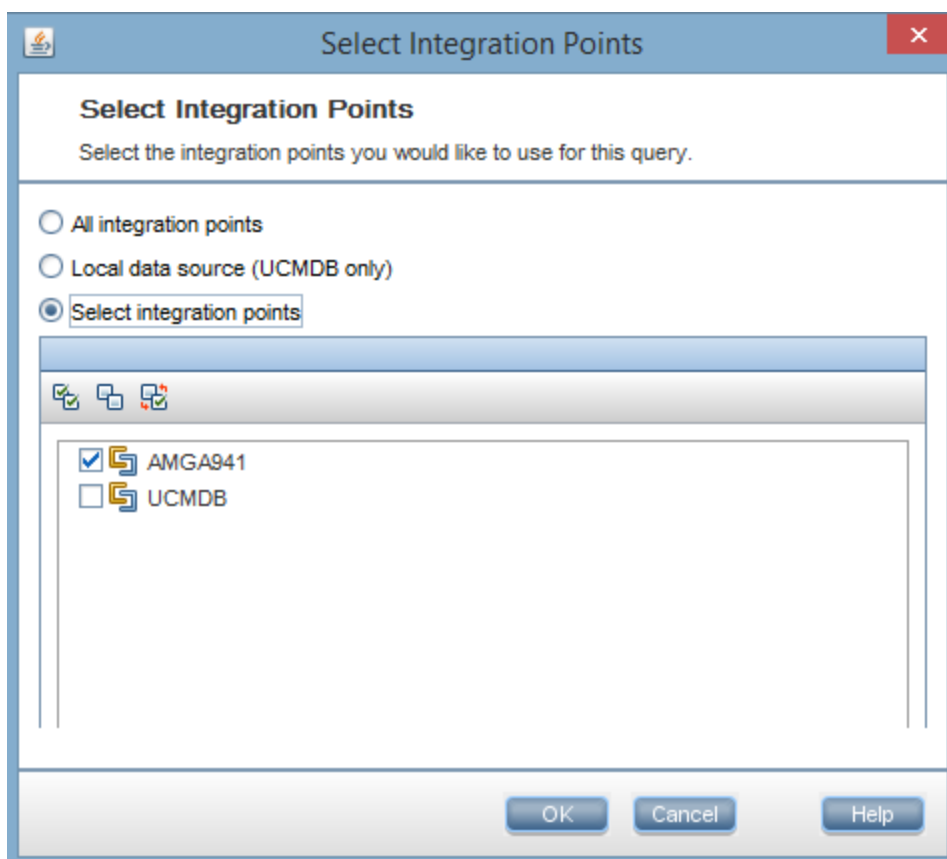
To federate these supported CI in UCMDB, you can create a query in **UCMDB client > Modeling > Modeling Studio > Resources**. Then, add a supported CI to federate data from AM. For example: Printer.



Click the CI icon, and then find Data Sources tab from the following area.



Edit Data Sources, select integration points from the following options, and then select integration points, for example, AMGA941.



Verify Calculate Query Result Count or Preview, you can federate those printers from AM server.




Integration Jobs Configuration

This section describes the configurations to be made to the integration jobs.

How to Schedule Data Integration Jobs

UCMDB allows you to schedule job executions directly from integration jobs. To do this, follow these 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 the UCMDB - AM integration.
4. Select the push job.
5. Click the  button.

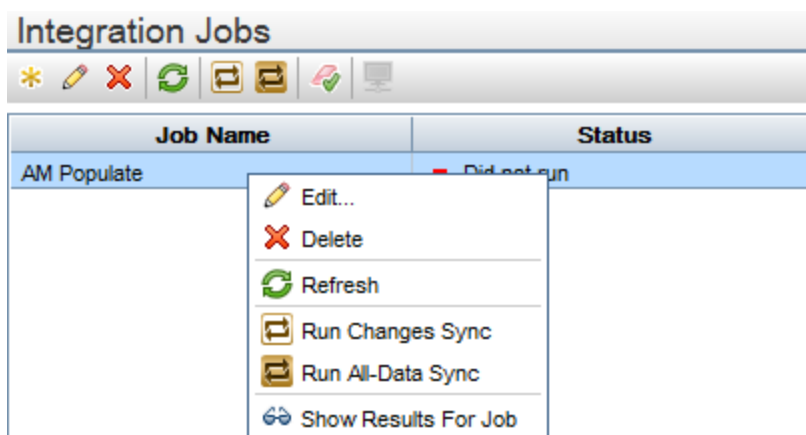
Note: UCMDB allows you to define two different schedules for two types of data push: **Changes Synchronization** and **All Data Synchronization**. We recommend that you use the Changes Sync schedule to only synchronize changes and avoid synchronizing the entire set of data each time.

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

Edit Data Integration Jobs

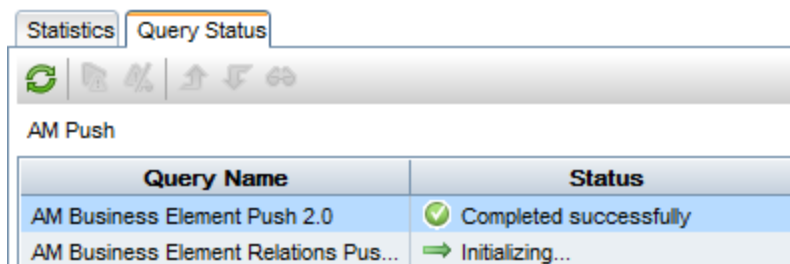
When you create an AM Generic Adapter integration point, a default job is created for data population and data push respectively. You can perform the following actions in the jobs.

- Add or remove TQLs.
- Clear cache  (available only for population jobs)
- Show Results For Job  (available only for population jobs)



Query Status

- Push selected failed data
- Suppress selected failures/warnings
- Up One Level/Down One Level
- View details



Job Definition

- Allow Integration Job to delete removed data
- Add/Delete
- Move Query Up/Move Query Down
- Edit Query Resources

Job Definition

☐ Allow Integration Job to delete removed data



AM Location Population 2.0

AM Node Population 2.0

AM Interface Population 2.0

AM BusinessElement Population 2.0

AM BusinessElement Relations Population 2.0

AM Printer Population 2.0

Standards and Concepts

This section describes standards and concepts.

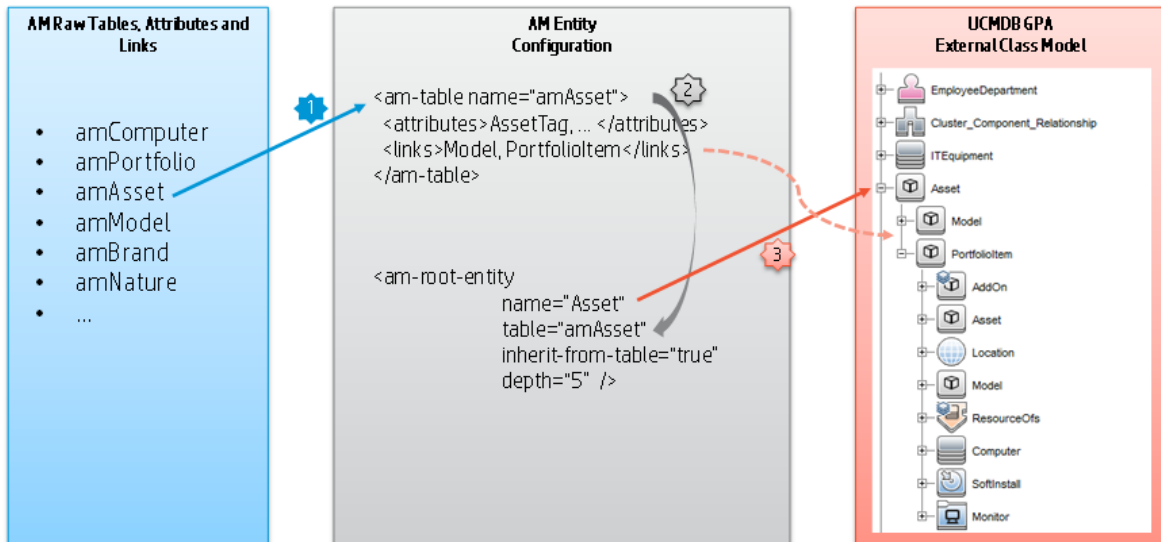
Asset Manager Entity

The AM Generic Adapter introduces a new concept named AM Entity. The AM Entity is a type of object that represents a collection of attributes abstracted from an AM table. All entities defined in the AM Generic Adapter make up the AM Class Model which represents the subset of AM objects that can be used in the population, push, as well as the graphic mapping UI. With the introduction of the AM Class Model (i.e. Entity) as an additional layer on top of AM database API, population and push mappings do not need to directly specify exact AM database tables or views. Extending AM Class Model is as simple as defining new entities by editing the `am-entity-config.xml` file. The entity name can be customized to a value that best fits your preference.

The entity is used mainly in the following places.

- It is displayed in the external class model pane on the mapping UI.
- It is referenced in mapping scripts.
- It is used in `am-populate-config.xml` for configuring initial AQL (AM Advanced Query Language) conditions to produce data from AM (used by population and federation).
- It is used in `am-push-config.xml` for reconciliation rule definition (used by push).

Asset Manager Entity Definition Steps



The AM entity definition steps are as follows.

1. Import AM tables.
2. Create a new AM entity based on an imported AM table.
3. Define the entity's attributes and links to be used in the AM Generic Adapter.

The definition of the entities is saved in the am-entity-config.xml file. To access the file, go to **Data Flow Management > Adapter Management > Packages > AMGenericAdapter > Configuration Files**.

The am-entity-config.xml file consists of two sections.

- The **Import Tables** section contains the declaration of AM tables imported to the AM Generic Adapter, as well as what attributes and links in a table can be used in the adapter.
- The **Entity Definition** section contains the definition of AM entities which are used as the root level entity for data mapping. An entity is based on a table imported in the **Import Table** section. Additionally, an entity can specify a subset of attributes and links in the imported table to limit its own attributes and links.

Import Tables and Entity Definition

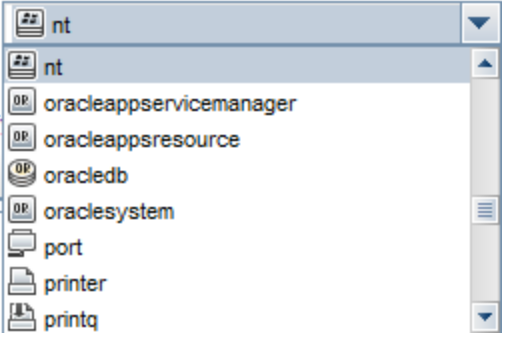
The following tables describe the schema of the am-entity-config.xml file.

Import Tables

Name	Type	Description
am-tables	Tag	
name	Attribute	The SQL name of the AM database table.
attributes	Tag	Specifies attributes (SQL name) in the table that can be used in the adapter. Multiple attributes are separated by comma. Use * to import all attributes.
links	Tag	Specifies links (SQL name) in the table that can be used in the adapter. Multiple links are separated by comma. Use * to import all links. Note: The table to which the link references must also be imported.

Entity Definition

Name	Type	Description
am-entities	Tag	
am-root-entity	Tag	
name	Attribute	AM entity name. The entity name is used to reference an entity in the data mapping, AM Class Model, configuration for push, and population. We strongly recommend that you give it a meaningful name.
table	Attribute	The SQL name of the AM database table where the entity's attributes and links come from. The table must be imported in the Import Tables section.
inherit-from-table	Attribute	<ul style="list-style-type: none">• True: merge attributes and links defined in the imported table to the entity.• False: only attributes and links defined in the entity can be used. The default value is False.
depth	Attribute	Defines the depth of level links in the entity can be extended. The default depth is 4, which means the deepest remote link you can use is entity.linkA.linkB.linkC.linkD.
icon-type	Attribute	The icon to be displayed in the graphic mapping UI to represent the entity. It

Name	Type	Description
		<p>refers to the values from UCMDB > Modeling > CI Type Manager > Icon.</p> <p>Configuration Item Type Main Icon: </p> <p><input type="checkbox"/> Change the Configuration Item Type</p> <p>Attributes</p> <p>ack_cleared_time ack_id Actual Delete Time</p>
attributes	Tag	Specifies the attributes (SQL name) in the table that can be used in the adapter. Multiple attributes are separated by comma. Use * to import all attributes.
links	Tag	Specifies the links (SQL name) in the table that can be used in the adapter. Multiple links are separated by comma. Use * to import all links. Note: The table to which the link references must be imported.

Out-of-Box Entity Definition

The following table covers all of the out-of-box entities defined in the AM Generic Adapter. For the list of tables imported to the adapter, see the am-entity-config.xml file.

AM Entity Name	AM Table	Comment
ITEquipment	amComputer	<p>You can customize entity names by suffixing a sub CI type to define a more exact AM Entity. For example,</p> <ul style="list-style-type: none"> ITEquipment_Windows ITEquipment_Unix <p>If you want to query an AM table with different AQL conditions, you need to separate entities.</p>
Location	amLocation	
NetworkCard	amNetworkCard	
PhysicalDrive	amPhysicalDrive	
SoftwareInstallation	amSoftInstall	

AM Entity Name	AM Table	Comment
Asset	amAsset	
LogicalDrive	amLogicalDrive	
Cluster	amCluster	
ClusterComponent	amClusterComponent	
ExtensionCard	amExtensionCard	
Monitor	amMonitor	
MonitoredApplication	amMonitoredApp	
EmployeeDepartment	amEmplDept	
Portfolio	amPortfolio	<p>In practice, you can customize the entity name by AM natures or models to define more exact AM entities. For example,</p> <ul style="list-style-type: none"> • Portfolio_NetworkHardware • Portfolio_StorageConsumer • Portfolio_StorageAggregation • Portfolio_StorageProvider • Portfolio_VirtualMachine
Cluster_Component_Relationship	amClusterComponent	
Client_Resource_Relationship	amClientResource	
Portfolio_Printer	amPortfolio	<p>This entity is for local printers that only saved in amPortfolio table but not in the amComputer table. It is used in the data population. Its query condition is:</p> <pre>Model.Nature.Code = 'PRN' AND lParentId <> 0</pre>

UCMDB TQL

AM Generic Adapter communicates with UCMDB through TQL queries.

- For push operations, a TQL query produces CIs which are the source to be translated to Asset Manager.
- For population, a TQL query defines CI types to be pulled from Asset Manager.
- For federation, a TQL query is used to configure the Data Sources of a CI node. For example, you can specify the Integration Points of a CPU node to an AM Generic Adapter integration point.
- For graphic mapping UI, a TQL query displays the CI relation tree.

A TQL query used for the data push job must contain a root query node.

In the OOTB TQLs for data push, some query nodes are set with criteria to query valid CIs to be translated to Asset Manager. For example, "Not NodeRole is null" is one of the conditions specified in the Node CI of the AM Node Push TQL.

Any attribute used in the mapping flow must be marked in the selected layout of the query node. Each TQL query may only have one mapping. We recommend that you only add attributes which are used in the data mapping to the selected layout. Add unused attributes to the selected layout may have impact on the performance.

You can customize TQLs, for example, add a condition to a query node or add a new query node .

TQL Naming Convention

All OOTB TQLs follow the naming convention below. We recommend that you use the same naming convention when creating new TQLs for the AM Generic Adapter.

AM <CI Type> <Push | Population> <version #>

For example:

- AM Node Population 2.0
- AM Business Element Push 2.0

Groovy Functions

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

Basic Functions

There are two basic function groovy files:

- AMPopulate.groovy
- AMPush.groovy

Basic functions in these two files are for implementing data conversion and transformation in mapping scripts, and they will simplify mapping scripts, although mapping scripts also support basic groovy syntax.

AM Population Groovy

There are some functions in population groovy implement translation and discrimination logic. The following list shows all the functions in AMGenericAdapter/mappings/scripts/AMPopulate.groovy.

- convertIpAddressProperty
- getAssignment
- getCRState
- getHostDataCI
- getKpiComparisonOperator
- getLocationType
- getIpAddressProperty
- getIpAddressValue
- getNodeRole
- getNodeType
- getServiceClass
- isAcceptableNode
- isDebugEnabled
- isDeletedNode
- isDesktop

- isValidLocationParent
- isValidIp
- isValidNode
- isVirtual
- setLog

AM Push Groovy

There are some functions in push groovy implement business logic conversion, especially for Asset Manager SAM calculation. The following list shows all the functions in AMGenericAdapter/mappings/scripts/AMPush.groovy.

- calculateComputerType
- convertCsvBytesToString
- flsSAIValid
- fEDDIGetMACAddressEx
- fEDDIGetACComputerBrand
- fEDDIGetACWorkGroup
- fEDDIGetComputerManufacturer
- fEDDIGetDomainNameEx
- fEDDIGetSCLogicalName
- fEDDITruncate
- getAMPPrimaryID
- getAMPPrimaryID
- getAssetTagCRSystem
- getBrandName
- getCardID
- getCardName
- getCardVendorName
- getClusterModelName

- getClusterTechnology
- getCpuInternal
- getCpuType
- getCsvFile
- getDdEdition
- getDNSSuffix
- getDomainName
- getEnd1ExternalId
- getEnd2ExternalId
- getEnumValueByKey
- getExternalId
- getFinalModelId
- getFirstIpV4Address
- getFirstIpV6Address
- getFolder
- getLMSOptionStatus
- getLMSPacksStatus
- getInterfaceDescription
- getLogicalCpuCount
- getIpAddress
- getIpAddress
- getIpV4SubnetMask
- getIpV6SubnetMask
- getIsInventModelExist
- getIsInventModelResolved
- getModelIdByBarCode
- getModelName
- getModelParentCode
- getMonitorCode

- getMonitorModelName
- getNatureCode
- getNatureCodeFromType
- getNatureNameFromType
- getNormalizedCPU
- getNormalizedModelName
- getOracleComponentList
- getOracleEditionShortName
- getOracleFolder
- getParentBarCodeFromType
- getParentName
- getPhysicalAddress
- getPrimaryIpAddress
- getSafeAMPPrimaryID
- getSoftwareBarCode
- getUsername
- getValueFromCsv
- getValueFromStringCsv
- getVMType
- hasFinalModelId
- isDHCPEnabled
- isExtensionCardValid
- isLparVpar
- isSolarisZone
- nicelyPrintExternalId
- setLog
- shouldMapMonitor
- sortIpList
- validateAndRetrieveSingleUCCaseAsset

Utility Functions

Some general logic and functions are the same in the utility groovy file. `AMPopulate.groovy`, `AMPush.groovy`, and `AMReconcil.groovy` are inherited from this groovy. So in push or population mapping, even the utility groovy is not imported, all of the following functions in `AMGenericAdapter/mappings/scripts/AMUtils.groovy` can be used normally.

- `boolToInt`
- `compareDate`
- `containsIgnoreCase`
- `Extractvalue`
- `fCleanValueWithDefault`
- `fisContainOne`
- `fIsContainList`
- `fIsEmpty`
- `fIsEmptyOrZero`
- `isMatchVersion`
- `isNull`
- `leftPart`
- `rightPart`
- `toBoolean`
- `toSmart`
- `toStringList`
- `trimRight`
- `uCase`

Reconciliation Functions

Reconciliation groovy file provides advanced features about AM reconciliation proposal, insert or update script, and so on. The following list shows all the functions in

AMGenericAdapter/mappings/scripts/AMReconcil.groovy.

- createReconcProposalAdvance
- getCurrDate
- getCurrDateLong
- getIndexByName
- getModelFromInstalledSoftwareCI
- getShortHostName
- getSqlTextConst
- getValueByIndex
- getValueByName
- isDateAfter
- isResultEmpty
- setReconcProposalInvalidate
- setReconcProposalObsolete
- stringToDom
- updateMemorySize

Data Mapping Schema

AM Generic adapter is built on top of the Generic Adapter framework. Its data mapping schema follows the standard schema in the framework. For more information, see *Universal CMDB Developer Reference Guide*.

The AM Generic Adapter uses some of the elements specifically to fit the AM-UCMDB integration requirement.

Automatic Creation of Relation CI in Data Population

When populating two CIs that are defined in TQL, their relation CIs are created in UCMDB automatically according to the TQL query definition. The explicit mapping for the relation between the two CIs is not required. For example, in the AM Interface Population XML, it does not define the data mapping for the relation between Interface and Node. However, the relation will be created automatically when you run the AM Interface Population job.

If you want to create the relation CI with non-default values, you need to define a relation CI mapping in the mapping XML file.

Define Reconciliation Rule for Target AM Entity in Push

The name attribute of the target_entity tag represents:

- The entity name defined in the am-entity-config.xml file when defining the target_entity for the root CI.
- The link's SQL name in the AM Class Model when defining the target_entity for non-root CIs.

For example, in the AM Node Push.xml, the name of the target_entity for the root CI is set to ITEquipment, which is an entity type based on the amComputer table. The name of the target_entity for the Root.File_System CI is set to LogicalDrives, which is the SQL name of the link to the amLogicalDrive table in the ITEquipment entity.

The type attribute of the target_entity tag represents an alias to be used in the am-push-config.xml file to define the reconciliation rules for the data mapping. For more information about how to define reconciliation rules, see ["Reconciliation" on page 66](#).

It is an optional attribute. When it is not given a value, it takes the value of the name attribute as its value.

For example, in the AM Node Push.xml, the type of the target_entity for the Root.Display_Monitor CI is set to Monitor-amPortfolio. In the am-push-config.xml file, it uses the type value Monitor-amPortfolio to define the reconciliation rule for the mapping.

```
<am-mapping ci-type="Monitor-amPortfolio" name="AddOn" primary-
key="lPortfolioItemId" operation-type="update_else_insert" parallel-push-
allowed="true" merge-allowed="true" to-version="9.4*">
  <reconciliation>
    <reconciliation-keys>
      <reconciliation-key>Folder</reconciliation-key>
      <reconciliation-key>lParentId</reconciliation-key>
    </reconciliation-keys>
  </reconciliation>
...
</am-mapping>
```

Specify Sub CI Type in Population Data Mapping

In the population data mapping, the type attribute of the target_entity is used to specify the sub type of the target CI.

For example, in the AM Node Population.xml, the AM ITEquipment entity is mapped to the Node CI (the Node CI is set to Root in the TQL query). When the population job is running, the exact CI type the

ITEquipment entity is converted and is determined by the value of the type attribute. In the out-of-box mapping, it calls a groovy function to set the value of the type attribute.

Determine the Deletion of AM Entities in Population

In the population data mapping, the is-deleted attribute of the target_entity tag is used to determine if the target UCMDB CI corresponding to the AM entity needs to be deleted.

Produce Log

You can use the Logger object to produce log in the before-mapping and after-mapping tag.

For example,

```
<before-mapping>Logger.debug('before')</before-mapping>  
<after-mapping>Logger.debug('after')</after-mapping>
```

Population and Federation

AM Generic Adapter is able to retrieve data from AM database via AM DLL API. This section describes the detailed criteria before query AM Entity.

Criteria for Asset Manager Records to be Populated

AM Entity	AQL
ITEquipment	TcplpHostName <> " AND (TcplpHostName IS NOT NULL)
NetworkCard	Computer.TcplpHostName <> " AND (Computer.TcplpHostName IS NOT NULL) AND (TcplpAddress IS NOT NULL) AND (PhysAddress IS NOT NULL) AND (TcplpAddress <> ") AND (PhysAddress <> ")
SoftwareInstallation	IParentPortfolioId <> 0 AND ParentPortfolio.CMDBId IS NOT NULL AND ParentPortfolio.CMDBId <> " AND ParentPortfolio.Computer.TcplpHostName <> " AND (ParentPortfolio.Computer.TcplpHostName IS NOT NULL)
Portfolio_Printer	Model.Nature.Code = 'PRN' AND IParentId <> 0
LogicalDrive	Computer.TcplpHostName <> " AND (Computer.TcplpHostName IS NOT NULL)
PhysicalDrive	Computer.TcplpHostName <> " AND (Computer.TcplpHostName IS NOT NULL)

AM Entity	AQL
Asset	IAstId <> 0 AND PortfolioItem.Model.Nature.bSystem = 1
Client_Resource_Relationship	CRSystem.IAstId <> 0 AND CRSystem.Model.Nature.bSystem = 1

Note: If you want to populate the same AM entity with different query condition, you should duplicate AM entity, rename it to a new entity, and then add an AQL for this new entity.

Transformation for Asset Manager Records to be Populated

IT equipment

The default population mapping will populate an IT equipment when its status is In use or In stock.

According to the IT equipment type, data will be populated to UCMDB as different CI types:

CI types in UCMDB	IT Equipment Type in AM
host_node	<ul style="list-style-type: none">• Computer• Desktop computers• Computer servers• Laptop• Virtual Machine• Smart phone
nt	<ul style="list-style-type: none">• Windows computer• Windows desktop computer• VMware VirtualCenter
unix	<ul style="list-style-type: none">• Unix server computer• Unix desktop computer• Solaris Zone server
vmware_esx_server	VMware ESX Server
mainframe	<ul style="list-style-type: none">• Mainframe• Mainframe CPC
lpar	Mainframe Logical Partition

When a CI is populated to UCMDB, the Host is Virtual attribute of the CI is defined according to the value of IT Equipment type for the CI in AM.

Host is Virtual	IT Equipment Type in AM
Yes	<ul style="list-style-type: none"> • Virtual Machine • Mainframe Logical Partition
No	<ul style="list-style-type: none"> • Windows computer • Windows desktop computer • VMware VirtuaCenter • VMware ESX server • Unix server computer • Unix desktop computer • Solaris Zone Server • Desktop computers • Computer servers • Laptop • Mainframe • ATM switch • Firewall • Router • Switch • Network printer • Smart phone • Mainframe CPC

When a CI is populated to UCMDB, the Host is Desktop attribute of the CI is defined according to the value of IT Equipment for the CI in AM.

Host is Desktop	IT Equipment Type in AM
Yes	<ul style="list-style-type: none"> • Windows desktop computer • Unix desktop computer • Desktop computers
No	<ul style="list-style-type: none"> • Windows computer • VMware VirtuaCenter • VMware ESX server

Host is Desktop	IT Equipment Type in AM
	<ul style="list-style-type: none">• Unix server computer• Solaris Zone Server• Computer servers• Laptop• Mainframe• ATM switch• Firewall• Router• Switch• Network printer• Smart phone• Mainframe CPC

Business Element

In AM, only those business service assets with status (amAsset.Status) Built, Catalogued, Chartered, Designed , and Requested will be populated to UCMDB. If the status is retired, it will be removed from UCMDB.

Relations between AM business service asset and CI types are as follows.

Nature code	CI Type
BIZSVC	business_service
BIZAPP	business_application
INFRASVC	infrastructure_service

In AM Generic Adapter, all the transformers and discriminators are implemented in groovy functions. For more information, see ["Groovy Functions" on page 48](#).

Reconciliation

For each CI Type, the data reconciliation is governed by the reconciliation rule set in UCMDB.

You can check the reconciliation rule for each CI Type on the Details tab of the CI Type. The field name is Identification.

Population Condition and Push Back Definition

Name	Type	Description
am-populations	Tag	
am-population	Tag	
entity-name	Attribute	AM Entity name. Its value is from <code>am-entity-config.xml</code> , it is defined by <code>am-root-entity</code> . An entity is an alias of an AM table, which helps customers understand its usage. An entity can only be used at the first level.
push-back-field	Tag	<p>It defines a field of an AM table, which is used to store the UCMDB global ID in the current entity population. The following list describes how to use the push-back-field tag.</p> <ul style="list-style-type: none"> If you configure a field by <code>push-back-field</code>, the adapter will validate it before starting the population. <ul style="list-style-type: none"> If the field is invalid, an exception will be thrown and the job is not run. If the field is valid, the adapter will continue to run the job. You can configure <code>push-back-field</code> like: <pre><push-back-field>Portfolio.CMDBId</push-back-field> or <push-back-field>CMDBId</push-back-field></pre> If you do not use <code>push-back-field</code> in <code>am-population</code> or it is empty, the adapter will try to find the <code>GlobalId</code> field in the current population AM entity. <ul style="list-style-type: none"> If <code>GlobalId</code> is found, it will be used to save UCMDB global ID value. If <code>GlobalId</code> is not found, the adapter does nothing.
delta-sync-date-field	Tag	<p>It is used to define a time condition field for AQL in changes population. By default, it is the field 'dtLastModif' that is from the AM entity which is used as the 'root-element' in the population mapping. It is used together with <code>query-condition</code> to define the condition of the query.</p> <p>Example:</p> <pre><am-population entity-name="xxxx"> ... </pre>

Name	Type	Description
		<pre><delta-sync-date-field> DELTA TIME FIELD HERE! </delta-sync-date-field> ... </am-population></pre>
query-condition	Tag	<p>The condition of the query to retrieve the entity data from AM. It supports AM AQL. If the condition includes special characters, for example: >, <, make them contained in <![CDATA[]]>. <![CDATA[]]> is an XML tag that enables the XML parser to handle the content in the tag as plain text.</p> <pre><query-condition><![CDATA[lAstId <> 0 AND PortfolioItem.Model.Nature.bSystem = 1]]></query-condition></pre>

Built-in attributes from AM

- AM_PUSHBACK_ID

This attribute is created by population adapter automatically from AM. It is used to identify which AM record is written with push back global id.

- LAST_SYNC_TIME

This attribute is created by population adapter automatically. It keeps the last job changes sync time in order to be used in population mapping.

Federation Tags

The configuration must be provided if you want to use federation.

It defines all valid federation TQL names.

Name	Type	Description
supported-federation-queries	Tag	
tql-query	Tag	
name	Attribute	TQL name

Population Tags

The configuration is used to define all valid population TQL names.

Name	Type	Description
supported-population-queries	Tag	
enable	Tag	<ul style="list-style-type: none">• True: only displays valid population TQL names for you, to select required population TQLs for a population job.• False: displays all TQL names from UCMDB for you, to select required population TQLs for a population job.
tql-query	Tag	
name	Attribute	TQL name

Push and Reconciliation

This section describes push and reconciliation operations.

Data Flow Architecture

The data flow architecture is as follows.

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 Generic Adapter.
5. The Generic Adapter loads the correct mapping for the specific TQL query.
6. All **dynamic_mappings** are executed and saved to maps, to allow usage in the next mapping

stage.

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

7. Data is mapped from the UCMDB data Model into the AM Data model according to the mapping XML.
8. Data is sent to the AM Connector.
9. AM Connector orders all the data in a set of dependency trees, starting with the records that do not depend on any other record.
10. AM Connector attempts to merge any duplicate records
11. AM Connector starts reconciling and pushing any record without any dependencies, or a record whose dependencies have already been reconciled/pushed to Asset Manager.
 - a. AM Connector first tries to reconcile with existing records.
 - b. If it finds a match, it attempts to update that record.
 - c. If it does not find a match, it attempts to create a new record.
12. AM Connector deletes any records that are required to be deleted in AM, as permitted by action-on-delete.

Integration TQL Queries

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

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

Each TQL query for data push job may only have one mapping.

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

Reconciliation Proposals

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

Operator validation or action to support AM business processes.

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

How to use Reconciliation Proposals

In the OOTB configuration **IMemorySizeMb** is marked for attribute-reconciliation. The update script calls the **updateMemorySize** function. This function verifies if the memory size of the computer was decreased. It initializes all the parameters that are passed to the function

validateReconcUpdateAdvance. Calls **validateReconcUpdateAdvance** and return its returned value.

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

Parameter	Description
AMApiWrapper	The wrapper that is used to communicate with the AM.
newVal	The value of the attribute in the pushed data.
oldVal	The value of the attribute that is retrieved from AM.
recordId	The primary key of the table that the attribute belongs to.
strCode	The prefix of the code field in the reconciliation proposal.
strName	The name of the reconciliation proposal.
path	The name of the attribute.
recordTable	The table that the attribute belongs to.

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

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

1. Add the **<attribute-reconciliation>** tag for this attribute.
2. The update-script should call a new function that initializes the parameters passed to **validateReconcUpdateAdvance**, and returns the value returned from **validateReconcUpdateAdvance**.

Note: We recommend that you use the **updateMemorySize** function as a reference.

Asset Manager Rules and Flows

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

Mapping Attributes

Attributes in the <am-mapping-config> tag are as follows

Attribute	Description
ci-type	<p>It is used in am-push-config.xml to recognize the reconciliation rule. It must be the same as the 'type' attribute in AM mapping XML files.</p> <p>For example:</p> <ul style="list-style-type: none">• Mapping file AM Node Push.xml: <target_entity name="AddOn" type=""Printer-amPortfolio">• Reconciliation file am-push-config.xml: <am-mapping ci-type="Printer-amPortfolio" name="AddOn" ...>
name	AM entity name or sub link name.
primary-key	The primary key column in the AM database schema.
operation-type	<p>It defines what operations may be done with the record.</p> <p>insert - Only allows creation of new records; if it already exists, an exception is thrown.</p> <p>update - Only allows updates of an existing record; if it does not exist, an exception is thrown.</p> <p>update_else_insert - If the record exists, it is updated. Otherwise, the record is created.</p> <p>reference-only - The record is only used for being referenced by other records (and is not updated). An exception is thrown if the record does not exist.</p> <p>insert_else_reference - If the record does not exist, it is created. Otherwise it is only used as a reference and is not be updated; see reference-only.</p> <p>optional_reference - If reconciliation fails, it will not fail dependent CIs. Instead, 0</p>

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

Reconciliation

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

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

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

Example:

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

Target CI Validation

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

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

Name	Description
attribute-name	The attribute that you want to use for validation.
validation-script	<p>A Groovy based script that returns true if this record is to be pushed, and false if it is not to be pushed. The script may access any external Groovy code in the path in order to run the evaluation.</p> <ul style="list-style-type: none">• vNewVal - Attribute value of the record in memory.• vOldVal - Attribute value of the record in the AM database.
failed-validation-error-code	<p>This optional attribute holds the error code that appears if there is a validation failure.</p> <p>The arguments that can be referenced in the error message are:</p> <ul style="list-style-type: none">• {0} - The validated attribute name.• {1} - The property value in UCMDB.• {2} - The property value in Asset Manager.• {3} - The validation script.• {4} - The additional message from the additional-failure-message attribute, or 'null' if there is no additional message.
additional-failure-message	This optional attribute holds an additional error message that can be referenced by the error message in the properties.error file. See failed-validation-error-code , above.

Example:

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

Reference Attribute

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

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

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

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

Example:

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

Attribute Reconciliation

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

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

Name	Description
attribute-name	The attribute to be reconciled.
update-script	<p>The script to execute in case of an update operation on the record. The returned value by the groovy script will be push to AM as the value of this attribute.</p> <ul style="list-style-type: none">• vNewVal - Attribute value of the record in memory.• vOldVal - Attribute value of the record in the AM database.
Insert-script	<p>The script to execute in case of an insert operation on the record. The value returned by the Groovy script is pushed to AM as the value of this attribute.</p> <p>vNewVal - Attribute value of the record in memory.</p>

Example:

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

Action on Delete

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

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

The following actions are possible

- **<ignore>** - Do nothing.
- **<delete-ci>** - Delete this record from the AM database.
- **<set-attribute-value>** - Change the value of one or more attributes in the AM database.

Example:

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

Action on Update

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

The following actions are possible

- **<set-attribute-value>** - Change the value of one or more attributes in the AM database.
- **<target-ci-name>** - The table that contains the changed attribute.
- **<set-attribute-value>** - The condition attribute to identify the changed record.

Example:

```
<action-on-update>  
  <set-attribute-value name="bMissing" datatype="INTEGER" value="1" target-ci-  
name="amSoftInstall" target-ci-foreign-  
key="ParentPortfolio.Computer.lComputerId" />  
</action-on-update>
```

Enum Attribute

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

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

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

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

Ignored Attributes

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

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

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

Example:

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

Deletion

To allow the AM Generic Adapter to delete CIs in UCMDB on data population or delete assets in AM on data push, additional configurations are required. The following sections describes how to configure the AM Generic Adapter to allow data deletion in the AM-UCMDB integration.

Population Deletion Configuration

When a computer is retired in Asset Manager, the corresponding CI in UCMDB should be deleted as well. By default, it is not allowed to delete CIs from UCMDB through the AM population jobs. You need to complete the following three steps to allow population jobs to delete CIs from UCMDB.

1. In population job definition, enable Allow Integration Job to delete removed data to allow mapping script to send delete CI action to the UCMDB server.

Name

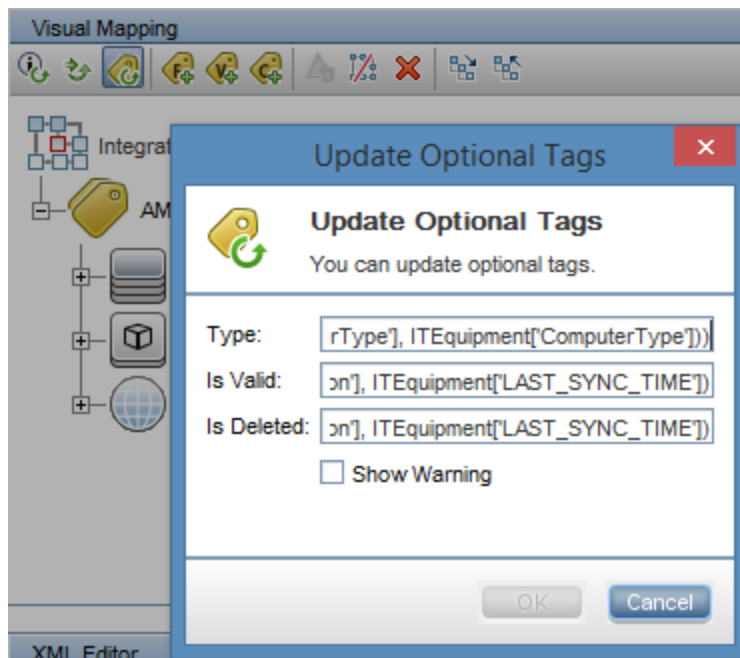
Job Definition

☐ Allow Integration Job to delete removed data

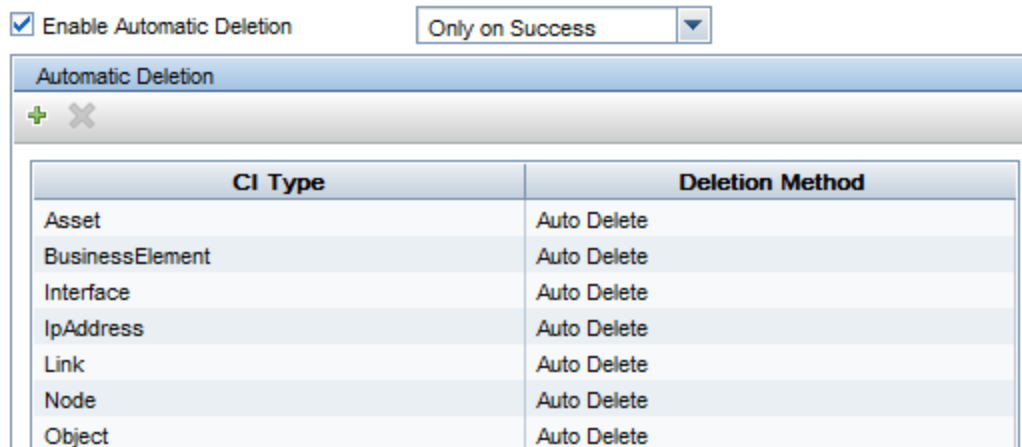
+ × ↑ ↓ | ✎

- AM Location Population 2.0
- AM Node Population 2.0
- AM Interface Population 2.0
- AM BusinessElement Population 2.0
- AM BusinessElement Relations Population 2.0
- AM Printer Population 2.0

2. In population mapping script, use the Is-deleted attribute on target-entity to determine what CIs should be deleted. You may also configure it from the graphic mapping UI.



3. Choose the Deletion Method in the Adapter Setting.



Push Deletion Configuration

When a CI is deleted from UCMDB, the corresponding asset record in AM should be deleted as well. By default, it is not allowed to delete assets from AM through the AM push jobs. You need to complete the following steps to allow push jobs to delete assets from AM.

1. In the push job definition, enable Allow Deletion for TQL queries which are allowed to delete asset records in AM.

Name

Job Definition

+ × ↑ ↓ ✎

Query Name	Allow Deletion
AM Node Push 2.0	<input checked="" type="checkbox"/>
AM Business Element Push 2.0	<input checked="" type="checkbox"/>
AM Business Element Relations Push 2.0	<input checked="" type="checkbox"/>
AM Host Server And Running VM Relations Push 2.0	<input checked="" type="checkbox"/>
AM Host Server And Running Solaris VM Relations Push 2.0	<input checked="" type="checkbox"/>
AM Host Server And Running LPAR VM Relations Push 2.0	<input checked="" type="checkbox"/>
AM Computer Relations Push 2.0	<input checked="" type="checkbox"/>
AM Net Device Relations Push 2.0	<input checked="" type="checkbox"/>
AM Installed Software Push 2.0	<input checked="" type="checkbox"/>

2. Define the Action on Delete for the target AM entity type in the am-push-config.xml file. For more information, see ["Action on Delete" on page 70](#).

Installed Software / Software Utilization

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

Note: The following sections describe three different flows for Installed Software. The same flows also apply to Software Utilization (User_Software_Utilization).

Note: The flows below show a simplified high-level flow of the different Installed Software or Software Utilization synchronization behavior. The actual behavior may be more complex in some cases, mainly for performance improvement. See also ["Switching between Installed Software and Software Utilization Flows" on page 76](#) and ["Installed Software / Software Utilization" above](#).

Normalized Installed Software

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

versions as needed.

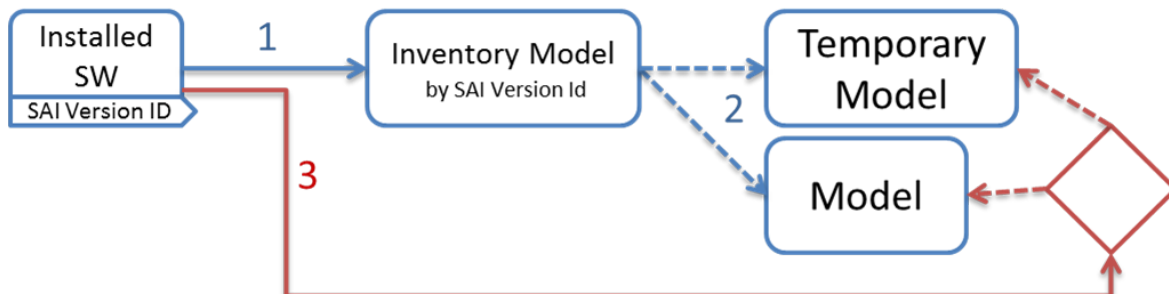


1. Each Installed Software is first mapped to an Inventory Model. If one does not exist, it creates one. The mapping is done according to the SAI Version ID which is an inventory ID from the Universal Discovery Scanner, or by using the Installed Software's name, version, and vendor.
2. It then sees if the InventoryModel has a final mapping to a Model. If it is a new InventoryModel, or the InventoryModel has no final mapping to a Model, it attempts to search for one with the same name and version. If one is found, it connects the InventoryModel to it; otherwise it creates a new Model.
3. It then connects the Installed Software to the Model as well.

Note: Normalized Installed Software is the default flow.

Normalized Installed Software – No Model Creation

This flow uses an InventoryModel to catalog each exact Software Version. Therefore, if the AM Operator decides to map certain Software version to a different model, he only has to do it once to the InventoryModel, and does not have to process all the Installed Software in AM. This flow does not automatically create a Model. Instead, the Model must be connected to the InventoryModel by a different flow, or manually by an Asset Manager Operator.



1. Each Installed Software is first mapped to an Inventory Model. If one does not exist, it creates one. The mapping is done according to the SAI Version ID, which is an inventory ID from the

Universal Discovery Scanner, or by using the Installed Software attributes: name, version, and vendor.

2. It then sees if the InventoryModel has a final mapping to a Model. If not, it chooses the temporary model (an Unknown Software Model).
3. It then connects the Installed Software to the Model found in the step 2.
4. Later, an Asset Manager Operator manually connects each Inventory Model to a final Model, as he wishes.

Non-Normalized Installed Software

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



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

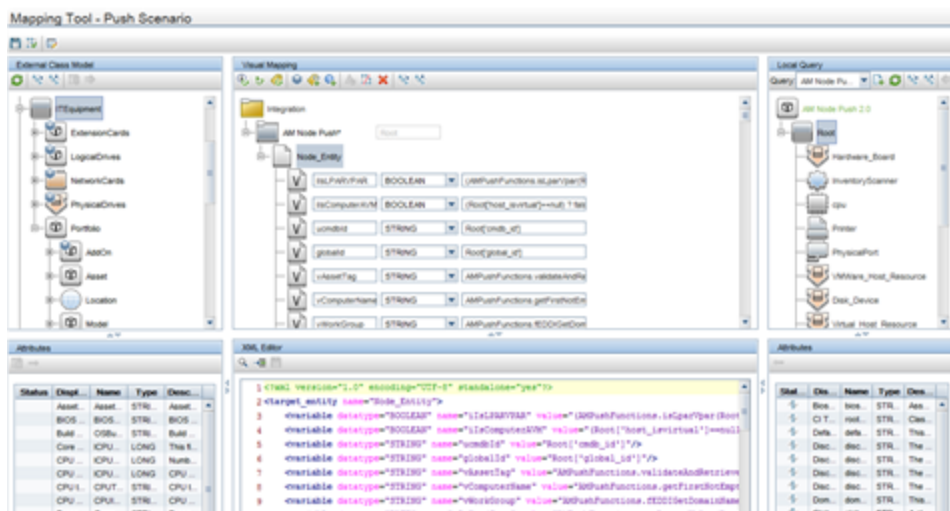
Switching between Installed Software and Software Utilization Flows

There are four OOTB TQLs for the Installed Software and Software Utilization push in the AM Generic Adapter.

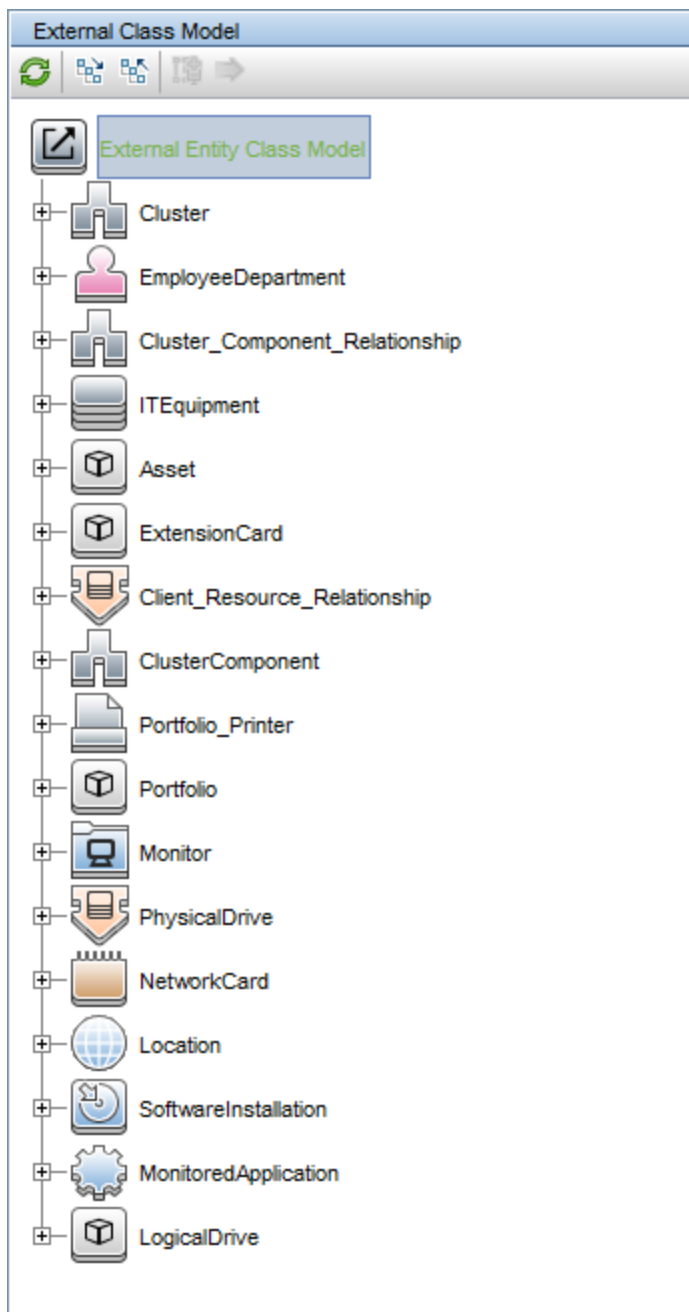
- **AM Installed Software Push 2.0** is the TQL that contains the Installed Software CIs which need to be normalized in Asset Manager. Running this TQL job triggers the installed software flow that is normalized.
- **AM Software Utilization Push 2.0** is the TQL that contains the Software Utilization CIs which need to be normalized in Asset Manager. Running this TQL job triggers the software utilization flow that is normalized.
- **AM Installed Software Normalized Push 2.0** is the TQL contains the Installed Software CIs which are already normalized in UCMDB and do not need to be normalized in Asset Manager. Running this TQL job triggers the installed software flow that is not normalized.
- **AM Software Utilization Normalized Push 2.0** is the TQL contains the Software Utilization CIs which are already normalized in UCMDB and do not need to be normalized in Asset Manager. Running this TQL job triggers the software utilization flow that is not normalized.

Mapping UI

You can use the graphic mapping UI to develop data mappings for the AM Generic Adapter. For more information about the mapping UI, see the Data Flow Management Guide.



AM Class Model displayed on Mapping UI



Prerequisites for using Mapping UI

- You have established an integration point with AM Generic Adapter.
- The integration point has been activated.

How to Tailor the Integration

This section describes how to tailor the integration.

How to Change Adapter Settings

To change adapter settings, follow these steps.

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

Setting	Default	Description
replication.chunk.size	4000	Defines the requested number of CIs and Relations sent in each chunk. It is possible to adjust this setting to try and improve the performance of the server and the probe.
replication.chunk.root.limit	850	Defines the maximum requested number of Roots sent in each chunk. This works with replication.chunk.size as a limiter on the amount of data sent in each chunk.
PushConnector.class.name	com.hp.amadapter.AMPushAdapter	It is AM Push Adapter entry, its value can be "com.hp.amadapter.AMPushAdapter" or "com.hp.amadapter.ucmdb1021.AMPushAdapter". "com.hp.amadapter.AMPushAdapter" is compatible with UCMDB versions lower than 10.21, but it cannot use new functions in higher

Setting	Default	Description
		UCMDB versions. The value "com.hp.amadapter.ucmdb1021.AM PushAdapter" can be used with UCMDB 10.21 and higher versions. It supports new UCMDB functions and a new interface that collects more statistical information when fatal error occurs.
parallel.thread.pool.size	8	The amount of threads used in Parallel Push Mode. The more threads, the more CPU used. Increasing the pool size may lower performance.
mapping.size.fuse	100000	The maximum number of records the AM Connector may to retrieve in <dynamic_mapping> .
transaction.deadlock.max.retry.count	15	The maximum number of attempts the AM Connector reconnects to the database when deadlock occurs, network is broken, or database is down. The interval time is 1.5 ~ 3 seconds.
am.reconnect.db.code	<pre> <!--MSSQL--> SQLState:08001; SQLState:08S01; SQLState:01000; <!--ORA--> ORA-01033; ORA-03113; ORA-12543; ORA-12541; ORA-12518; <!--DB2--> SQL30080N; SQL30081N; </pre>	<p>The error codes are used to check if the database exception is raised by network or database instance down. The AM Connector will try to reconnect to the database if one of these error codes is received.</p> <p>You can also add other error codes to meet your requirements.</p>
am.ignore.error	true	<p>This option makes the adapter ignore the push errors, thus allowing he push job to continue.</p> <p>Notice that the following types of errors will not be ignored even if the option is set to "true".</p>

Setting	Default	Description
		<ul style="list-style-type: none">◦ Job timeout error, this error is controlled by UCMDB platform.◦ Network issue, this error makes adapter unable to connect to the Asset Manager database for push.◦ Database issue, this error makes adapter unable to push data to the Asset Manager database.
am.pushbackid.ignore.error	false	This option defines whether to ignore the error when pushing back UCMDB global ID to AM in a population job.

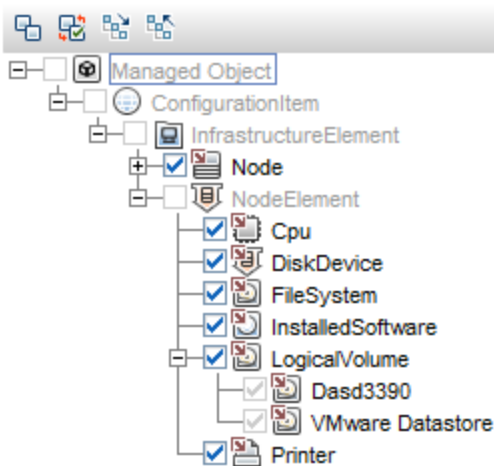
Configure Federation CI types

In **adapterInfo > adapter-capabilities > support-federated-query > supported-classes > supported-class**, all supported CI types are defined in the out-of-box federation mapping.

- node
- installed_software
- Cpu
- disk_device
- file_system
- logical_volume
- Printer

You can modify them to enable or disable the supported CI Types as shown in the **Integration point > Federation** pane.

Supported and Selected CI Types



How to Customize an Existing Mapping

AM Generic Adapter allows you to modify existing out-of-box mappings with Mapping UI. You can edit the mapping XML file directly.

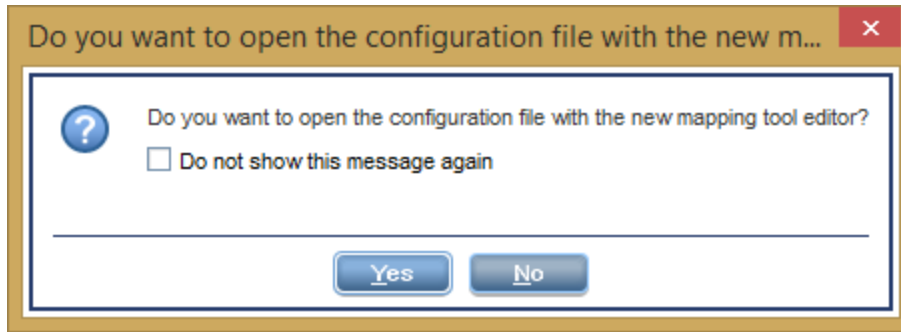
To modify a mapping file using the visual mapping tool

Go to **UCMDB Client > Data Flow Management > Adapter Management > AMGenericAdapter > Configuration Files**. Mapping UI can open those XML mapping files in the following sub folders in the AMGenericAdapter/mappings folder:

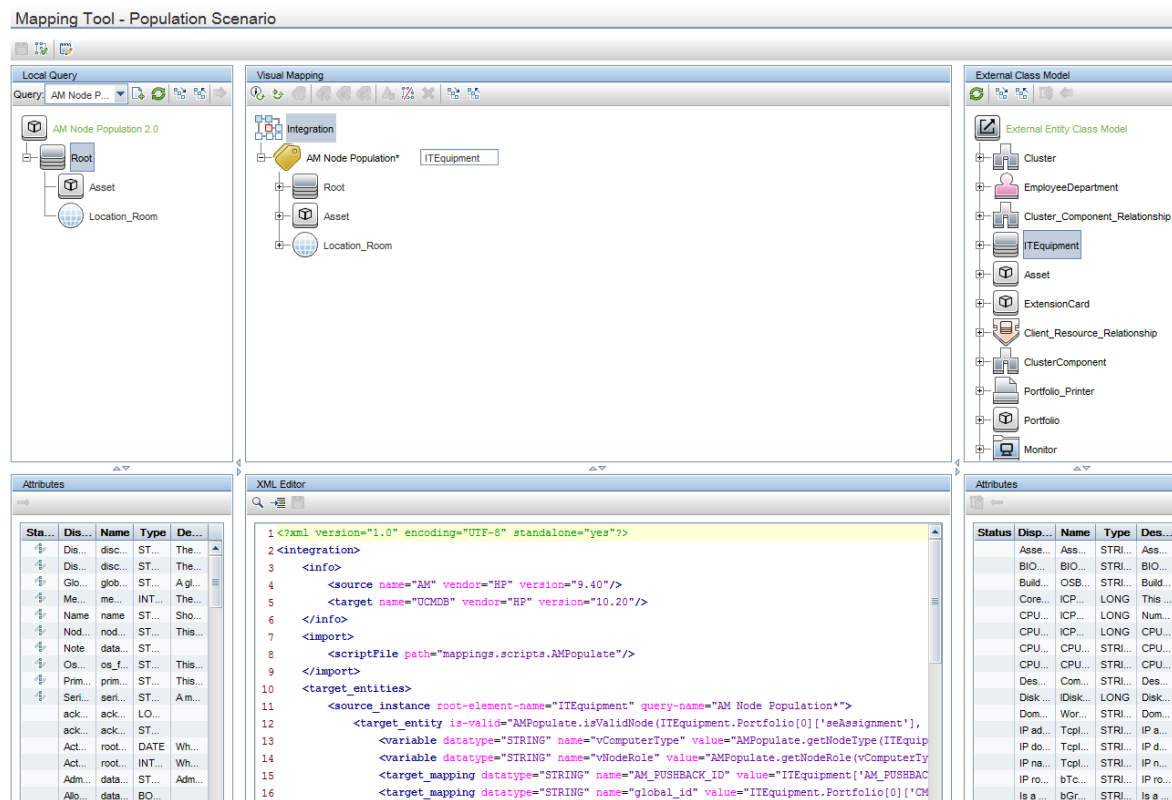
- Federation
- Population
- Push

[AMGenericAdapter/mappings/federation/Printer/AM Printer Federtion.xml](#)
[AMGenericAdapter/mappings/federation/Printer/AM Printer Node Relations Federtion.xml](#)
[AMGenericAdapter/mappings/population/AM BusinessElement Population.xml](#)
[AMGenericAdapter/mappings/population/AM BusinessElement Relations Population.xml](#)
[AMGenericAdapter/mappings/population/AM Interface Population.xml](#)
[AMGenericAdapter/mappings/population/AM Location Population.xml](#)
[AMGenericAdapter/mappings/population/AM Node Population.xml](#)
[AMGenericAdapter/mappings/population/AM Printer Population.xml](#)
[AMGenericAdapter/mappings/push/business-element/AM BE Relations Push.xml](#)
[AMGenericAdapter/mappings/push/business-element/AM BusinessElement Push.xml](#)

When you click a mapping XML file, a dialog pops up. You can choose to use new mapping tool editor to edit this mapping file.



If you click **Yes**, the mapping XML file will be opened in the graphic mapping UI. The following screenshot shows the AM Node Population.xml file opened in the mapping UI.



For details about the mapping UI, please see Data Flow Management Guide.

If you want to populate one more attribute AssetStatus of Asset from AM to UCMDB, you need to complete the following steps.

1. If the attribute in AM is not exposed from the AM database to the AM Class Model, add it to the am-entity-config.xml file. For example, the Field1 attribute is not part of the out-of-box AM Class Model, you need to add it to the amAsset table in the am-entity-config.xml file.

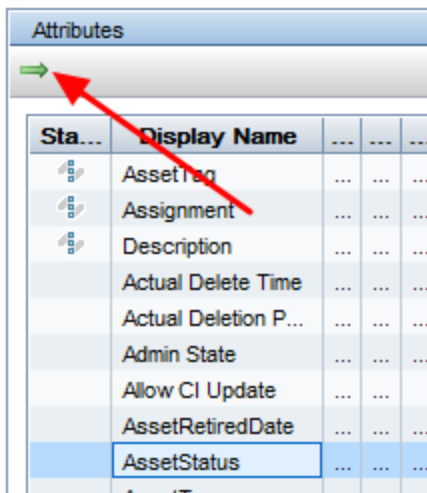
```
<am-table name="amAsset">

<attributes>AssetTag,sMaxCnxCount,SerialNo,Description,Status,AssetStatus</attributes>

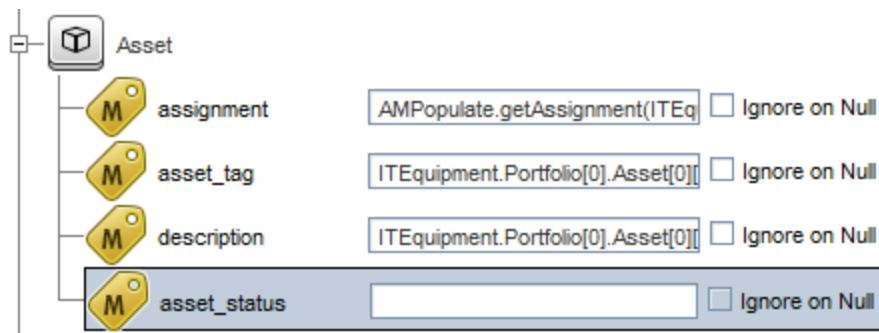
    <links>Model,PortfolioItem</links>
</am-table>
```

In this example, the attribute in AM to map to the AssetStatus attribute is status. As it is already in the out-of-box AM Class Model, you do not need to modify the am-entity-config.xml file.

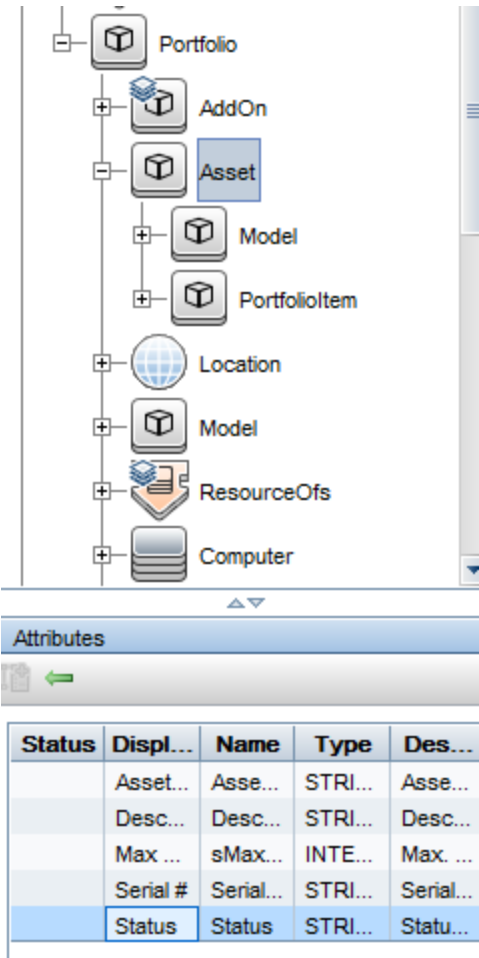
2. In the Local Query pane, select the Asset node.
3. Select the AssetStatus attribute and add it to the mapping definition.



4. In the mapping definition pane, you will see asset_status is presented under the Asset node.



5. From the right side External Class Model, select Asset, find the attribute Status, then click the Add button, or drag it to asset_status in middle directly.



6. The mapping is created.



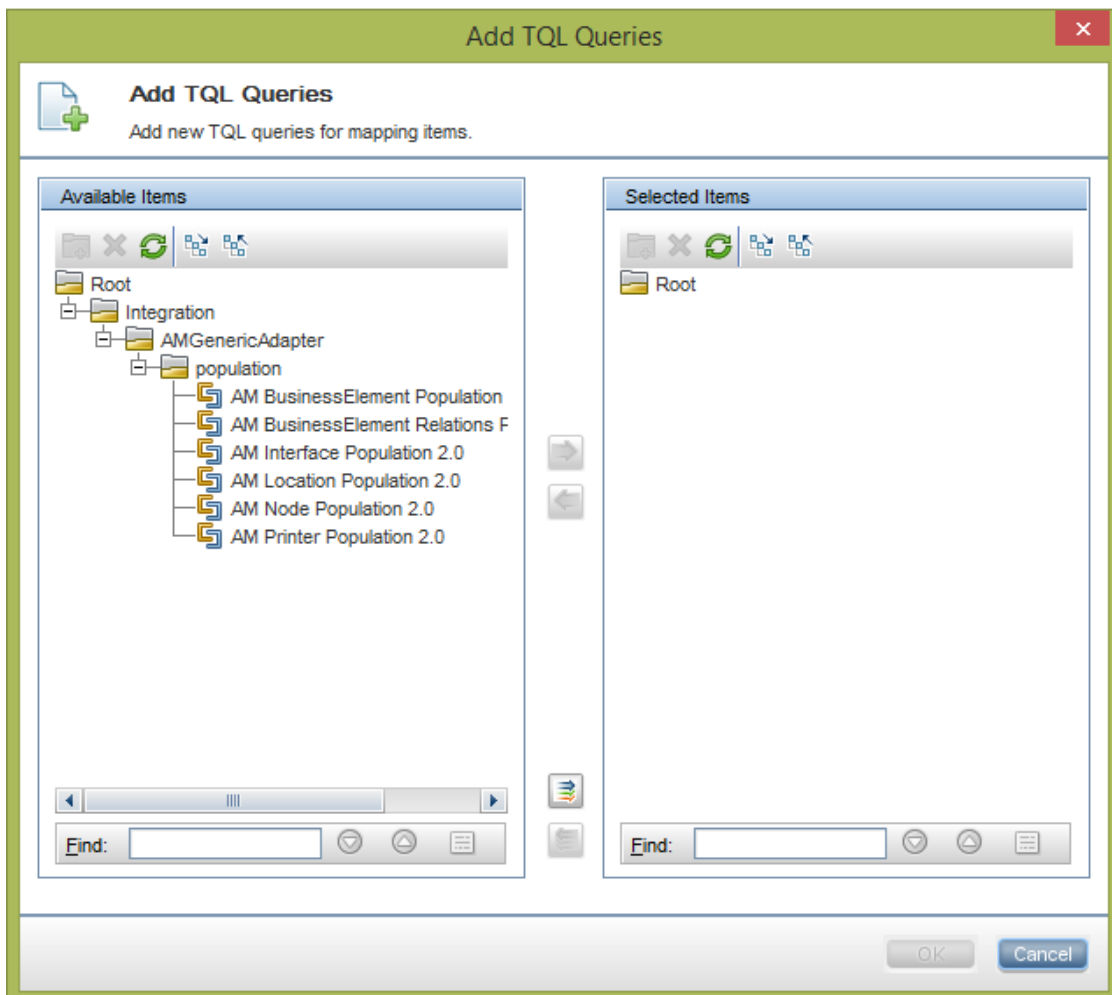
You can also populate the value part of the mapping by using the auto-complete feature in the mapping UI. For example, type ITEquipment, it will show all of the links and attributes of ITEquipment. After select Portfolio, it will show portfolio's links and attributes. Continue operating this way, you will be able to get the same text in the value field.

For more information about Mapping UI features, see Data Flow Management Guide.

How to Add a New Mapping to the Integration

To create a mapping file using the visual mapping tool, you need to complete the following steps.

1. Go to **UCMDB Client > Data Flow Management > Adapter Management > AMGenericAdapter**, click the Create New Resource icon.
2. Click New Configuration file.
3. In the popup window, Package is already specified to AMGenericAdapter. Type a name for mapping file, for example, AMGenericAdapter/mappings/population/AM example Population.xml.
4. In the window asking you if you want to open it in the mapping tool editor, click Yes.
5. After the Mapping UI is loaded, you need to add a TQL Query in the Local Query pane. If you are editing a push mapping, it displays all TQL queries when adding TQL query. If you edit population and federation mapping, it displays those TQLs defined in the supported-population-queries and supported-federation-queries in the am-populate-config.xml.



6. Other steps are same as editing an existing mapping.

For more information, see *UCMDB Data Flow Management Guide*.

How to Populate Asset Manager Contract

Contract management is one of the modules built in Asset Manager. The out-of-box Asset Manager to UCMDB data population content package does not contain the data mapping for contract. This sample is to introduce the procedure of mapping the contract data from Asset Manager to UCMDB from scratch.

Step 1. Define the Contract Entity

1. In the UCMDB client, go to **Data Flow Management > Adapter Management**.

2. Expand **Packages > AMGenericAdapter > Configuration Files**, open AMGenericAdapter/config/am-entity-config.xml.

3. Append the following lines to the am-tables section.

```
<am-table name="amContract" icon-type="contract">
  <attributes>Ref,seType,Purpose,ContractNo,seStatus</attributes>
  <links></links>
</am-table>
```

4. Append the following line to the am-entities section.

```
<am-root-entity name="Contract" table="amContract" inherit-from-table="true"/>
```

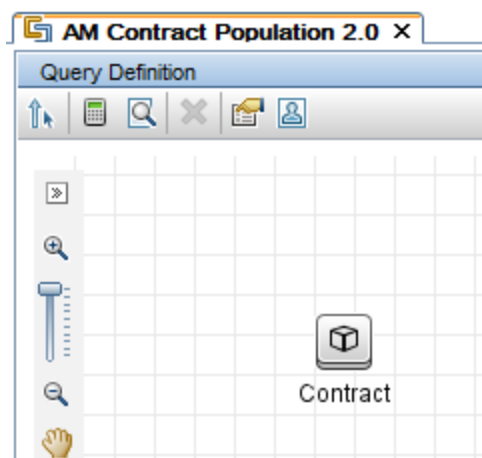
5. Save the changes.
6. Open the AMGenericAdapter/config/am-populate-config.xml file.
7. Append the following lines to the am-populations section.

```
<am-population entity-name="Contract">
  <query-condition><![CDATA[ Purpose <> " ]]></query-condition>
</am-population >
```

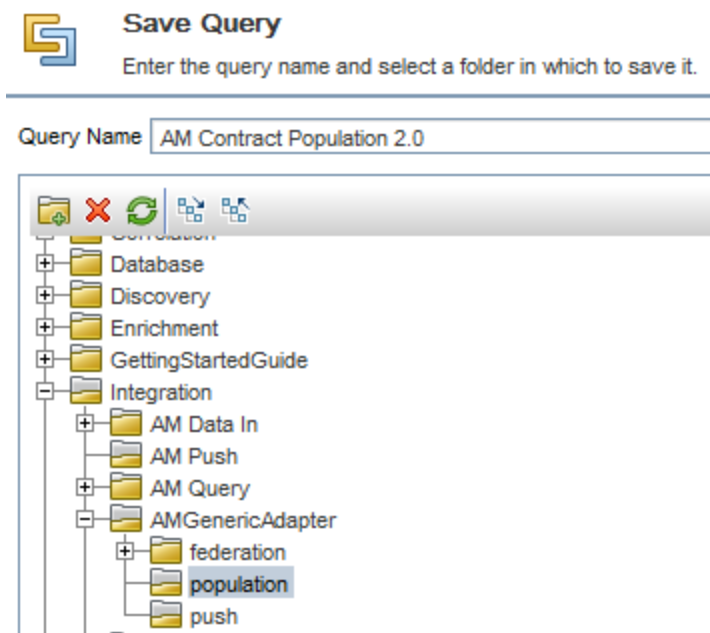
8. Save the changes.

Step 2. Create a TQL Query

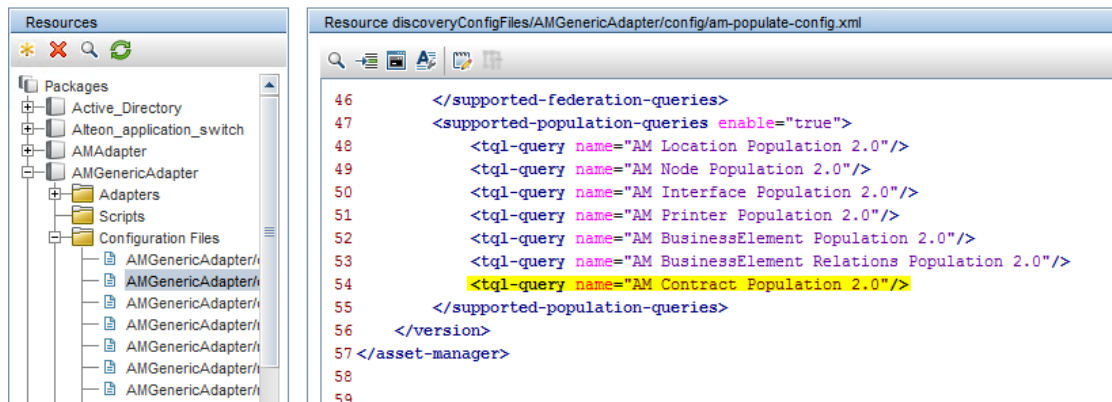
1. In Model Studio, create a new TQL query "AM Contract Population 2.0" with Contract CI type as its only content.



2. Save the query under Integration/AMGenericAdapter/population.

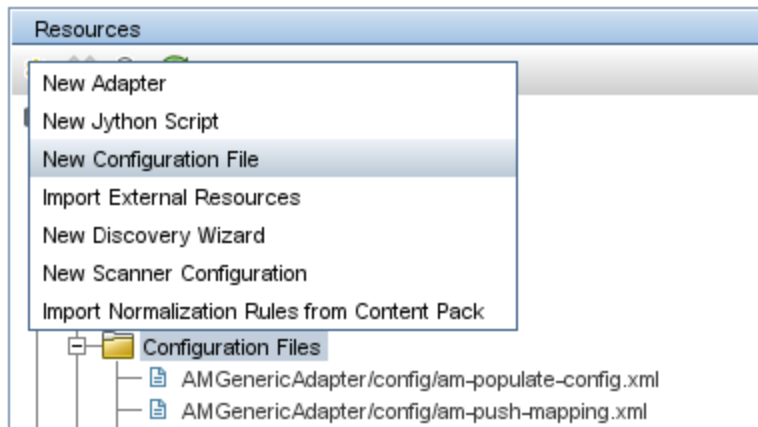


3. Add the TQL to the supported population queries in the population configuration file: am-populate-config.xml.



Step 3. Create a Data Mapping File

1. In Adapter Management, expand **Packages > AMGenericAdapter > Configuration Files**.
2. Click the **Create new resource** button and select **New Configuration File**.



3. In the pop-up window, fill in the fields with the following values.
Name: AMGenericAdapter/mappings/population/AM Contract Population.xml
Package: AMGenericAdapter
4. In the pop-up window, click **Yes** to use Mapping UI to edit mapping file.
5. In the mapping UI, create a target mapping for the document_reference attribute. Then, you should get the following content of the mapping file.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<integration>
  <info>
    <source name="AM" vendor="HP" version="9.41"/>
    <target name="UCMDB" vendor="HP" version="10.20"/>
  </info>
</integration>
```

```
<import>
  <scriptFile path="mappings.scripts.AMPopulate"/>
</import>
<target_entities>
  <source_instance root-element-name="Contract" query-name="AM Contract
Population 2.0">
    <target_entity name="Contract" type="'license_contract'">
      <target_mapping datatype="STRING_LIST" ignore-on-null="false"
name="document_reference" value="AMPopulate.toStringList(Contract['Ref'])"/>
    </target_entity>
  </source_instance>
</target_entities>
</integration>
```

Note: You can also overwrite the content of the AM Contract Population.xml file with the above content in the text edit mode.

Step 4. Specify Sub CI Type for Contract

As the Contract CI type is abstract, which cannot be instantiated. A sub CI type must be specified in the target mapping to inform UCMDB of what exact CI should be created. In the above mapping file, it explicitly sets the CI type to `license_contract`. In Asset Manager, there are a number of contract types that correspond to the sub type of Contract in UCMDB. You can use the type attribute of the `target_entity` element to specify the exact type of the contract according to the contract type in Asset Manager.

1. Go to **Data Flow Management > Adapter Management**.
2. Expand **Packages > AMGenericAdapter > Configuration Files** and open the file `AMGenericAdapter/mappings/scripts/AMPopulate.groovy`.
3. Add the following line below the line 24.

```
private static Map<Integer, String> contractTypes = new HashMap<Integer,
String>();
```

4. Add the following lines to the end of the static section.

```
contractTypes.put("0", "contract");contractTypes.put("1",
"lease_contract");
contractTypes.put("2", "lease_contract");
contractTypes.put("3", "insurance_contract");
contractTypes.put("4", "maintenance_contract");
contractTypes.put("5", "license_contract");
contractTypes.put("6", "procurement_contract");
contractTypes.put("7", "service_level_agreement");
contractTypes.put("8", "service_level_agreement");
```

```
contractTypes.put("9", "operational_level_agreement");
contractTypes.put("10", "operational_level_agreement");
contractTypes.put("11", "procurement_contract");
```

5. Create a groovy function to get the sub CI type of the contract corresponding to the AM contract type.

```
public static String getContractType(String seType) {
    return contractTypes.get(seType);
}
```

6. Update the mapping file to replace the value of the type attribute with a call to the groovy function.

```
<target_entity name="Contract" type="AMPopulate.getContractType(Contract
['seType'])">
```

7. Add a variable to retrieve the data of the seType field.

```
<variable name="vType" datatype="STRING" value="Contract['seType']"/>
```

8. Add a mapping for the contract_ref attribute which is required in the identification rule of some sub type of Contract.

```
<target_mapping datatype="STRING" name="contract_ref" value="Contract['Ref']"/>
```

9. In the end, you should have the following content in the mapping file.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<integration>
  <info>
    <source name="AM" vendor="HP" version="9.41"/>
    <target name="UCMDB" vendor="HP" version="10.20"/>
  </info>
  <import>
    <scriptFile path="mappings.population.scripts.AMPopulate"/>
  </import>
  <target_entities>
    <source_instance root-element-name="Contract" query-name="AM Contract
Population 2.0">
      <target_entity name="Contract" type="AMPopulate.getContractType
(Contract['seType'])">
        <variable name="vType" datatype="STRING" value="Contract
['seType']"/>
        <target_mapping datatype="STRING_LIST" ignore-on-null="false"
name="document_reference" value="AMPopulate.toStringList(Contract['Ref'])"/>
        <target_mapping datatype="STRING" name="contract_ref"
value="Contract['Ref']"/>
      </target_entity>
    </source_instance>
  </target_entities>
</integration>
```

Step 5. Map More Fields

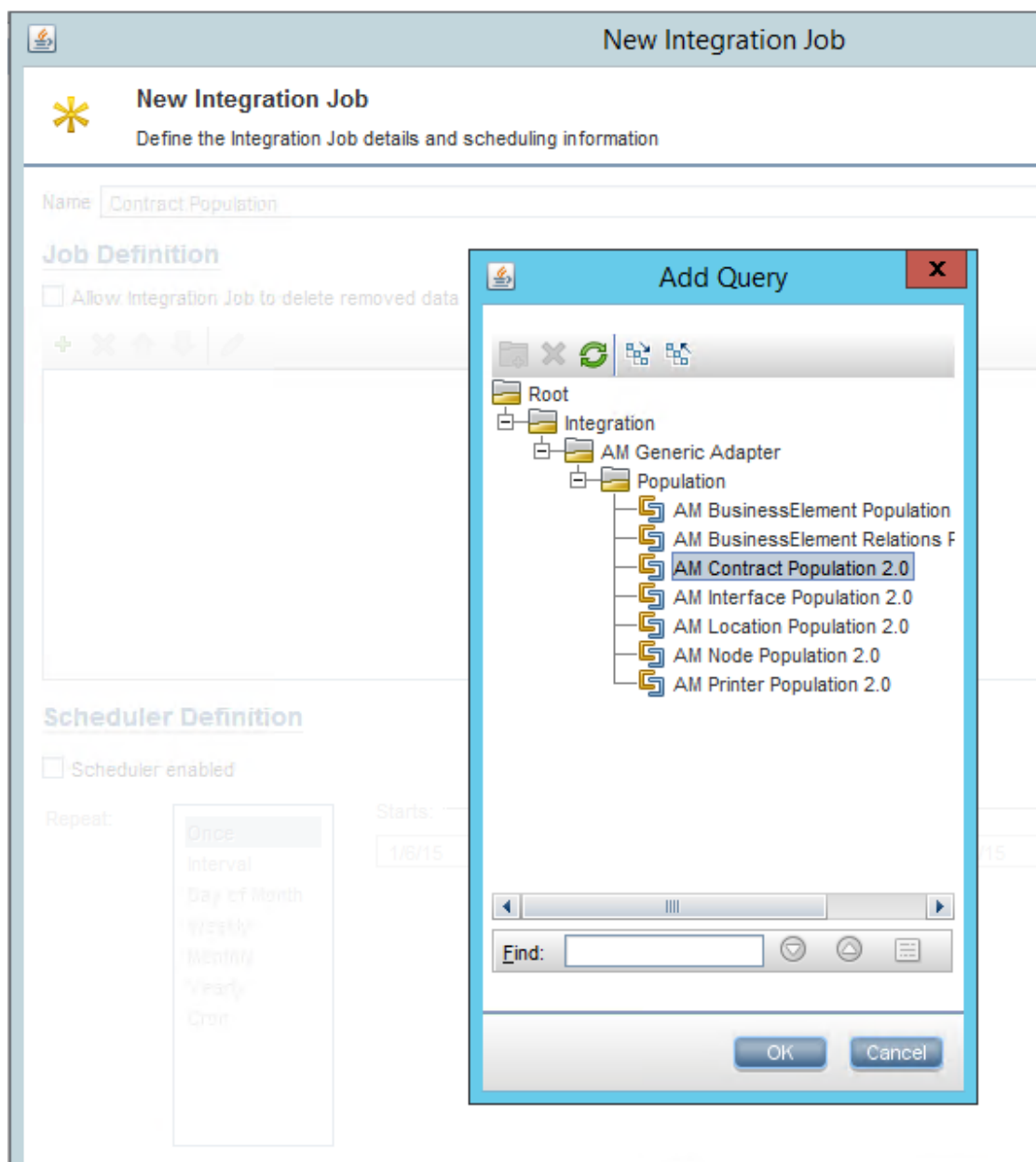
Besides the mapping for the document_reference and contract_ref fields, you can map more fields as needed. Below is the content of the mapping file with more field mappings.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<integration>
  <info>
    <source name="AM" vendor="HP" version="9.41"/>
    <target name="UCMDB" vendor="HP" version="10.20"/>
  </info>
  <import>
    <scriptFile path="mappings.scripts.AMPopulate"/>
  </import>
  <target_entities>
    <source_instance root-element-name="Contract" query-name="AM Contract
Population 2.0">
      <target_entity name="Contract" type="AMPopulate.getContractType
(Contract['seType'])">
        <variable datatype="STRING" name="vType" value="Contract
['seType']"/>
        <target_mapping datatype="STRING_LIST" ignore-on-null="false"
name="document_reference" value="AMPopulate.toStringList(Contract['Ref'])"/>
        <target_mapping datatype="STRING" name="contract_ref"
value="Contract['Ref']"/>
        <target_mapping datatype="STRING" name="description"
value="Contract['Purpose']"/>
        <target_mapping datatype="STRING" name="name" value="Contract
['Ref']"/>
        <target_mapping datatype="STRING" name="data_note"
value="Contract['ContractNo']"/>
      </target_entity>
    </source_instance>
  </target_entities>
</integration>
```

Step 6. Create a population job

In the Integration Studio, locate the integration point you created to connect to Asset Manager. Follow these steps to create the job for contract data population.

1. Switch to the **Population** tab.
2. Click the **New Integration Job** button.
3. Add the AM Contract Population 2.0 query to the **Job Definition**.



4. Name the job and click **OK** to save the job.

Step 7. Execute the Job

Select the newly created job and click the **Full Synchronization** button to start the job. After the job completes, the statistics pane displays the result of the job execution.

Statistics Query Status					
Filter: Time Range[From now(10/29/2014 04:24:42 PM)]					
CIT	Created	Updated	Deleted	Failed	
InsuranceContract	3	0	0	0	
LeaseContract	6	0	0	0	
LicenseContract	6	0	0	0	
MaintenanceContract	13	0	0	0	
OperationalLevelAgreement	1	0	0	0	
ProcurementContract	5	0	0	0	
ServiceLevelAgreement	2	0	0	0	
Total	36	0	0	0	

You can double-click each line of the statistics pane to view the details of the CIs.

Created by DS_GAAM941Local_Contract Populate 7 - CIs							
Show CI instances of: MaintenanceContract (13)							
Updated By	DocumentReference	Create Time	Owner Tenant	ContractRef	Description	Last Acc	
GAAM941Local: DS_GAAM941Local_Contract Populate 7	[C001001]	Wed Oct 29 2014 04:33 PM CST	System Default Tenant	C001001	Warranty extension	Wed Oct 29 2014	
GAAM941Local: DS_GAAM941Local_Contract Populate 7	[C001002]	Wed Oct 29 2014 04:33 PM CST	System Default Tenant	C001002	Warranty extension	Wed Oct 29 2014	
GAAM941Local: DS_GAAM941Local_Contract Populate 7	[C001003]	Wed Oct 29 2014 04:33 PM CST	System Default Tenant	C001003	Warranty extension	Wed Oct 29 2014	
GAAM941Local: DS_GAAM941Local_Contract Populate 7	[C001004]	Wed Oct 29 2014 04:33 PM CST	System Default Tenant	C001004	Warranty extension	Wed Oct 29 2014	
GAAM941Local: DS_GAAM941Local_Contract Populate 7	[C001005]	Wed Oct 29 2014 04:33 PM CST	System Default Tenant	C001005	Warranty extension	Wed Oct 29 2014	
GAAM941Local: DS_GAAM941Local_Contract Populate 7	[C001006]	Wed Oct 29 2014 04:33 PM CST	System Default Tenant	C001006	Warranty extension	Wed Oct 29 2014	
GAAM941Local: DS_GAAM941Local_Contract Populate 7	[C001007]	Wed Oct 29 2014 04:33 PM CST	System Default Tenant	C001007	Warranty extension	Wed Oct 29 2014	
GAAM941Local: DS_GAAM941Local_Contract Populate 7	[C001008]	Wed Oct 29 2014 04:33 PM CST	System Default Tenant	C001008	Warranty extension	Wed Oct 29 2014	
GAAM941Local: DS_GAAM941Local_Contract Populate 7	[C001009]	Wed Oct 29 2014 04:33 PM CST	System Default Tenant	C001009	Warranty extension	Wed Oct 29 2014	
GAAM941Local: DS_GAAM941Local_Contract Populate 7	[C001010]	Wed Oct 29 2014 04:33 PM CST	System Default Tenant	C001010	Warranty extension	Wed Oct 29 2014	
GAAM941Local: DS_GAAM941Local_Contract Populate 7	[C001011]	Wed Oct 29 2014 04:33 PM CST	System Default Tenant	C001011	Warranty extension	Wed Oct 29 2014	
GAAM941Local: DS_GAAM941Local_Contract Populate 7	[DEMO-MAI3]	Wed Oct 29 2014 04:33 PM CST	System Default Tenant	DEMO-MAI3	On-site maintenance	Wed Oct 29 2014	
GAAM941Local: DS_GAAM941Local_Contract Populate 7	[MTHPHWMMAINT25177]	Wed Oct 29 2014 04:33 PM CST	System Default Tenant	MTHPHWMMAINT25177	HP On-site maintenance	Wed Oct 29 2014	

How to Set up Federation

AM Generic Adapter supports federation feature. For more information, refer to *Universal UCMDB Developer Reference Guide, Chapter: How to Set Up Federation*.

Federation mappings have the same design as population, it is the same to retrieve AM data from AM entity and to use same query condition definition.

When you create TQL queries for federation in AM Generic Adapter, you need to save them in the Integration/AMGenericAdapter/federation folder.

How to set up integration in a multi-tenant environment

When a single instance of software runs on a server, serving multiple client organizations (also known as tenants), the environment is called a multi-tenant environment. Both Asset Manager and UCMDB support multi-tenancy. UCMDB also has a legacy multi-customer setting that is different from the multi-tenant feature described in this document.

When UCMDB enables multiple tenants, each CI belongs to a particular tenant, known as the owner tenant. Other tenants can be designated as consumer tenants.

When AM enables multiple tenants, a record belonging to the users whose Primary Tenant is the same tenant linked to this record. It is read-only to other users with Viewable Tenants is the same tenant linked to this record.

In AM adapters, multi-tenant solution is implemented based on:

- 'Owner Tenant' of CI in UCMDB
- Tenant.Code of record in AM for population; ITenantId of record in AM for push

	AM	UCMDB
Push	ITenantId (The key of Tenant)	TenantOwner (Owner Tenant)
Population	Tenant.Code	TenantOwner (Owner Tenant)

Note: This feature can also be applied in AM Push Adapter, to do this, follow the instructions for Push in this section.

Quick start configuration

Options for enabling multi-tenant

Multi-tenant job should be manually specified for push or population in AM adapter's configuration file.

By default, in AMGenericAdapter\mappings\scripts\AMUtils.groovy, multi-tenant is disabled:

- `public static final boolean ENABLE_PUSH_MT = false;`
- `public static final boolean ENABLE_POP_MT = false;`

To enable multi-tenant, set `ENABLE_PUSH_MT` (push) or `ENABLE_POP_MT` (population) to true.

Map tables for mapping tenants between AM and UCMDB

There are two map tables defining the mappings between AM and UCMDB.

In `AMGenericAdapter\mappings\scripts\AMUtils.groovy`:

- `public static Map<String, String> mtMapForPop = new HashMap<String, String>();`
- `public static Map<String, String> mtMapForPush = new HashMap<String, String>();`

Map table for push

- `mtMapForPush.put("UCMDBTENANT1", "AMTENANT1");`
- `mtMapForPush.put("UCMDBTENANT2", "AMTENANT2");`

Note: AMTENANT1 is Tenant Code, UCMDBTENANT1 is TenantOwner.

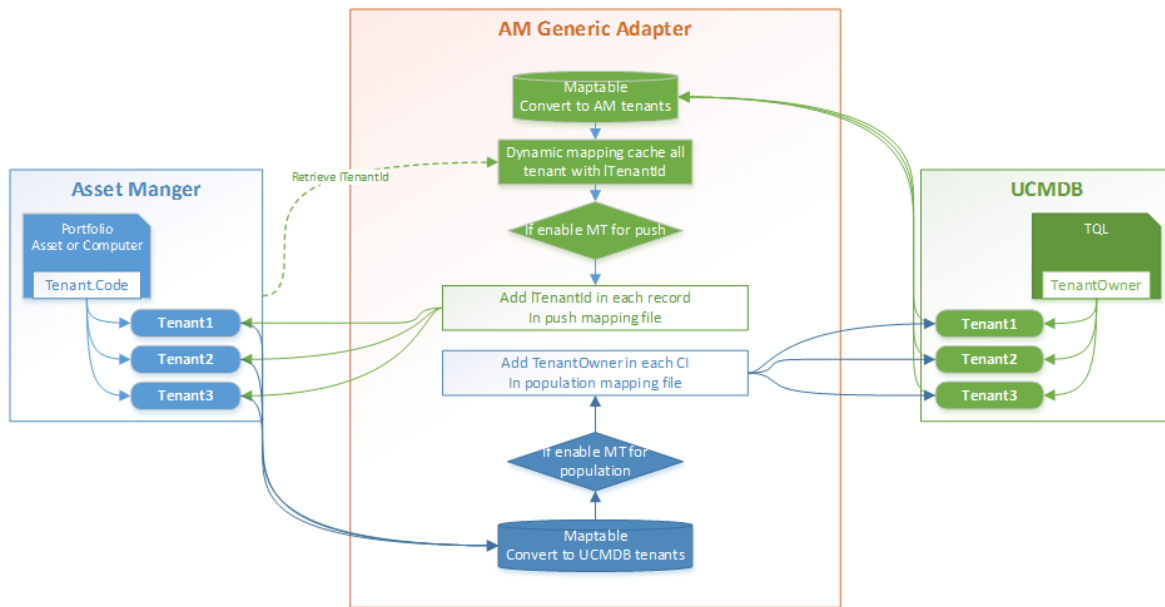
Maptable for population

- `mtMapForPop.put("AMTENANT1", "UCMDBTENANT1");`
- `mtMapForPop.put("AMTENANT2", "UCMDBTENANT2");`

Note: AMTENANT1 is Tenant Code, UCMDBTENANT1 is TenantOwner.

Solution design

This section demonstrates the multi-tenant implementation solution for both push and population parts of AM Generic Adapter.



For push

- Add the 'TenantOwner' property to the Root CI in each out-of-box TQL. If the Root CI is a relationship, add the 'TenantOwner' property to the source CI. For example, A -> B, Root CI is the relationship ->, so add 'TenantOwner' to A.
- At the beginning of each mapping, add dynamic mapping to cache all AM tenant's code and key (query AM database only once for each mapping).

```
<dynamic_mapping name="TenantIdByCode" keys-unique="true">
  <map_property property-name="AQLQuery" datatype="STRING" property-
value="SELECT Code, lTenantId FROM amTenant"/>
</dynamic_mapping>
```

- Query the specified AM tenant(Tenant.Code) mapped to the UCMDB tenant(TenantOwner) from the map table.

```
mtMapForPush.put("UCMDBTENANT1", "AMTENANT1");
mtMapForPush.put("UCMDBTENANT2", "AMTENANT2");
```

- Define a variable to return the key (lTenantId) of the AM tenant.

```
<variable datatype="INTEGER" name="vTenantId" value="AMPush.getAMTenantID
(DynamicMapHolder, Root['TenantOwner'])"/>
```

- Add lTenantId under each target_entity (AM record) that needs a tenant specified.

```
<target_mapping datatype="INTEGER" name="lTenantId" ignore-on-null="true" is-
valid="AMPush.ENABLE_PUSH_MT" value="vTenantId" />
```

- User needs to modify the global variable ENABLE_PUSH_MT from false (default) to true, and then

start the push job.

```
public static final boolean ENABLE_PUSH_MT = true;
```

For population

- Add Tenant link on each AM-table in AM entity configuration file am-entity-config.xml.
- Query the specified UCMDB tenant (TenantOwner) mapped to the AM tenant (Tenant.Code) from the map table.

```
mtMapForPop.put("AMTENANT1", "UCMDBTENANT1");  
mtMapForPop.put("AMTENANT2", "UCMDBTENANT2");
```

- Add TenantOwner under each target_entity (UCMDB CI) that needs a tenant specified.

```
<target_mapping datatype="STRING" name="TenantOwner" is-  
valid="AMPopulate.ENABLE_POP_MT" value="AMPopulate.getOwnerTenant  
(ITEquipment.Tenant[0]['Code'])"/>
```

- User needs to modify the global variable ENABLE_POP_MT from false (default) to true, and then start the population job.

```
public static final boolean ENABLE_POP_MT = true;
```

Tailoring

Tenants can be merged

Define the mappings in map tables. Several tenants are allowed to be mapped to one tenant, for example: AMTENANT1 and AMTENANT2 map to UCMDBTENANT1.

```
mtMapForPop.put("AMTENANT1", "UCMDBTENANT1");  
mtMapForPop.put("AMTENANT2", "UCMDBTENANT1");
```

Push model tenant (default is shared)

If in your environment, model is also managed by each tenant, you can add ITenantId under <target_entity name="Model">.

```
<target_mapping datatype="INTEGER" name="lTenantId" ignore-on-null="true" is-  
valid="AMPush.ENABLE_PUSH_MT" value="vTenantId" />
```

Push tenant by other fields or logic (default is by Owner Tenant)

If in your environment, UCMDB is not managed by multi-tenant and AM is managed by multi-tenant, and the CI can be separated to AM tenants by following some defined rules, for example:

domain_name of Node, **Prefix** Node Name, or **primary_ip_address** group of Node

You can define your own function (for example: `getUCMDBTenantOwnerByDomain`) map to specify the UCMDB tenant (virtual TenantOwner), and then retrieve the key of AM Tenant(ITenantId) by virtual TenantOwner.

```
<variable datatype="STRING" name="vTenantOwner"
value="AMPush.getUCMDBTenantOwnerByDomain(Root['domain_name'])"/>
<variable datatype="INTEGER" name="vTenantId" value="AMPush.getAMTenantID
(DynamicMapHolder, vTenantOwner)"/>
```

Populate tenant by other fields or logic (default is by Tenant.Code)

If in your environment, AM is not managed by multi-tenant, but UCMDB is managed by multi-tenant, and the AM records can be separated to UCMDB tenants by following some defined rules, for example:

TcpIpDomain of amComputer, **Prefix** of **TcpIpHostName**, or **TcpIpAddress** group of amComputer

You may define your own function (for example: `getAMTenantCodeByDomain`) map to specify the AM tenant (virtual Tenant Code), and then convert to UCMDB TenantOwner by virtual Tenant Code.

```
<variable datatype="STRING" name="vTenantCode"
value="AMPopulate.getAMTenantCodeByDomain(ITEquipment['TcpIpDomain'])"/>
<target_mapping datatype="STRING" name="TenantOwner" is-valid="AMPopulate.ENABLE_
POP_MT" value="AMPopulate.getOwnerTenant(vTenantCode)"/>
```

Do tenant mapping when creating a new push or population for UCMDB CIs or AM records

In this solution, each entity's tenant should be manually specified, either AM ITenantId or UCMDB TenantOwner. They are enabled/disabled by is-valid and global variable.

If you transfer a new entity between UCMDB and AM, we recommend that you copy the mappings for enabling multi-tenant in future.

Push:

```
<target_mapping datatype="INTEGER" name="ITenantId" ignore-on-null="true" is-
valid="AMPush.ENABLE_PUSH_MT" value="vTenantId" />
```

Population:

```
<target_mapping datatype="STRING" name="TenantOwner" is-valid="AMPopulate.ENABLE_
POP_MT" value="AMPopulate.getOwnerTenant(ITEquipment.Tenant[0]['Code'])"/>
```

Performance Tuning and Best Practice

This section shows the best practices to optimize the performance.

Chunk Size

When pushing CIs from UCMDB to Asset Manager, the Generic Adapter framework retrieves data from a TQL query. The CIs are loaded in a chunk with respect to the chunk size setting. Adjusting the chunk size has impact on the performance of data push. By default, the AM Generic Adapter chunk size for push is 4000. You can adjust it by changing the replication.chunk.size setting on the AM Generic Adapter.

To reduce the communication between adapter and UCMDB server, you may increase the chunk size.

To avoid task timeout, you may decrease the chunk size.

```
<adapter-settings>
  <adapter-setting name="use.optimized.push">true</adapter-setting>
  <adapter-setting name="replication.chunk.size">1000</adapter-setting>
  <adapter-setting name="replication.chunk.root.limit">850</adapter-setting>
  <adapter-setting name="is.new.generic.push.adapter">true</adapter-setting>
  <adapter-setting name="PushConnector.class.name">com.hp.ucmdb.adapters.ampush.AMPi
  <adapter-setting name="parallel.thread.pool.size">8</adapter-setting>
  <adapter-setting name="mapping.size.fuse">100000</adapter-setting>
  <adapter-setting name="transaction.deadlock.max.retry.count">3</adapter-setting>
  <adapter-setting name="shared.class.loader.parent">amVersion</adapter-setting>
</adapter-settings>
```

Time out

There are some global settings of timeout which affect AM Generic Adapter integration jobs. The following figure shows a screenshot of the relevant timeout settings.

Wa...	Name	Value	Description	Refresh
*	adapter.adhoc.task.remote.action.tim...	360000000	adapter.adhoc.task.remote.action.timeout	Minutes
*	Data Push Job Timeout	360000000	Data Push job timeout (this setting is used when source adapter support changes ...	Minutes
	DB Connection Pool Timeout	1800000	This setting determines how much time a connection can remained opened against ...	Reboot
	Enable session timeout	False	Specifies whether or not a UCMDB Browser session will expire. False (session wil...	Login
	Federation Remote Probe Operation Pi...	20000	Defines the timeout for the probe to take the federation operation in milliseconds	Minutes
	Federation Remote Probe Operation Ti...	180000	Defines the timeout for executing remote federation operations on the probe in millis...	Minutes
*	Integration Point Adapter Ad Hoc Tas...	200000	Defines the time out for probe pickup of remote adapter adhoc tasks operations on ...	Minutes
*	Integration Point Maintenance Actions...	200000	Defines the time out for probe pickup of remote maintenance operation on the prob...	Minutes
*	Integration Point Maintenance Actions...	350000	Defines the time out for executing remote maintenance operation on the probe in mil...	Minutes
	Push Remote Probe Operation Pickup ...	30000	Defines the timeout for the probe to take the push operation in milliseconds	Minutes
*	Push Remote Probe Operation Timeout.	12000000	Defines the timeout for executing remote push operations on the probe in millisecon...	Minutes
	Remote package deploy Timeout	300000	Defines the timeout for deploying a package on a remote data store	Minutes
*	Session Timeout	36000	Session timeout for CMDB operations	Reboot

TQL Tailoring

The CI to be pushed depends on the TQL's conditions and structure. Out-of-box TQLs are designed for out-of-box mappings. If you do not need CI data, for example, the pushed data is for the use of SAM software counter calculation, you can tailor TQL conditions to filter out unrequired data to speed up the push job.

Here are some best practices in tailoring TQL queries.

- Separate Node TQL by sub CI Type, for example, Unix and Windows.
- Separate Node TQL by logical prefix computer name, for example, Name like XXX%.
- Add CPU relation in Host and VM TQL, in order to only push those Host with specific CPU model.
- To avoid pushing CIs which data is incomplete because some UD discover jobs are not completed, add conditions to query only the CIs which required attributes are complete. For example, a condition with Create time Unchanged during hours.

Push Mapping Simplification

The Host Server and VM relation push can be very slow for a high volume of CIs. If you do not need to see the records in the amClientResource table, the Host Server and VM relation mapping can be simplified to only link the host and its VM in AM and do not create client resource and business service.

```
<targetcis>
  <!-- TQL: E1 (Server) ==CompoundLink (Hypervisor)==> E2(VM Machine)-->
  <source_instance_type query-name="AM Host Server And Running VM Relations Push" rc
    <target_ci_type name="VM_ESX_amPortfolio-Client">
      <variable name="vEnd1CompId" datatype="STRING" value="AMPushFunctions.getEnd1Ex
        <variable name="vEnd2CompId" datatype="STRING" value="AMPushFunctions.
          <variable name="vEnd2Id" datatype="STRING" value="vEnd2CompId"/>
          <after-mapping>Logger.debug('vEnd1CompId: ' + vEnd1CompId); Logger.debu
          <!-- Because we want to do an attribute reconciliation from end1, we as
          <target_ci_type name="VM_ESX_amPortfolio-Resource">
            <variable name="vEnd1Id" datatype="STRING" value="vEnd1CompId"/>
          </target_ci_type>
        </target_ci_type>
      </target_ci_type>
    </source_instance_type>
  </targetcis>
```

Global ID

Turn on UCMDB Global ID

Global ID is the recommended reconciliation key for both push and population. It is more reliable and takes less time to reconcile CIs/Assets in the target system.

Configure Push Back ID in population mapping

When IT equipment is populated from AM to UCMDB, it allows AM to save CI's global Id. The global Id will be used in all future population, which helps UCMDB to identify a CI faster than populate CI without global Id.

Replace Global ID reconciliation by normal attributes

The adapter engine can save pushed CI's external ID, for example, a node CI can save amComputer IComputerId in cache. When the push relation is between the node and other CIs, you can use groovy function AMPush.getAMPrimaryID to get the external ID from the pushed node.

The out-of-box push reconciliation design uses external ID for link relation in AM.

```
<target_entity name="PortfolioItem" type="'Sw_comp_amPortfolio'">
  <!-- CI['external_cmdb_id'] returns the String external id of the CI-->
  <variable name="vCompId" datatype="STRING" value="AMPush.getAMPrimaryID(Root.Node[0]['external_id_obj'])"/>
```

Due to issues in the underlying engine, the external ID might become out-of-date or may refer to the incorrect AM objects. In this case, you can use normal attributes reconciliation to reconcile in AM record.

```
<reconciliation>
  <reconciliation-keys>
    <reconciliation-key>Name</reconciliation-key>
    <reconciliation-key>lParentId</reconciliation-key>
  </reconciliation-keys>
</reconciliation>
```

Population Mapping Optimization

AM Generic Adapter population allows populating multiple AM entities and relationship in one mapping file. In practice, we recommend that you map one AM entity in a single data mapping.

For example, you can create a population mapping to convert Node, Location, and Interface to UCMDB. However, it is better to split the mapping to three separate mappings to convert Node, Location, and Interface respectively. In this way, Location should be populated first, then Node, Interface should be populated at last. In the mapping for Node, you need to map the link to Location so that the relation between Node and Location can be created in UCMDB.

We strongly recommend that you only reference Node's Global Id in the Interface population mapping. This will improve reconciliation performance for UCMDB server. In order to use global Id for Node, you must enable the Push Back Id in the Integration point configuration, and also need to configure AM_PUSHBACK_ID in Node population mapping.

```
<target_mapping datatype="STRING" name="AM_PUSHBACK_ID" value="ITEquipment['AM_PUSHBACK_ID']"/>
<target_mapping datatype="STRING" name="global_id" value="ITEquipment.Portfolio[0]['CMDBId']"/>
```

Push Mode

An Asset Manager (AM) Generic Adapter push job can run in three different push modes.

- Non-parallel. In this push mode, a push job runs in a single thread to transfer data from UCMDB to AM one by one in sequence.
- Multi-threading. When a push job runs in multi-threading mode, it creates multiple threads to push data to AM in parallel. Each thread transfers a subset of data to push. This push mode leverages the multi-threading capability of the AM native APIs to increase the throughput of data push. All threads share the same AM native API library.
- Multi-processing. A push job running in this push mode uses a number of AM proxy processes to transfer data to AM in parallel. Each AM proxy process has a dedicated AM native API library to communicate with the AM database.

Push Mode Comparison

The following table outlines the advantages and disadvantages of each push mode.

Push Mode	Advantages	Disadvantages	Recommended for
Non-parallel	<ul style="list-style-type: none">• Stable• Low failure rate• Easy error troubleshooting• Low memory footprint	<ul style="list-style-type: none">• Slow	<ul style="list-style-type: none">• Small databases (less than 1M software installations), delta push mode• Small push probe (Limited memory and CPU power on the push probe)
Multi-processing	<ul style="list-style-type: none">• Fastest mode• Scalable• Less concurrency issues	<ul style="list-style-type: none">• Highest memory footprint• Configuration complexity• Troubleshooting difficulties	<ul style="list-style-type: none">• High-end AM push probe and AM database server (CPU, Memory, disk IOs)• All AM database sizes

Push Mode	Advantages	Disadvantages	Recommended for
Multi-threading	<ul style="list-style-type: none">• Faster than non-parallel• Simple configuration	<ul style="list-style-type: none">• Concurrency issues• High memory footprint• Slower than Multi-processing• Limited AM API multi-threading scalability• Troubleshooting difficulties	<ul style="list-style-type: none">• Hardware of push probe or AM database server not large enough to allow multi-processing push mode• Small and middle AM database sizes (up to 5 M software installs)

How to Set Push Mode

In an AM Push Adapter integration point, you can set the **Parallel Push Mode** field to specify the push mode used to run the adapter in the integration point.

When setting the push mode to multi-threading or multi-processing, in order to avoid deadlock on the AM database side, you need to follow the instructions described in ["Prepare Asset Manager for Parallel Push" on page 1](#).

In addition to the above instructions, you need to complete the following steps to enable the multi-processing push mode.

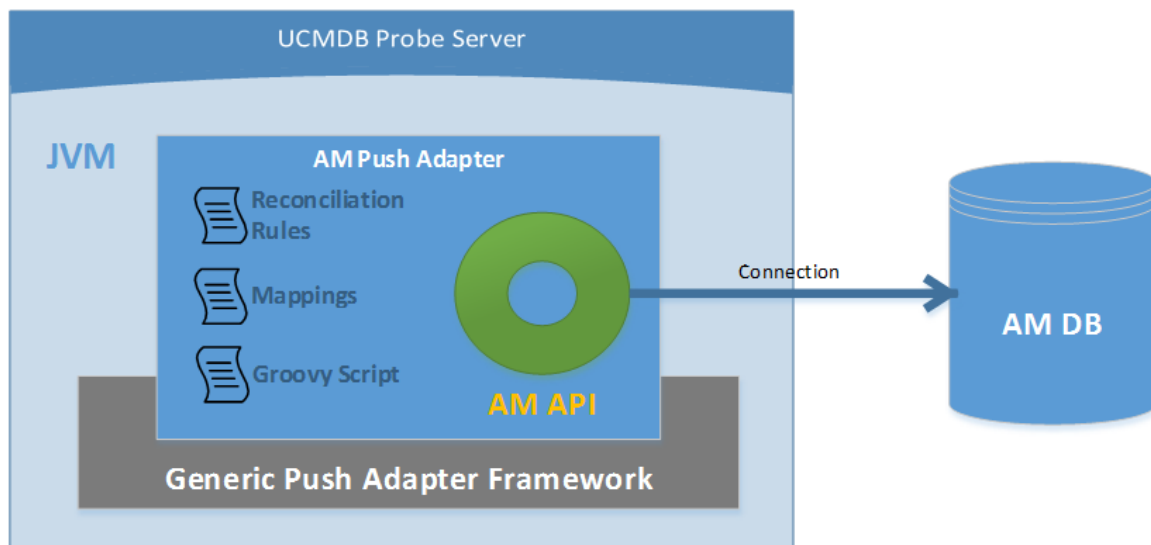
- It is necessary to select the parallel mode again after you deploy the new package that contains multi-processing because the parallel mode option is different from the old version.
- If the Push Adapter connects to an Oracle AM database, the Oracle library 'oci.dll' needs to be placed in the following directories.
 - <UCMDB installation path>\DataFlowProbe\runtime\probeGateway\discoveryResources\AMGenericAdapter\amVersion\<AM version>
 - <UCMDB installation path>\DataFlowProbe\lib

Note: Sufficient memory should be reserved outside the JVM space of the probe server.
Typically:

- Non-parallel mode: 200 MB.
- Multi-processing mode: 200 MB for each parallel process
- Multi-threading mode: 150 MB for each thread.

Non-parallel Push Mode

Architecture



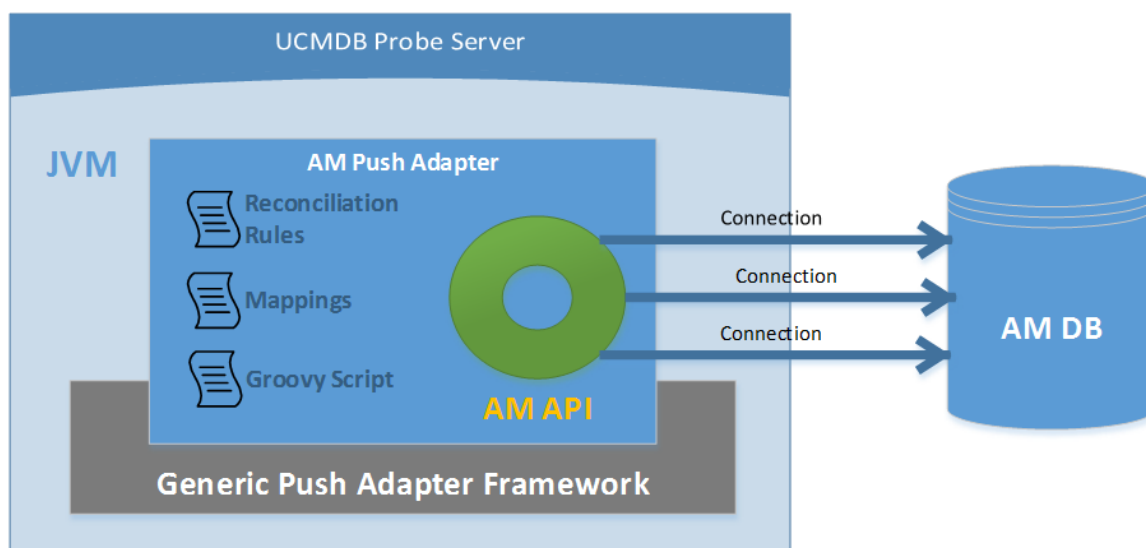
In the non-parallel push mode, the AM Push Adapter loads one AM native API library and the AM APIs use a single connection to the AM database to query, update, insert or delete data in the AM database. Because data is transferred in one single thread in sequence, it does not have concurrency issues but it takes significant amount of time to push a high volume of data.

Parameters

There is no specific parameter for non-parallel push mode.

Multi-threading Push Mode

Architecture



In the multi-threading push mode, the AM Push Adapter loads one AM API library, which creates multiple connections to the AM database to transfer data in parallel. This mode leverages the AM API multi-threading capability to increase the throughput of data push. Due to known issues and limitations of the AM API multi-threading, using this mode might produce concurrency errors and high failure rate.

Parameters

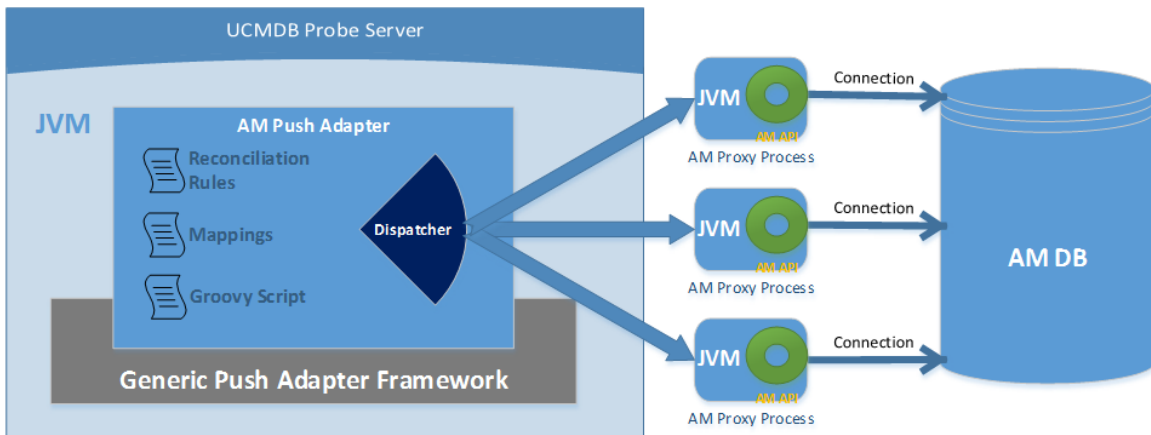
Setting	Default Value	Description
parallel.thread.pool.size	8	The option sets the number of threads in probe to be used in multi-threading parallel push.

Caution: This mode is not recommended in production.

In this mode, the AM API connections are handled in the connection pool by the Asset Manager API library. Additional parameters can be used. See the Asset Manager Web documentation for more information.

Multi-processing Push Mode

Architecture



In multi-processing push mode, the AM Push Adapter creates a number of AM proxy processes and evenly dispatches a subset of data to each process. Each process has a dedicated AM native API library to communicate with the AM database. In contrast to the multi-threading mode, an AM proxy process establishes a single connection to the AM database.

AM proxy process is a child process of AM Push Adapter and it is running in a dedicated JVM instance. Because every AM proxy process loads its own dedicated AM API library, the memory consumption increases as more processes are added to improve data push throughput.

The approach of using separate AM proxy processes avoids the scalability limit caused by AM API lock bottle-neck. Therefore, it improves the push performance more significantly than the multi-threading mode. It also prevents most concurrency errors caused by the AM API multi-threading limitations and known issues. The throughput of a push job running in multi-processing mode is decided by the network latency and the hardware configuration including UCMDB server, probe server and AM database server.

Parameters

The following table shows the specific settings for multi-processing mode.

Setting	Default Value	Description
parallel.thread.pool.size	8	The option sets the number of AM proxy processes to be used in parallel push.
am.api.proxy.client.retry.count	30	The option sets the maximum number of times the Push Adapter tries to connect to an AM proxy process.
am.api.proxy.jvm.options	-Xms16m;-Xmx56m;-XX:MaxNewSize=16m;-XX:MaxPermSize=16m	The JVM option is for AM proxy process. Multiple options are separated by ‘;’.
am.api.proxy.port.range	50000;51000	The option specifies the range of the network socket port that can be used by AM proxy processes.
am.api.proxy.idle.seconds	300	The option sets how long an AM proxy process waits for requests from the AM Push Adapter. If no requests from AM Push Adapter, AM proxy process will exit after waiting the idle seconds

Tuning

The performance of the multi-processing data push is based on the environment where it runs. In particular, it is affected by the following items.

- Hardware configuration of the Probe server including the number of CPUs and RAM.
- Hardware configuration of the AM database server.
- Network latency between the Probe server and the AM database server (critical).
- Network latency between the UCMDB server and the Probe server.
- Hardware configuration of the UCMDB database server.

You can increase the value of the parallel.thread.pool.size parameter to improve the throughput with higher hardware configuration.

In a production environment, the Probe server and the AM database server should be placed on the same LAN with less than 1ms latency.

The probe server and the UCMDB server can be set farther away from each other (150 ms network latency has been tested).

The recommended practices are as follows.

- The memory consumption of one AM proxy process is around 200 MB. It should not exceed 500 MB. Choose a thread size and avoid use up all memory of the current server. The default thread size is 8, so AM proxy processes will take up to 4 GB memory in the multi-processing mode.
- CPU core number is important. If the memory is enough and you do not need to consider other programs, keep the “parallel.thread.pool.size” value equal to the CPU core number to get the best performance.
- Multi-processing push creates more sessions on the AM database server than other push modes. A single AM proxy process creates two database sessions. One is for data push, the other is used by the groovy script. The AM Push Adapter instance in the Probe server also needs 2 dedicated database sessions to handle shared resources such as dynamic mappings. In order for multi-processing push to run successfully, you need to ensure that sufficient database sessions on the AM database are available for push jobs. Below is the formula to calculate the number of database sessions needed for a push job.

of processes x 2 + 2

18 sessions on the AM database server are required for a multi-processing push job with the default setting of 8 processes.

Troubleshooting

- **Multi-processing mode adapter log**

Log information is printed when the probe log level is set to INFO. For information about how to set the probe log level, refer to the UCMDB documentation.

The following is sample log information of the AM Adapter when multi-processing is enabled. The thread size and process size are both 8. The thread pool timeout time is 800 seconds. In the sample log, 57 CIs are assigned to a worker thread. An AM proxy process is created on network port 50009. After AM starts to process data, the worker thread connects to the AM proxy process to execute AM AQLs. The worker thread closes the connection when it completes the task.

```

INFO [AdHoc:AD_HOC_TASK_PATTERN_ID-647-1452758725341] - push4 >> [AM Adapter] Starting Push to AM
INFO [AdHoc:AD_HOC_TASK_PATTERN_ID-647-1452758725341] - push4 >> [AM Adapter] RTN List: 451
INFO [AdHoc:AD_HOC_TASK_PATTERN_ID-647-1452758725341] - push4 >> [AM Adapter] Wrapped Root List: 451, Operation Name: ADD
INFO [AdHoc:AD_HOC_TASK_PATTERN_ID-647-1452758725341] - push4 >> [AM Adapter] Parallel is enabled with thread pool size [8] !
INFO [AdHoc:AD_HOC_TASK_PATTERN_ID-647-1452758725341] - push4 >> [AM Adapter] Thread pool wait time [800s] !
INFO [AdHoc:AD_HOC_TASK_PATTERN_ID-647-1452758725341] - push4 >> [AM Adapter] Starting to generate Dependency Node trees.
INFO [AdHoc:AD_HOC_TASK_PATTERN_ID-647-1452758725341] - push4 >> [AM Adapter] Starting In bulk merge
INFO [AdHoc:AD_HOC_TASK_PATTERN_ID-647-1452758725341] - push4 >> [AM Adapter] Start a worker to process 57 RTN nodes
INFO [AdHoc:AD_HOC_TASK_PATTERN_ID-645-1452758715498] - push4 >> [AM Adapter] 10200@adaptervm26 RMI is initialized
INFO [AdHoc:AD_HOC_TASK_PATTERN_ID-645-1452758715498] - push4 >> [AM Adapter] Created new process: port@50009
INFO [AdHoc:AD_HOC_TASK_PATTERN_ID-645-1452758715498] - push4 >> [AM Adapter] New AMApiRmiClient...
INFO [AdHoc:AD_HOC_TASK_PATTERN_ID-645-1452758715498] - push4 >> [AM Adapter] Starting to generate Dependency Node trees.
INFO [AdHoc:AD_HOC_TASK_PATTERN_ID-647-1452758725341] - push4 >> [AM Adapter] Try to connect to new sub-process time 1, @port: 50010
INFO [AdHoc:AD_HOC_TASK_PATTERN_ID-645-1452758715498] - push4 >> [AM Adapter] Starting In bulk merge
INFO [AMPushThreadPool-2-6] - push4 >> [AM Adapter] Created new socket: 56315 -> 50009
INFO [AdHoc:AD_HOC_TASK_PATTERN_ID-645-1452758715498] - push4 >> [AM Adapter] Start a worker to process 47 RTN nodes
INFO [AdHoc:AD_HOC_TASK_PATTERN_ID-647-1452758725341] - push4 >> [AM Adapter] initializeRmiApi, socket: 56314 -> 50010
INFO [AMPushThreadPool-1-6] - push4 >> [AM Adapter] [Thread AMPushThreadPool-1-6] Finished pushing CIs to AM in 30500 ms
INFO [AMPushThreadPool-1-6] - push4 >> [AM Adapter] Close client connection...

```

• AM proxy process log

You can enable the AM proxy process log to generate information about what an AM proxy process is doing. The logging configuration file (log4j.properties) is located in Adapter Management > AMGenericAdapter > Configuration Files > AMGenericAdapter/config/.

The following is the default content of this file, which follows log4j rules. For more information about log4j rules, refer to the log4j documentation. By default, log files are generated in the <UCMDB Installation Path>\DataFlowProbe\runtime\log\amPushLog directory.

```

log4j.rootLogger = ERROR, fileout
log4j.appender.stdout = org.apache.log4j.ConsoleAppender
log4j.appender.stdout.layout = org.apache.log4j.PatternLayout
log4j.appender.stdout.layout.ConversionPattern = %d{ABSOLUTE} %5p - %c - %m%n
log4j.appender.fileout = org.apache.log4j.RollingFileAppender
log4j.appender.fileout.File = ${amPushLogPath}/ampush_${am.api.proxy.port}.log
log4j.appender.fileout.MaxFileSize = 50MB
log4j.appender.fileout.MaxBackupIndex = 100
log4j.appender.fileout.layout = org.apache.log4j.PatternLayout
log4j.appender.fileout.layout.ConversionPattern = %d %-5p - %c - %m%n

```

- When running a multi-processing push, each thread is given a unique name to distinguish them from each other. In the AM Adapter log, messages generated in a thread are prefixed with the thread name. Each thread name uses the following syntax:

AMPushThreadPool-<thread pool counter>-<thread counter>. The following is an example:

```
AMPushThreadPool-1-5
```

- If there are errors when running in multi-processing mode, you can check to see if the actual number of AM proxy processes matches the value of the parallel.thread.pool.size parameter. If they do not match, further investigation is required to determine whether this is a problem. To find out the number of AM proxy processes, you can open Windows Task Manager and look for processes named java.exe and check their file path in the Command Line column.
- Monitor the CPU usage of AM proxy processes (named java.exe). If some of them are always idle,

there may be configuration errors in the Push Adapter or the Probe server. In this case, investigation is required to identify the root cause.

- In multi-processing mode, we recommend that you do not enable the AM adblog because the adblog is modified by different processes and may cause unexpected errors.
- The `am.api.proxy.idle.seconds` option cannot be set to 0. By doing so you make the AM proxy process wait forever without a timeout, even if there are no AQLs to be executed.

Differences between Push Adapter and Generic Adapter

This section describes the differences between Push Adapter and Generic Adapter. For more information about how to migrate from the Push Adapter to the Generic Adapter, see the *Migration from AM Push Adapter to AM Generic Adapter* guide.

Differences between Old Push Adapter and Push in Generic Adapter

The Generic Adapter framework is an extension to the Enhanced Generic Push Adapter framework to support bi-direction data transformation between UCMDB and external systems. The AM Generic Adapter is built on top of the Generic Adapter framework, thus it supports most of the features of the old AM Push Adapter.

To migrate from the old AM Push Adapter to the new AM Generic Adapter, you need to be aware of the following changes.

Mapping Schema Changes

Data Push uses the existing Enhanced Generic Push Adapter framework with minor XML schema changes. Please see the *Achieving Data Push using the Generic Adapter* chapter in *Universal CMDB Developer Reference Guide*.

File Renaming and reorganizing

- TQL names
- Mapping file names
- am-push-mapping.xml is renamed to am-push-config.file
- AMPushFunctions.groovy is renamed to AMPush.groovy

- AMReconciliationAdvanced.groovy is renamed to AMReconcil.groovy
- Some of the functions in AMPushFunctions.groovy are moved to AMUtils.groovy

Asset Manager Entity Reference in Mapping Script and Reconciliation

It is a big change for involving AM entity in both push mappings and reconciliation files.

In old push mapping, name of target_ci_type is specified ci-type of am-mapping in reconciliation file.

In new push mapping, name of target_ci_type is AM entity name or sub link name of AM entity. This change is intended to add attributes into mapping file use Mapping UI.

Type of target_ci_type in mapping file is the new attribute to be used in reconciliation file.

For more information, see ["Define Reconciliation Rule for Target AM Entity in Push" on page 55](#).

Differences between Old Population Adapter and Population in AM Generic Adapter

The old population adapter is implemented based on Generic Database Adapter, it is quite different and incompatible with the AM Generic Adapter.

The following sections describe some of the major differences between them.

No need to import views for AM database

New AM Generic Adapter retrieves data by AM API, not like old population adapter that connects to database directly. Therefore, the Generic Adapter does not need to import views any more, it communicates with AM database through AM Entity.

Population mapping needs to specify TQL

The old populate adapter mapping uses ORM.xml, it does not rely on any dedicated TQL. Instead, it retrieves data from AM database via OR mapping, according to each CI and relation.

The new population retrieves data from AM Entity through population mapping, and then synchronizes data to UCMDB by specifying TQL.

AM Entity -> Population Mapping -> TQL

ORM mapping is separated to several population mappings by CI and relation:

```
population
  AM BusinessElement Population.xml
  AM BusinessElement Relations Population.xml
  AM Interface Population.xml
  AM Location Population.xml
  AM Node Population.xml
  AM Printer Population.xml
```

Attributes conversion and discrimination

The AM Generic Adapter supports complex groovy syntax to implement attributes conversion and discrimination logic. It does not need to configure lots of properties and xml files. Two examples are shown as follows.

discriminator.properties

Its contents can be replaced by AMPopulate.groovy, see hostDataCilist.

```
// Host data CI
hostDataCilist.add(["host_node", "Computer,Desktop computers,Computer servers,Laptop,Virtual Mac"]);
hostDataCilist.add(["nt", "Windows computer,Windows desktop computer,VMware VirtualCenter"]);
hostDataCilist.add(["unix", "Unix server computer,Unix desktop computer,Solaris Zone server"]);
hostDataCilist.add(["vmware_esx_server", "VMware ESX Server"]);
hostDataCilist.add(["mainframe", "Mainframe,Mainframe CPC"]);
hostDataCilist.add(["lpar", "Mainframe Logical Partition"]);

hostDataCilist.add(["firewall", "Firewall"]);
hostDataCilist.add(["router", "Router"]);
hostDataCilist.add(["switch", "Switch"]);
hostDataCilist.add(["atmswitch", "ATM switch"]);
hostDataCilist.add(["netprinter", "Network printer"]);
hostDataCilist.add(["netdevice", "ADSL Modem,Appletalk Gateway,Bandwidth Manager,Cable modem,CSU_"]);
hostDataCilist.add(["storagearray", "Storage Array"]);
```

node_role.properties

Its contents can be replaced by AMPopulate.groovy, see nodeRoles.

```
// Node roles
nodeRoles.put("Computer servers", "server");
nodeRoles.put("Unix server computer", "server");
nodeRoles.put("Windows computer", "server");
nodeRoles.put("VMware ESX Server", "server");
nodeRoles.put("Solaris Zone server", "server");

nodeRoles.put("Desktop computers", "desktop");
nodeRoles.put("Computer", "desktop");
nodeRoles.put("Laptop", "desktop");
nodeRoles.put("Windows desktop computer", "desktop");
nodeRoles.put("Unix desktop computer", "desktop");
```

Frequently Asked Questions

What is a Root CI node?

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

How do I get the version information of Asset Manager adapter?

In UCMDB, select **Package Manager** on the Administration menu, click **Readme** of the AM generic adapter or AM push adapter. In the text file, you can find the version and package ID of the AM adapter.

When is a new Asset created in Asset Manager?

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

- **Nodes**
- **Business Elements**
- **Printers**
- **Display Monitors**

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

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

See "[Action on Delete](#)" on page 70.

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

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

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

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

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

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

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

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

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

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

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

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

- SW_amSoftInstall
- Complete_amSoftInstall_User

- soft_Hyper_amSoftInstall
- SW_amSoftInstall_User

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

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

Example:

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

See ["Attribute Reconciliation" on page 69](#).

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

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

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

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

Should I select the 'Enable Parallel' Feature?

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

Asset Manager configured over an SQL Server database, needs some tuning before enabling this capability. See ["Prepare Asset Manager for Parallel Push" on page 14](#).

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

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

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

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

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

What should I do before running Cluster mapping job?

You can use cluster with Asset Manager 9.50 and later versions. Or, if you want to use cluster with Asset Manager versions before 9.50, you must install the latest version of the SLO Best Practice package.

Otherwise, you should follow these steps to manually add the **ITCLUSTER** nature in Asset Manager before running cluster mapping jobs.

1. Log on to Asset Manager as an administrator.
2. Go to **Portfolio management > Asset configurations > Natures**.
3. Check if there is a nature with Code of value "ITCLUSTER".
4. If it does not exist, click the **New** button to create it with the following values.

```
Name="Cluster"
Code="ITCLUSTER"
Create=Portfolio item
Also create = Cluster(amCluster)
Management constraint = Unique asset tag
Has software installed = checked
Business service =checked
```

5. Click the **Create** button to save the new nature record.

Troubleshooting and Limitations

- **Limitation:** A single probe may only connect to one version of Asset Manager. (Use of multiple instances of the same version is supported). This is due to the JVM limitation of loading only one Asset Manager API per process.
- **Limitation:** The Data Flow Probe must be installed on a Windows OS.
- **Limitation:** DB2 parallel push mode is not supported in UCMDDB 10.01.
- **Problem: Missing DDLs or jars.**

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

Asset Manager DLLs and/or Jars are missing

Solution: See ["Deploy Asset Manager Zip Package" on page 16](#).

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

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

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

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

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

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

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

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

The Relations flow (TQL query) assumes that you schedule (either in the same job, or in a different job) the different flows that this Relations push depends on, to run before this flow. For more information, see ["Asset Manager Push Jobs" on page 26](#).

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

The software flows assume that you schedule the computer push flow (either in the same job, or in a different job) to run first. See ["Frequently Asked Questions" on page 117](#).

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

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

See ["Frequently Asked Questions" on page 117](#).

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

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

Solution: Validate the following:

1. The database client is installed on the Data Flow Probe machine. See ["Install a Database Client" on page 21](#).
2. The installed client is a 64 bit version.
3. The client is installed on the actual Probe selected in the integration point configuration.

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

Solution 1: This can happen when the maximum length of an attribute is too short compared to the attribute's value in UCMDB. It causes the attribute value to truncate when pushed to Asset Manager. However, on a different execution, when attempting to reconcile the attribute, there will not be a match because of the truncated value in asset Manager. Therefore, increase the attribute length. See ["Update Asset Manager Schema" on page 13](#).

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

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

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

Solution 1: Download and install the Microsoft Visual C++ 2008 SP1 Redistributable Package (x64). You may download this from:

<http://www.microsoft.com/download/en/details.aspx?id=2092>.

Thereafter, reboot Windows and restart the Probe.

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

- **Problem: The number of the virtual CPUs is pushed into AM**

Some customers may tailor UD, add virtual CPUs as physical CPU CI, and then set the **Is Virtual** attribute to **True**. In this situation, the push adapter will count the number of both virtual and physical CPUs to set the **fCPUNumber** field in AM.

Solution 1: To avoid counting virtual CPU in AM, add the **Is Virtual = False** condition in AM Computer/Node Push TQL.

Solution 2: Create a groovy function (for example, `getPhysicalCoreCount`) to count the CPU CIs whose **Is Virtual** is **False**.

Logs

- **Log files**

The push adapter framework uses logs different from 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`

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\`

- **Unique identifier**

When multiple-processing is enabled, there is an identifier of push adapter threads in log. The syntax is as follows

`AMPushThreadPool-<id of data chunk>-<id of thread to handle a subset of a chunk>`

For example, `AMPushThreadPool-2-1`, `AMPushThreadPool-3-7`, `AMPushThreadPool-1-1`, each thread has its unique name.

AM API Logs

The AM Generic Adapter uses the AM native APIs to communicate with the AM database. You can turn on the AM API log to trace the calls invoked to the AM native APIs and SQL queries issued to the AM database.

The following steps are used to configure the AM API log for the AM Generic Adapter.

1. On the computer that hosts the Data Probe server, find the Windows account to run the uCMDB Probe service.
2. If the account is Local System, perform the next actions in the directory "C:\Windows\System32\config\systemprofile\AppData\Roaming\Micro Focus\AssetManager\conf". Otherwise, do it in the directory "<user profile dir>\AppData\Roaming\Micro Focus\AssetManager\conf", where <user profile dir> is the profile directory of the user who runs the uCMDB Probe service.
3. In the directory, locate the aamapixx.ini file where 'xx' represents the AM version, e.g. aamapi94.ini for AM 9.4x. Create the file if it does not exist.

4. Add the following lines to the file.

```
[Option]
/AdbLog/LogApiCalls=1
/AdbLog/AdbLogStartup=1
/AdbLog/AdbLogFileName=c:\temp\am-generic-adapter.log
/AdbLog/AdbLogFileSize=100000000
/Advanced/TraceLDAP=1
```

5. Restart the uCMDB Probe server.
6. After running an integration job, the file is created in the location you specify in the AdbLogFileName property.