Guidelines for Deploying HPOM Dependency Mapping Automation (DMA)

Table of Contents

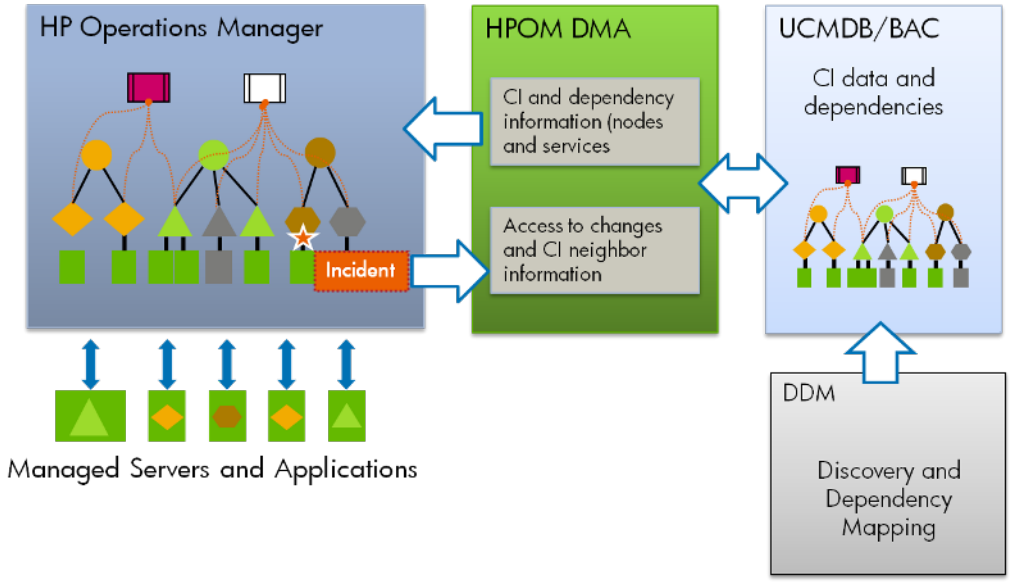
Table of Figures

Introduction

This technical white paper is intended primarily for Service Designers and Integrators who wish to gain additional insight into the process of deploying HP Operations Manager Dependency Mapping Automation (HPOM DMA) software, especially where customization will be required to achieve the desired results. The material covered here assumes prior knowledge of HP Operations Manager (HPOM) and HP Universal CMDB (UCMDB). Please refer to "For More Information" at the end of this paper for a link to the public HP web site where HPOM DMA documentation can be obtained.

DMA delivers flexible integration capabilities which enable the customer to leverage the typed model of discovered Configuration Items (CIs) and relationships (topology) in the UCMDB for use with HPOM. Its core function is to automate and simplify the process of creating nodes and service views in HPOM. Additionally, DMA enables operational drill-down from HPOM managed nodes and services into the change history and neighbor CIs maintained in the UCMDB. For customers with multiple HPOM servers, DMA serves as the mechanism to maintain consistent and shared service views across all of them, with the UCMDB as the single source of CIs and relationships. And finally, DMA can leverage the discovery part of UCMDB (Discovery and Dependency Mapping, or DDM) to "inform" HPOM of new nodes in the infrastructure that need to be put under proper management in order to support important business services.

Figure 1. HPOM Dependency Mapping Automation



It's important to understand up front that DMA is an integration product and toolkit. It leverages, and significantly enhances, an environment that already has:

- A UCMDB that has been populated with CIs and relationships, and
- One or more HPOM instances (either HPOM on UNIX or HPOM on Windows)

Deploying DMA requires knowledge of *both* UCMDB and HPOM. Although DMA includes out-of-box functionality for customers using the Database and OS Smart Plug-Ins (SPIs), it is likely that a customer

will need additional customization to get maximum benefits for their specific operational environment. In some cases this may simply involve modifying or leveraging the existing out-of-box sync packages and TQLs provided with DMA, and in other cases a completely new sync package and associated TQLs will be required to produce the desired results. So it will be important for Service Designers and Integrators to fully understand the process and boundaries of implementing DMA as well as the technical details.

The first part of this white paper will focus on an overall understanding how to approach modeling with DMA and some of the key things to know about using the UCMDB with DMA and HPOM. This will include topics such as understanding the differences between how UCMDB relates CIs and how HPOM represents and instantiates services, important things to know about designing TQLs to be used with DMA, and the concept of DMA "ownership" of nodes and services in HPOM.

The second part of the paper will cover some best practices and "how to" information, and will generally be more technically detailed. This part will be subdivided into three sections: a) node synchronization, b) service synchronization, and c) using the Smart Message Mapper (SMM).

Overall, our approach in this white paper will be to augment and emphasize certain relevant topics which (in most cases) are described in the existing HPOM DMA documentation. This paper is NOT a substitute for the product documentation; in fact readers who have already completed a review of the **DMA Installation and User's Guide** will get the most from this material. Some of the material (especially in Part II) relates especially to extending DMA for custom application environments, and so prior review of the **DMA Extensibility Guide** is also recommended.

Part I: Understanding the DMA modeling process

At the simplest level, there are three essential steps in the process of achieving the desired modeling results with DMA:

1. *Understand and define the target results in HPOM*

   You must visualize and plan for the nodes, node groups, services, and service hierarchy that you want to create in HPOM. If this is not done up front as the first step in DMA implementation, you will not likely achieve the desired results.

2. *Produce the required source data from the UCMDB*

   Ultimately a TQL query will be the mechanism that provides the data for DMA, but additional UCMDB configuration may be required to facilitate design and implementation of the *right* TQL. And of course it's important that the UCMDB actually contains the desired CIs to start with.

3. *Create or configure the required transformation artifacts for DMA synchronization*

   Synchronization is driven by one or more synchronization packages containing mapping rules and optionally scripts. Creation of other elements such as Service Type Definitions (STDs) and node groups in HPOM may also be required . (More information regarding sync packages and STDs will be found later in the document.)

In the remainder of Part I, we will focus primarily on the UCMDB-related aspects of the DMA Modeling process (step 2 above). For most customers and Integrators mainly familiar with HPOM, this is the area of least knowledge and experience.

The role of the UCMDB

The UCMDB is the source of "truth" regarding CIs and relationships in the infrastructure. The CIs and topology maintained in the UCDMB are typically instantiated through DDM discovery patterns that can periodically run at intervals as appropriate. The UCMDB is therefore also a *dynamic* source of truth; its content represents what is actually in the environment based on the currency of the discovery jobs. The dynamic nature of the UCMDB makes it especially useful as THE source of information for automating the management of nodes and service views in HPOM.

UCMDB as DMA data source

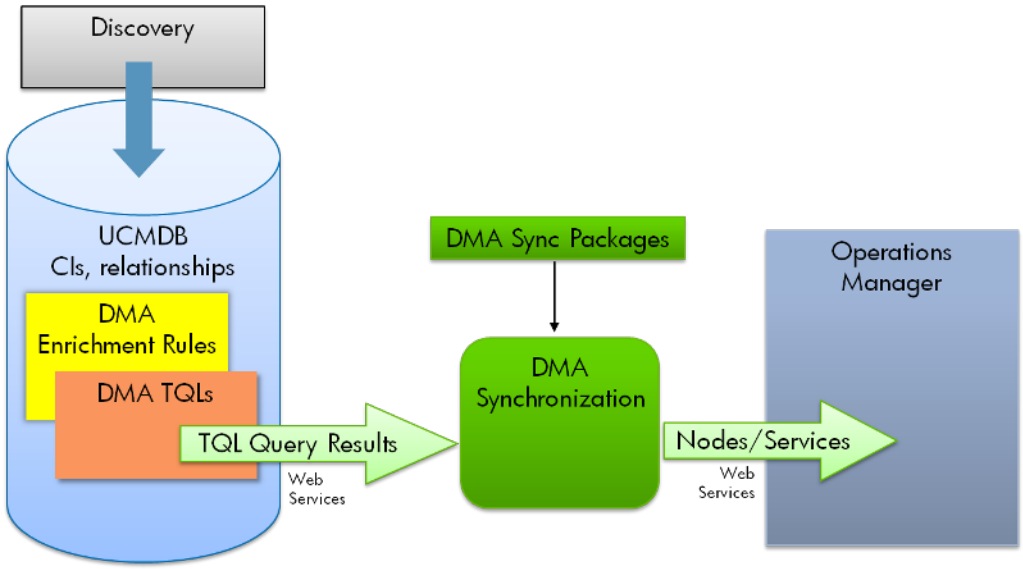
DMA is a *consumer* of information in the UCMDB; it does not populate the UCMDB. There are two exceptions: When DMA's UCMDB packages are deployed, a new CI Type (Service Group), a new relationship type (Service Group Contained) and new valid link settings are created to facilitate representing virtual services in HPOM. In addition, UCMDB enrichment rules (also installed as part of the DMA UCMDB packages or configured as part of DMA customization) may instantiate new Service Group CIs, new CI relationships and update certain CI attributes. These UCMDB enrichments facilitate the creation of specific Topology Query Language (TQL) queries whose results are the "input" to DMA synchronization. But it's important to understand that DMA is *not* a discovery source for the UCMDB.

Synchronization is the core functionality in DMA. It is a "run time" process which can be executed whenever desired or scheduled to run at desired times. While the synchronization process is technically run within and controlled by DMA, it is just one part of a bigger modeling exercise that also includes UCMDB (as the source of CIs and relationships) and HPOM (as the target for nodes and service hierarchies).

For the OS and Database SPIs, DMA includes all the necessary elements and configuration needed in UCMDB and HPOM to support synchronization "out-of-the-box." However an implementation of DMA for custom applications may require new UCMDB enrichment rules (to add CI instances and relationships) and will require one or more custom designed TQLs to produce the desired CI result set as input for synchronization. The key point here is that getting the desired end results in HPOM, especially for custom application environments, involves more than just installing and configuring DMA. The UCMDB is the starting point for the whole process and having the necessary discovered CIs, Service Group CIs, relationships, valid links, and TQLs in the UCMDB is an essential part of the successful design and implementation of a DMA-based solution.

Figure 2 highlights the essential UCMDB components involved in the synchronization process.

Figure 2. UCMDB is the source of data for DMA Synchronization



Guidelines for successful DMA synchronization with UCMDB data

If it's not in the UCMDB, it can't be synchronized

It should be obvious, but it's worth stating: If the CIs you want in HPOM don't exist in the UCMDB, then DMA will be of little help. This is why it is important to start the DMA modeling process by fully understanding what is in the UCMDB. It's a good idea to explore the UCMDB using the View Manager or Universe Manager to understand what is there and to determine if specific DMA customizations or UCMDB enrichment will be needed in order to get the desired results with DMA.

If it's not returned by the TQL, it can't be synchronized

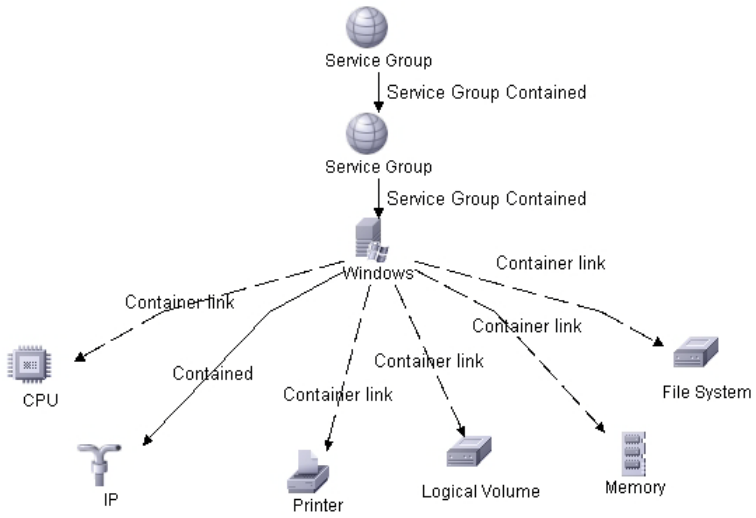
The first step in the DMA synchronization process is execution of one or more TQL queries as defined in the active sync packages. Understanding how to create appropriate TQLs for use with DMA is vitally important. The TQL must be crafted so that it returns the set of CIs that are needed as input to DMA synchronization. This may take some experimentation and tweaking to get the desired results. If you don't see something in HPOM that you expected DMA to create, the first thing to do is verify that the TQL is actually providing the required input to synchronization. Use the View Manager in UCMDB to get counts and preview TQL results.

Use cardinality and filtering in the TQL to constrain the input to synchronization

The TQL must include the appropriate CI Type nodes and topology, but it must also include the appropriate node filters and cardinality rules to filter out what is NOT needed or desired.

Consider the following out-of-box TQL provided by DMA for use with systems running the Microsoft Windows Operations System :

Figure 3. DMA TQL "Windows Operating System (Operations)"



At a minimum, you would want this TQL to return the top three CIs (both of the Service Group CIs and the Windows node itself). Depending on how discovery (DDM) is configured in the environment, you may or may not have the CPU, IP, Printer, Memory, etc. CIs in the UCMDB and related to the Windows host. The default cardinality rules for the Windows node in this TQL are:

Service Group Contained (Service Group, Windows) : 1..*
AND Container link (Windows, File System) : 0..*
AND Container link (Windows, Logical Volume) : 0..*
AND Container link (Windows, Memory) : 0..*
AND Container link (Windows, Printer) : 0..*
AND Container link (Windows, CPU) : 0..*
AND Contained (Windows, IP) : 0..*

These rules allow the TQL to return just the Service Group CIs and the Windows node without anything else. Of course, if there are instances of the CPU, IP, or Printer etc. connected with the indicated relationships to a Windows node these will be returned as well.

If you wanted to further constrain the TQL result so that only Windows nodes with *at least one* container link to a File System exists, you would modify the Windows File System cardinality rule as follows:

AND Container link (Windows, File System) : 1..*

Another way to constrain the TQL results is to put filters on specific CI Type nodes in the TQL. Let's say that you want to restrict the above TQL to return only those nodes in the UCMDB topology that belong to a specific domain. This can be accomplished by configuring the Node Properties for the Windows node in the TQL.

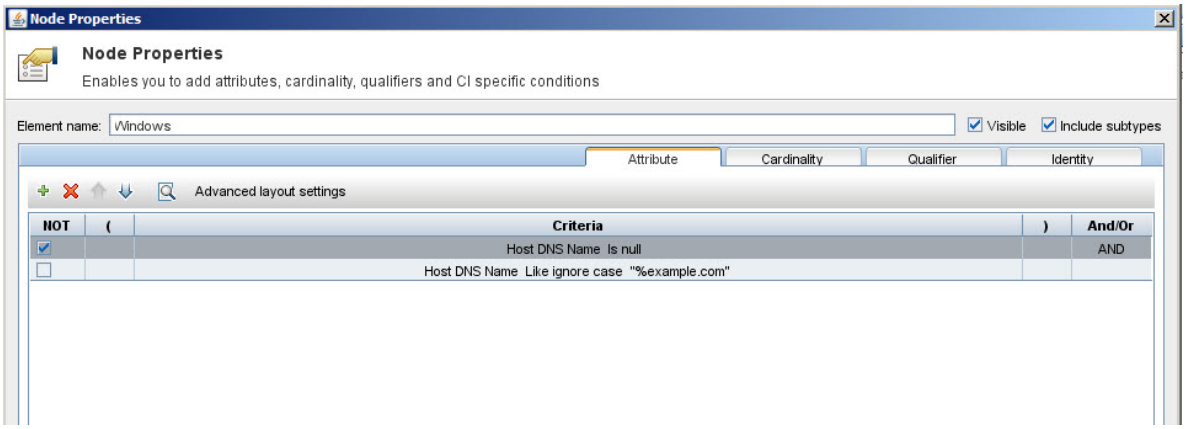
(Note: The TQLs provided out-of-box with DMA have the following default filter on host nodes:

Created By Equal "ProbeGW_Topology_Task_"

This means that the TQL will only return nodes which are running DDM probes. Since this is not what you would typically want, you should always plan on modifying the out-of-box TQLs when deploying DMA).

To have the above TQL return only nodes in the domain "example.com," you would configure a filter rule on the Windows node as shown in Figure 4.

Figure 4. Filtering nodes in the TQL



Note: The first filter rule in the figure above is one way to ensure that only hosts with a valid host name are returned as input to DMA synchronization, however this assumes that in fact the Host DNS Name attribute is being initialized by DDM. It should be noted that as of UCMDB/BAC 8.0, the Host DNS Name attribute has been deprecated. Host name information can actually appear in different attributes depending on the DDM implementation. This is why DMA 8.20 includes a new "DNS Mapping" sync package which checks the different attributes and does DNS lookups if needed to determine the host name. The package contains only a Pre-Mapping groovy script which does the work. You can examine the script directly in the DMA user interface by navigating to the "Extending DMA → Synchronization Packages" page and then expanding the link for the "DNS Mapping" sync package.

The importance of CI attributes

Each CI in the UCMDB has a set of properties known as "attributes." Depending on the CI Type, DMA requires certain attributes to be part of the CI data stream returned by the TQLs. Other attributes are recommended and if present can simplify the mapping rules and configuration steps that may be required once the CIs are inserted into HPOM (e.g. node properties such as "Host Operating System Version" and "Host Model.")

Figure 5 provides a summary of the required CI attributes for proper operation of DMA.

Figure 5. Required CI attributes for DMA

CI Type	Attribute Name	Required for	Sync Package
all	display_label	Setting caption on HPOM service (displayed by Service Navigator)	Default
"database"	database_dbtype	mapping rules – determining DB vendor	Informix
all node types	host_servertype	mapping rules – determining database running on a node	Informix, Oracle, MSSQLServer, Sybase
"sqlserver"	database_dbversion	mapping rules – determining version of SQL Server	MSSQLServer
all node types	host_os	mapping rules – OS type detection	Unix OS, Windows OS

Of course if you are not planning to use a specific sync package you would not need to worry about having the associated required attribute initialized. More information about required and recommended attributes can be found in the **DMA Extensibility Guide** and Appendix A of the **DMA Installation and User's Guide** (sections "HPOM DMA Default Synchronization Package" and "DNS Mapping Synchronization Package: dmaDnsMapping").

Beyond just making sure you have the required attributes, using them effectively can contribute a lot to a successful DMA implementation. CI attributes are used in several important ways:

- CI attributes can be mapped to create equivalent or derived service attributes in HPOM during the synchronization process (this is especially important for the Smart Message Mapper – more on this later)
- CI attributes can be leveraged in the *logic* of mapping rules, e.g. an enrichment rule could set a certain value in an attribute that is used by a mapping rule to determine how to assign a CI to a node group

Exposing CI attributes

It's important to verify that the CI attributes you need for DMA synchronization are actually exposed to the UCMDB web services interface when the TQL/View results are returned by UCMDB. You cannot assume that just because an attribute exists that it will be available to DMA. To ensure the desired CI attributes are being properly exposed, follow these steps when editing the TQL in View Manager:

1. Right click on a node in the TQL
2. Select "Node Properties"
3. Click on the "Advanced Layout Settings" link in the toolbar area of the Node Properties window
4. Check the box for each attribute which needs to be exposed for DMA
5. Remember to save your TQL when all changes and configuration are completed!

If you don't see an attribute in the list that you know exists (e.g. "Host DNS Name"), you will need to first modify the UCMDB Type Model to make the attribute "visible." This can be accomplished with the UCMDB CI Type Manager.

In DMA versions prior to 8.20, if the Host DNS Name attribute was missing or did not have a value, then it was not possible to get the host configured properly in HPOM. As indicated above, starting with DMA version 8.20, a "Pre-Mapping" script is included in the new "DNS Mapping" sync package which makes it much more likely that the host node will be configured properly in HPOM, even if the "Host DNS Name" attribute is not available. But it is still recommended that the "Host DNS Name" attribute be initialized if possible, made visible and exposed in the TQL. In addition, the new DNS Mapping sync package should always be marked active for synchronization.

The TQL is where it all starts for DMA synchronization

In summary, TQLs provide three essential things that are needed for successful DMA synchronization:

- A set of CIs (which represent the desired HPOM nodes and services)
- CI attributes (which are mapped to HPOM service attributes and used in mapping rules)
- Structure (the hierarchy of the desired HPOM services and relationships)

The Service Designer should always be asking the following questions during the TQL design process:

  - Does the TQL return the CIs I want in HPOM?
  - Are the required CI attributes "visible" (in the CI Type Model) and "exposed" to the UCMDB web services interface?
  - Are the required CI attributes initialized with appropriate values (either via DDM or through UCMDB enrichment rules)?
  - Do I need to create and initialize custom CI attributes specifically for use during DMA synchronization (e.g. in mapping rules)?
  - Does the TQL return a topology which matches the desired HPOM service view structure?

As the above discussion illustrates, getting desired results with DMA starts with having the right TQLs in place and verified. Furthermore, simple changes in the TQL configuration can have dramatic results on what is returned by the TQL. It's therefore essential that the DMA/HPOM Service Designer be completely familiar with creating and modifying TQLs and using the available filtering capabilities.

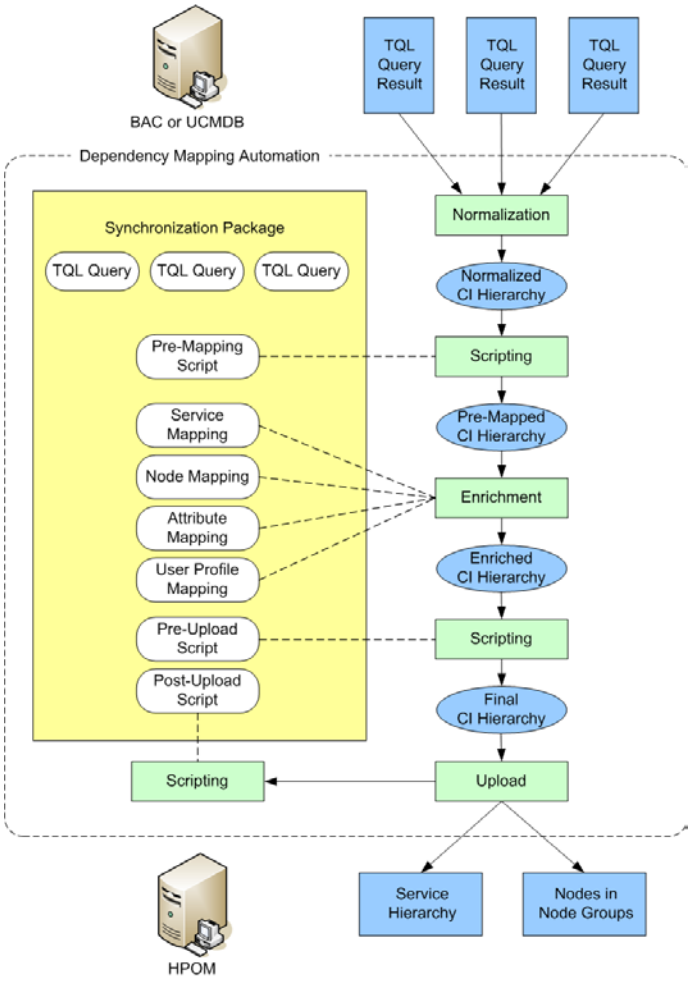
Transforming CIs (and topology) to nodes and services

Sync packages

As already stated, sync packages contain the mapping rules for transforming the stream of CI data provided by the TQLs into nodes and services that are instantiated in HPOM. The rules are contained in XML files and are activated and processed according to the settings you configure in the DMA user interface.

Figure 6 provides an overview of the sync package components and DMA processing steps.

Figure 6. DMA Synchronization Process



It is beyond the scope of this paper to cover all the details of creating sync packages, but for our discussion here it's important to understand the composition and key functions within a sync package. The components of the sync package are:

- Pre-Mapping, Pre-Upload, and Post-Upload scripts

  These can be used to do customizations of the CI hierarchy that are not possible with the mapping rules and to trigger certain actions before and after the CIs are uploaded into HPOM. An example of a Post-Upload action would be to initiate policy deployment to a node. (Note: The Pre-Mapping script is new in DMA version 8.20.)

- Service Mapping rules

  Maps UCMDB CIs to services in the HPOM service model by referencing Service Type Definitions (STDs).

- Node Mapping rules

  Maps host CIs to node groups

- Attribute Mapping rules

Sets node properties and service attributes in HPOM based on CI attributes. The attribute values can be modified and new attributes can be created in the mapping rules. This is where you determine things like the display label that will be used as the caption for services in the Service Navigator. The "default" sync package attributemapping.xml file is where node properties such as Host Operating System are initialized. The attribute mapping rules for services can have a lot of impact on how well the Smart Message Mapper works (more on this later).

– User Profile Mapping rules

Maps (assigns) a User Role or Profile to services in the HPOM service hierarchy

Note: As illustrated in the diagram above, DMA uses the term "Enrichment" to describe the process of modifying the stream of CIs and topology that is input from the TQLs. These enrichments (in the form of mapping rules) produce the ultimate nodes and services in HPOM. This should not be confused with enrichment rules in UCMDB which modify the CIs and topology model in UCMDB.

It's highly recommended that Service Designers and Integrators become familiar with the out-of-box sync packages before starting on the development of custom packages. Spend some time looking at the XML files with your favorite XML editor to learn the mapping syntax and how the rules work. More details on sync packages can be found in Chapters 7-12 of the **DMA Extensibility Guide**.

UCMDB and HPOM model differences

One of the key things for the Service Designer to grasp is that the UCMDB has a very robust type model for CIs. Each CI is of a certain type, e.g. "Host," "Network Interface," "Database," etc. The type model also includes a number of different relationships that may exist in the topology, e.g. "Container Link," "Depends," "Member," "Contains," "Contained," "Use," "Client Server," etc. And even further, the UCMDB type model defines *which* CI types can be related to other specific CI types using *which* relationships (these are known as "valid links"). Another important aspect of the type model is *inheritance*. For example, CI types "Websphere AS" and "Weblogic AS" are child types to the parent type "J2EE Server". Attributes defined at the parent level are inherited by the child types.

HPOM has a much simpler model. There are two service types (virtual and hosted_on) and two relationship types (Containment and Dependency). By default, DMA will create a containment relationship in HPOM if the UCMDB relationship label starts with the string "container" or "contained". (This behavior is controlled by the containmentrelations.xml configuration file in DMA.) And DMA does not recognize CI type inheritance; each CI type must be handled separately within the mapping rules.

It's important to carefully consider these model differences when you design the TQLs for DMA. The TQL should return a CI topology that is already constrained to what is possible in the HPOM service model. To do this you may need to add new valid links into the CI Type model (using the CI Type Manager) and create enrichment rules to add the new relationships where needed. You may also need to modify the containmentrelations.xml file to force additional mappings to the containment relationship in HPOM. Also give particular attention to setting the hosted_on service attribute for appropriate CIs in the attribute mapping file.

DMA control of synchronized CIs

Nodes

DMA puts all synchronized nodes into the "CMDB Nodes" node group which is initially created when DMA is installed. You also have the option to put specific node CIs into other node groups if desired, using node mapping rules in a sync package. It's no problem to have a node in multiple node

groups. Just be aware that the target node group must already exist in the HPOM configuration, or alternatively you could enable automatic node group creation (see below).

If during synchronization DMA determines that a previously synchronized node is no longer provided as input by any of the TQLs (which typically would mean it is no longer in the UCMDB), it is removed from the "CMDB Nodes" node group and placed in the "CMDB Removed Nodes" node group. This behavior is fixed and cannot be changed.

If nodes already exist in HPOM which are then imported for the first time using DMA, these nodes will automatically be removed from all user-assigned node groups and placed into the default DMA "CMDB Nodes" node group and any other node groups which are specified by the node mapping rules. And in similar fashion, if you make manual node group assignment changes to already imported nodes and then follow this with an import that no longer includes the node, it will be removed from all node groups and placed into the "CMDB Removed Nodes" node group.

> IMPORTANT: *You should consider DMA as having sole control over how synchronized nodes are configured into HPOM node groups.*

This is by design, since it allows for a fully automated node group assignment process. In environments which may contain many thousands of nodes, node group assignment may be extremely dynamic; automating the process therefore becomes very desirable. But this DMA feature should also be carefully understood before deploying DMA in a large production environment already managed by HPOM with existing node configurations. For example, existing Node Layout Group assignments in HPOM on UNIX can be effectively "undone" by DMA synchronization of nodes which already exist.

The bottom line with nodes is that node group configuration is either controlled by DMA or not. If you need to manually manage and configure the node group assignments, then these nodes must be excluded from DMA synchronization. For environments where more complex node configuration (e.g. layout groups) is needed for synchronized nodes, use the optional post-upload scripting capability provided by the sync packages.

Services

Services imported by DMA are placed under a single "CMDB" root service (service name is "ROOT_DMA_Service") in the HPOM model hierarchy. This keeps thing relatively simple and allows for a controlled migration from using SPI uploaded or manually created service trees to the services as discovered and instantiated in the UCMDB. It is no problem for these service trees to coexist at the same time. However, once the UDCMB branch has been well populated and matches the desired structure, you will need to remove the SPI services branch so that Smart Message Mapper will start to match incoming messages to the appropriate UCMDB services.

Like node group assignments, you should consider DMA as the "owner" of whatever is placed below the CMDB root service. The hierarchy as well as service configuration (including service attributes) is under complete control of DMA and is driven by both the TQL results feeding into synchronization as well as the mapping rules in the sync packages. As we will see later, there is quite a bit of complexity that can be included in the mapping rules if needed, including making changes to default settings for things like status and calculation rules. Each time synchronization is executed, the entire service tree under the CMDB root service will be re-created.

Part II: Getting what you want in HPOM

In this part of the white paper, we cover a few items which should be of interest to Integrators and customers wanting to implement DMA for use with custom application environments which may require more advanced techniques to get the desired results in HPOM.

Synchronizing nodes

How to put nodes into specific node groups

As mentioned in Part I, by default new nodes will be placed in the "CMDB Nodes" node group in HPOM. If you want certain nodes to be put into other node groups as well, you will need to control this through use of an appropriate sync package nodemapping.xml file. But before we dive into the mapping file details, let's review the motivations for putting nodes into specific node groups.

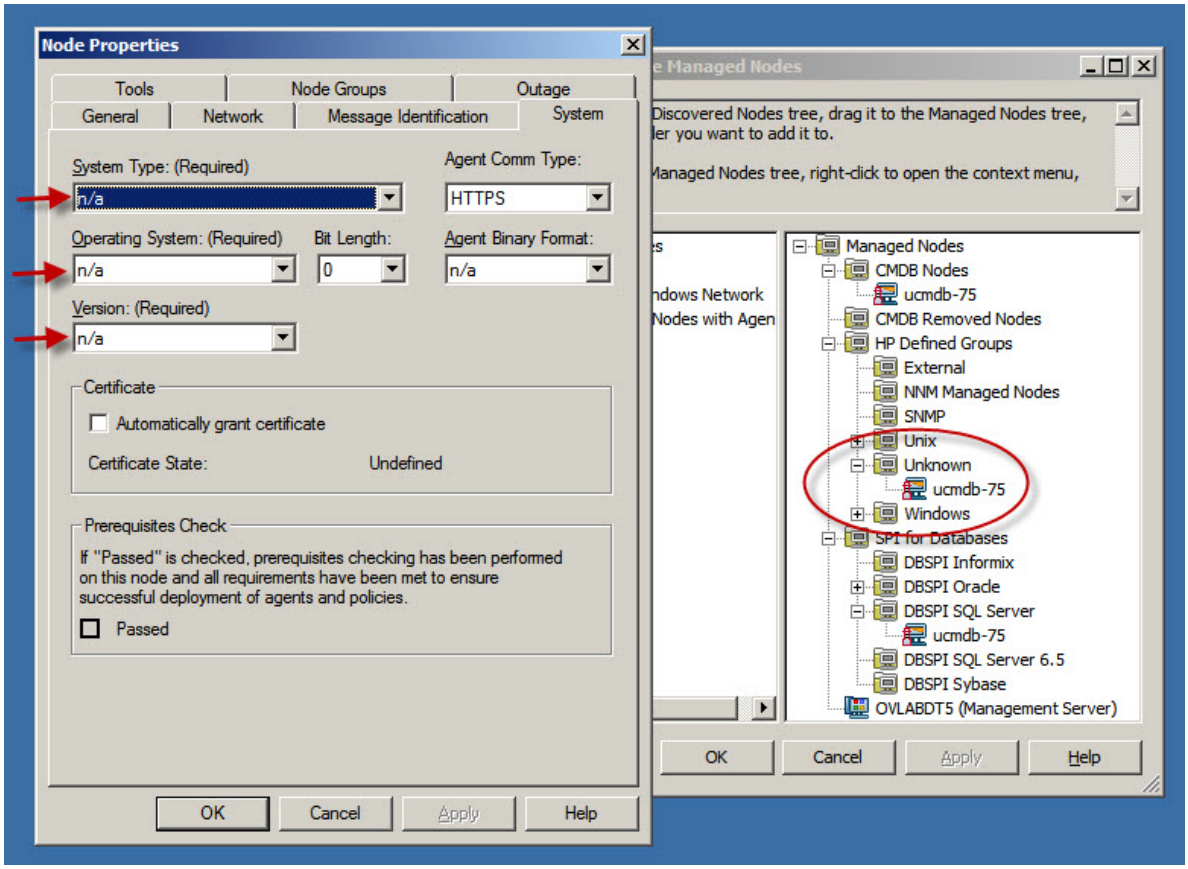
In most cases, we want to see nodes put into node groups which align with the major application that is running on the node. For example, if we have nodes that are hosting IIS, we might want them to appear in a node group called "IIS." For nodes running Apache, we would put them in a node group called "Apache." Nodes supporting a database would be put into node groups for the database type, e.g. "SQLServer" or "Oracle." It also might be desirable to have separate node groups for specific versions of these applications, e.g. "IIS 6.0" and "IIS 7.0." These two version-related node groups would most likely be subfolders under the generic service "IIS."

We would typically also want to see nodes in OS-related nodegroups, e.g. "Windows," "Unix," etc. When new nodes are added in HPOM on Windows (OMW), they are automatically placed into appropriate OS-related node groups (this is a function of OMW itself), but only if the required OS information exists. This is why it is important that the results returned by the UCMDB TQL for each host CI contain valid information in the "Host Model," "Host Operating System," and "Host Operating System Version" attributes. If these aren't provided to DMA during synchronization, then it will not be possible to determine which OS node group the node should belong to.

We should point out that there are differences between how node groups are setup in HPOM on Windows and HPOM on UNIX. In the former, there is a hierarchical or "folder" arrangement of node groups; in the latter there is just a simple "flat" arrangement of node groups. Since there are some special considerations for DMA, the following discussion will highlight the HPOM on Windows use case.

Consider the example in Figure 7 below. Since the needed OS information was not available, OMW placed the node ucmdb-75 into the "Unknown" node group in addition to DMA placing the node into "CMDB Nodes."

Figure 7. Example synchronized node in HPOM on Windows



Also note that the node has been put into node group "DBSPI SQL Server." This is because the "MSSQLServer" sync package was active during the synchronization and the mapping rules in the nodemapping.xml file determined that this node is hosting an instance of SQL Server.

The MSSQLServer sync package is worth reviewing since it contains a great example of an effective technique for handling node group assignments. Here's the contents of the nodemapping.xml file:

```
<?xml version="1.0" encoding="utf-8"?>
<Mapping>
     <Rules>
            <Rule name="SQL Server Nodes">
                <Condition>
                    <Equals>
                        <Attribute>host_servertype</Attribute>
                        <Value>sqlserver</Value>
                    </Equals>
                </Condition>
                <MapTo>
                    <NodeGroup>
                        <Value>DBSPI_SQL_Server_Nodes</Value>
                    </NodeGroup>
                </MapTo>
            </Rule>
     </Rules>
</Mapping>
```

This is a very simple mapping file with just one rule. The rule defines a *condition* and then an associated *mapping*. In the DMA user interface (under "Enrichment Summary"), this rule is summarized as follows:

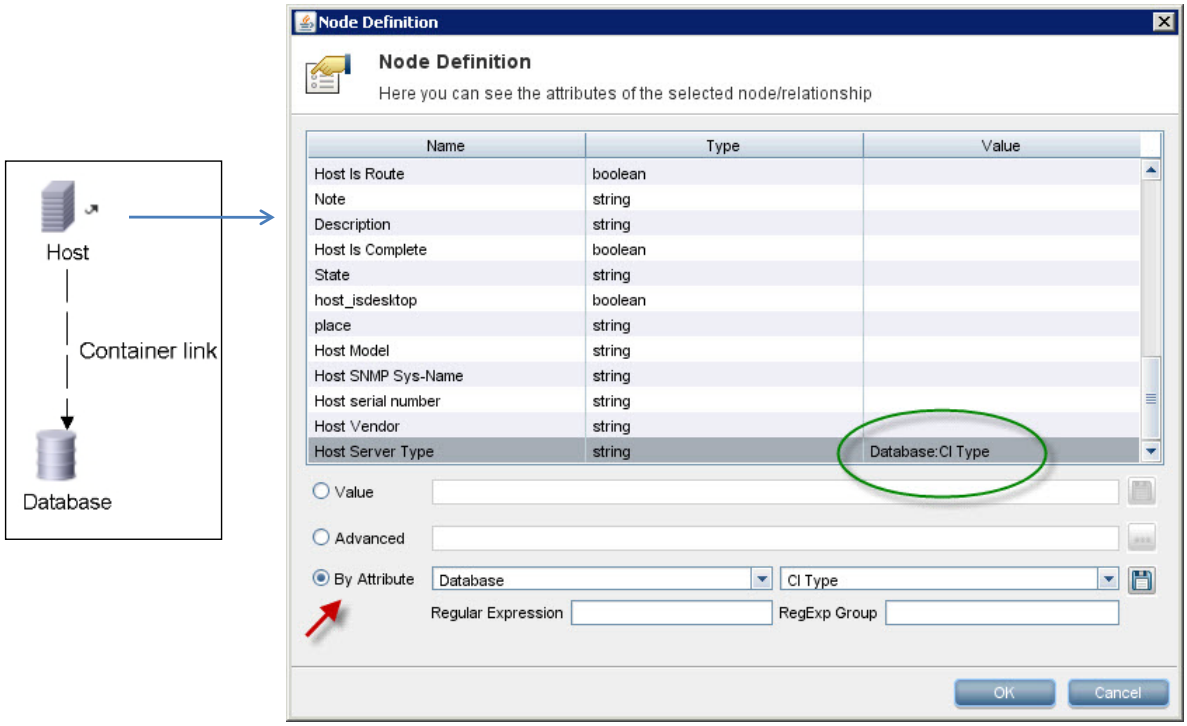
if
    the value of the attribute "host_servertype" equals the value "sqlserver"
then
    place the node into the node group named after the value "DBSPI_SQL_Server_Nodes"

Keep in mind that this rule is going to be applied to the complete stream of CIs and attributes that are being provided to DMA by the TQLs associated with active sync packages. The key to the simplicity of this rule is that we have an existing CI attribute ("host_servertype" in this case) that we can examine, and based on the value of that attribute we determine which node group the CI belongs to.

What may not be obvious, however, is that this attribute was actually initialized in the UCMDB using an enrichment rule provided by DMA. It turns out that this attribute is not typically populated with values by DDM, so to facilitate simplicity in our mapping rule we have "taken over" this attribute and designated it to be used for indicating which type of database is running on the host CI.

Here is the DMA-provided enrichment rule which initializes this attribute:

Figure 8. **NodeServerType** Enrichment Rule



This enrichment rule is also very simple: *Find all instances in the UCMDB of a Host which has a Container Link to a Database, then set the "Host Server Type" attribute of the Host CI to a value that is taken from the CI Type attribute of the Database CI.* In the example above (Figure 7), the enrichment rule found that ucmdb-75 had a Container Link to a SQL Server database and so the "Host Server

Type" attribute on ucmdb-75 was set to the value "sqlserver." During DMA synchronization, the simple rule in the nodemapping.xml file of the MSSQLServer sync package was then able to determine that ucmdb-75 should be put into node group "DBSPI_SQL_Server_Nodes."

You can use the same approach for your own applications if needed, however you would normally want to create an entirely new CI attribute in the CI Type model that you would then initialize with an appropriate enrichment rule. For example, to designate that a host is running IIS, you might create a new attribute called "web_server_flag" that you would then leverage in your mapping rules for node group assignment. The basic steps would be as follows:

- CI Type Manager: Add a new attribute "web_server_flag" to CI Type "host"; make sure it is marked "visible"
- Enrichment Manager: Create an enrichment rule to initialize the "web_server_flag" attribute to the value "iis" on hosts which contain IIS
- View Manager: Ensure that the new host attribute "web_server_flag" is exposed by the TQL being used to model the IIS application
- DMA Sync Package: Create a nodemapping.xml file in the package you are using to import IIS nodes and services into HPOM; include a rule which checks the "web_server_flag" (as in the SQL server example above) and designate the appropriate node group for the node

Before going through the effort of creating your own custom attributes, you should always examine the existing attributes for nodes in your TQLs to see if there is anything already there that could be used in the node mapping logic.

An important note on doing customizations with sync packages: It is *not* recommended to change the "default" sync package mapping files. All customizations needed for a specific customer environment should be implemented by either a) modifying one of the existing SPI sync packages (OS or database), or b) creating a new sync package. In general, the best approach would be to create a specific sync package for each application service hierarchy that you want in HPOM; this way you have better granular control over what happens during the synchronization process.

Automatic creation of node groups

By default, DMA assumes that the node groups you are mapping to already exist in HPOM. If a node group does not exist that is referenced during synchronization, an error will occur. On HPOM on Windows, it is possible to create node groups manually using the Node Configuration Editor, however this is not the recommended approach for DMA because you have no control over the Unique ID that will be assigned to the node group in HPOM. For example, if you manually create a node group "TEST GROUP," its unique ID may be something like {5A6030E3-FB14-4675-A3EA-BACF504A8890}. This is not very user-friendly for use in mapping files!

A better approach (again on HPOM on Windows) would be to create your own MOF files where you could designate node group names that are easier to read and reference. Prior to DMA synchronization, you would upload the node group configuration into HPOM using the mofcomp command line interface to get the definitions into the HPOM model. This would of course need to be done only once. The problem with this approach is that you need to understand the usage of mof files and the instance definition syntax for node groups.

The easiest approach of all (for both HPOM on Windows and HPOM on UNIX) is to simply have DMA create the node groups automatically if the target node group does not exist. On HPOM for Windows, the only downside of this approach is that the node groups will be created at the top level of the node group hierarchy which might not be the desired structure. But you can later go into the

Node Configuration Editor and move the new groups under the desired parent groups (except for the standard HPOM-provided groups which are "read only").

To setup DMA for automatic creation of node groups, you need to make a change to the XPL configuration on HPOM. From a command window, issue the following command:

```
ovconfchg –ovrg server –ns opc.WebService.ConfigurationItem
-set NodeGroupCreationEnabled true
```

You must then restart the Tomcat server as follows:

```
ovc –restart ovtomcatB
```

Thereafter, any non-existing node groups referenced by DMA mapping rules will be automatically created.

Synchronizing services

Service Type Definitions

HPOM on Windows introduced the concept of Service Type Definitions as the mechanism for creating services in the service model. In HPOM on Windows, referencing a STD is the ONLY way to instantiate a service. DMA must therefore use this mechanism during synchronization to get new services created. HPOM on UNIX does not itself implement the notion of STDs in the service model, however installing DMA on HPOM on UNIX will insert new service definitions into the service model which in effect represent STDs. In this way DMA can use the same mechanism and syntax in the service mapping files for both platforms.

You can think of STDs as templates for use when instantiating specific kinds of services in HPOM. The STD specifies the assignment of default calculation rules for status and propagation, the icon to be displayed in the Service Navigator, and tools to be associated when a new service is created. One nice aspect of this design is that you can easily modify ALL services created with a specific STD in your service model by simply changing the STD. In HPOM on Windows, you can do this manually using the Service Type Configuration Editor.

When you install DMA, a number of STDs for use with the OS and Database SPI sync packages are automatically loaded into the HPOM service model. (See Appendix A in the **DMA Installation and User Guide** for a complete list.) For example, here is the STD for creating a service representing a Unix host:

```
instance of OV_ServiceTypeDefinition
{
   CalcRuleId = "ucmdb_generic_CR";
   Caption = "Unix";
   CaptionFormat = "folder";
   Description = "Mapping of corresponding UCMDB type";
   GUID = "ucmdb_unix";
   Icon = "Unix.ico";
   KeyFormat = "folder";
   MsgPropRuleId = "ucmdb_generic_PR";
};

instance of OV_ServiceTypeComponent
{
   GroupComponent = "OV_ServiceTypeDefinition.GUID=\"folder\"";
   PartComponent = "OV_ServiceTypeDefinition.GUID=\"ucmdb_unix\"";
   PropRuleId = "ucmdb_generic_PR";
};
```

```
instance of OV_ServiceTypeDependency
{
   Antecedent = "OV_ServiceTypeDefinition.GUID=\"ucmdb_unix\"";
   Dependent = "OV_ServiceTypeDefinition.GUID=\"folder\"";
   PropRuleId = "ucmdb_generic_PR";
};
```

Notice that this STD also specifies how the service can be related to other services. In this case the template will allow *containment* from other services (OV_ServiceTypeComponent) and *dependency* from other services (OV_ServiceTypeDependency).

To complete the picture, here is the service mapping file from the "default" DMA sync package:

```
<?xml version="1.0" encoding="utf-8"?>
<Mapping>
    <Rules>
        <Rule name="ServiceGroup">
             <Condition>
                 <Equals ignoreCase="true">
                      <CiType />
                      <Value>servicegroup</Value>
                  </Equals>
              </Condition>
              <MapTo>
                 <STD>
                      <Value>folder</Value>
                  </STD>
              </MapTo>
        </Rule>
        <Rule name="Default">
            <Condition>
                <True/>
            </Condition>
            <MapTo>
                <STD>
                     <Value>ucmdb_</Value>
                     <CiType/>
                </STD>
            </MapTo>
        </Rule>
    </Rules>
</Mapping>
```

The second rule illustrates the power (and simplicity) possible in mapping rules: *For CIs of type "unix" encountered in the TQL result set, the rule concatenates the strings "ucmdb_" and "unix" and creates a new service in HPOM by referencing the "ucmdb_unix" STD.* Keep in mind that the service mapping file is only concerned with mapping the CI to a service type. Attributes like the service name and display label for the service are determined by rules in the appropriate attribute mapping files.

Automatic creation of Service Type Definitions

The default in DMA is that you must have STDs already defined in the HPOM model before you can reference them in the sync package mapping rules. You can reference any of the existing STDs in HPOM. To see what DMA has loaded into the service model on HPOM on Windows, examine the following file:

```
<InstallDir>\misc\dma\moffiles\en\default.mof
```

On HPOM on UNIX, you can use the "ServiceTypeDefinitionCLI" command to list the available STDs, or examine the following files which are used at install time to upload into the service model:

```
/opt/OV/misc/dma/default_calprop.xml
/opt/OV/misc/dma/default_std.xml
```

You may want to use your own STDs for synchronizing custom applications. In this case, you have two options:

1. Create mof files (HPOM on Windows) or xml files (HPOM on UNIX) containing the definitions and upload them to the service model before you synchronize, or

2. Enable automatic STD creation

Option 2 is the easiest, and can be accomplished by changing the XPL configuration with the following command:

```
ovconfchg –ovrg server –ns opc.WebService.ConfigurationItem
-set StdCreationEnabled true
```

As is usually the case when changing XPL configuration, you must then restart the Tomcat server:

```
ovc –restart ovtomcatB
```

After this change is made, if your service mapping rules reference a STD that does not exist, it will be created automatically with a default configuration in the service model. More information on creating and using STDs can be found in Chapter 4 of the **DMA Extensibility Guide**.

Overriding defaults of Service Type Definitions

In certain situations, it may be desirable to override defaults of a STD which you reference in the service mapping rules. Here are some example scenarios where this might be useful:

- You want to use custom icons in the service map for the services you create
- You want to modify the status calculation rule for a service to more accurately reflect how it is affected by subordinate services in the hierarchy (e.g. setting a cluster service calculation rule to “least critical” instead of the default “most critical”)

The service mapping syntax allows for overriding the following STD elements:

- Calculation rule
- Message propagation rule
- Parent propagation rule
- Icon
- Message weight
- Parent weight

The Parent propagation and Parent weight rules determine the propagation to the parent of the service. The Message propagation and Message weight rules determine the propagation of the incoming HPOM messages to the service.

Of course you could also manually create and upload a new STD with the desired custom settings and then reference it in your service mapping rules, but it’s much easier and less effort to specify overrides of an existing STD.

Consider the following simple example of how to specify overrides in the service mapping rules. In this case we want to change the default Message propagation rule and also specify a custom icon for an Apache server CI that resides in a cluster.

```
<?xml version="1.0" encoding="utf-8"?>
<Mapping>
   <Rules>
      <Rule name="Apache Server">
         <Condition>
            <And>
               <Equals ignoreCase="true">
                  <CiType />
                  <Value>apache</Value>
               </Equals>
               <Equals>
                  <Attribute>cluster</Attribute>
                  <Value>true</Value>
               </Equals>
            </And>
         </Condition>

         <MapTo>
            <STD>
               <Value>ucmdb_</Value>
               <CiType />
               <Value>_std</Value>
            </STD>
            <MessagePropagationRule>
               <Value>least_critical</Value>
            </MessagePropagationRule>
         </MapTo>
      </Rule>
      <Rule name="Set Icons">
         <Condition>
            <True />
         </Condition>

         <MapTo>
            <Icon>
               <CiType />
               <Value>_icon</Value>
            </Icon>
         </MapTo>
      </Rule>
   </Rules>
</Mapping>
```

This service mapping file has two rules which implement the following logic:

- Rule 1 (“Apache Server”)

  If the CI is of type “apache” and the CI attribute “cluster” has a value of “true,” then create a new service referencing STD “ucmdb_apache_std” and override the STD message propagation rule to use “least critical.”

- Rule 2 (“Set Icons”)

  For all CIs (the condition part of the rule is always “true”), set the icon name to be the concatenation of the CI Type string and “_icon”. For apache servers, the icon name would be evaluated to “apache_icon”.

Note that the results from Rule 2 may be overwritten by higher priority rules during synchronization. Service mapping rules are executed according to the priority of the bundle (sync package) and the position within the rule declarations; rules are applied in reverse order, i.e. the rules in the XML file are processed bottom up. If you wanted the icon logic to apply only to the Apache server CIs, you would modify the second rule's condition part to be the same as in the first rule.

For a more complex example of using STD overrides, see Chapter 8 "Service Mapping" in the **DMA Extensibility Guide**.

Extending existing HPOM services with data from UCMDB

In customer environments where important HPOM service hierarchies already exist, DMA supports the concept of extending these hierarchies by establishing dependencies to and from the data being imported from the UCMDB. DMA uses the following terms to define "ownership" of service CIs:

- ExternalCI

  Services which already exist in HPOM and are maintained completely outside of the DMA synchronization process

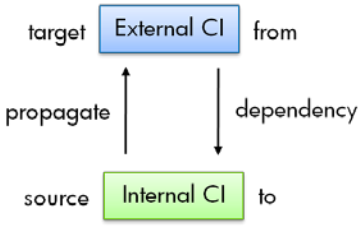
- InternalCI

  Services created and managed by the DMA synchronization process

In the service mapping file, you can establish dependency relationships (in either direction) between Internal and External CIs using special mapping rule syntax. Before we look at an example of service mapping rules that accomplish this, we need to first clarify how the concepts of dependency and propagation relate to each other. You need to know the "from/to" semantics in order to understand how the rules are written.

First, let's consider the case of establishing a dependency from External to Internal CIs. Figure 9 illustrates the concept and terminology:

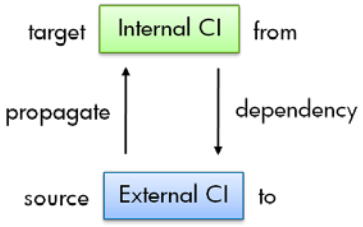
Figure 9. Dependency from External to Internal CIs



It may be counterintuitive, but note that the direction of propagation and dependency are opposite to each other. This is important to grasp. The *target* for propagation is where the dependency originates. You would use this type of dependency setup when you want an existing service in HPOM to be the propagation target from the new service CI being imported from UCDMB. (Note that when you implement this type of dependency, DMA can only create or update the relationship; it cannot delete it during synchronization. Deletion of these dependencies must be done manually in HPOM.)

In Figure 10, we have the opposite situation, i.e. dependency from Internal to External CIs:

Figure 10. Dependency from Internal to External CIs

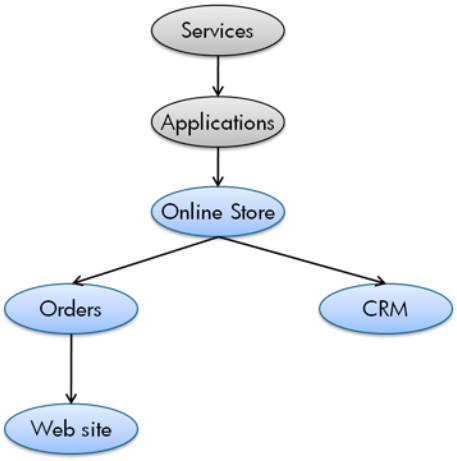


Again (it's worth repeating!), the *target* for propagation is where the dependency originates. You would use this type of setup when you want the new service CI from UCMDB to be the propagation target from an existing service in HPOM. In this type of dependency, all aspects of managing it (create, update, delete) can be accomplished within the DMA synchronization process.

The above discussion is important because when designing service hierarchies in HPOM we tend to think in "propagation" terms, however the semantics of the service mapping rules in DMA are in "dependency" terms.

Let's take a look at a simple example scenario and the required service mapping rules to implement it. Our scenario is as follows:

> An existing service model for managing a web-based online store has already been implemented in HPOM. Most of the key infrastructure supporting this application is already covered in the service model, however it is missing some important services (specific oracle databases) which are not yet included. Since DDM has discovered all the databases and related topology and populated the UCMDB, we now want to use DMA to enhance the existing web-store service tree to include the appropriate oracle database dependencies.
>
> Here's the current service tree in HPOM (in real life this would typically be quite a bit more comprehensive):



Our goal is to extend the service model to include a specific database, "MI6," as an important component which supports the Customer Relationship Management (CRM) service.

To accomplish the desired results, we need a dependency relationship from CRM (an External CI) to the database "MI6" (which will be an Internal CI once it has been synchronized). The easiest way to implement this is to modify the existing service mapping file in the Oracle Database sync package provided by DMA. We would add a new mapping rule as follows:

```
<Rule name="Create dependency from CRM to MI6">
  <Condition>
    <And>
      <Equals ignoreCase="true">
        <CiType />
        <Value>oracle</Value>
      </Equals>
      <Equals ignoreCase="true">
        <CiCaption />
        <Value>MI6</Value>
      </Equals>
    </And>
  </Condition>
  <MapTo>
    <DependencyFromExternalCi>
      <ExternalCiId>
        <Value>store_crm</Value>
      </ExternalCiId>
      <DependencyType>
        <Value>Dependency</Value>
      </DependencyType>
      <PropagationRuleName>
        <Value>ucmdb_generic_PR</Value>
      </PropagationRuleName>
      <PropagationWeight>1.0</PropagationWeight>
    </DependencyFromExternalCi>
  </MapTo>
</Rule>
```

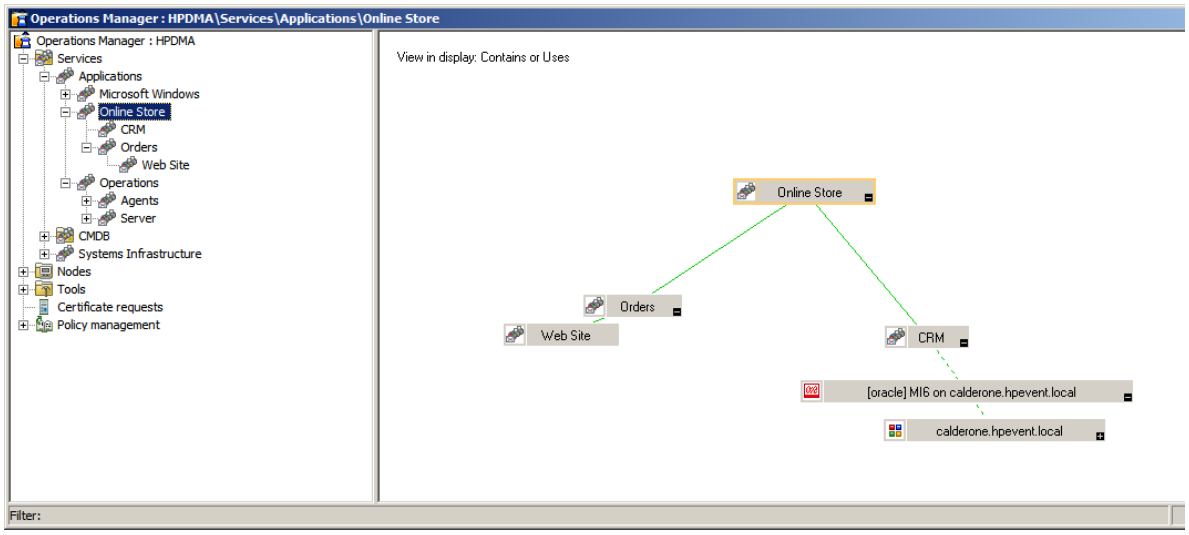
Here is the translation of this new rule in the DMA user interface:

if
the type of the CI equals the value "oracle"
and
the CI caption equals the value "MI6"

then
create a dependency from an external CI to this CI. The CI ID of the external CI is the value "store_crm". The dependency type is the value "Dependency". Also assign a propagation rule to the dependency. The name of the rule is the value "ucmdb_generic_PR". The rule weight is "1.0".

Note that the "ucmdb_generic_PR" propagation rule must already exist in HPOM (in this case it is one of the defaults loaded when DMA is installed). With this rule included in our Oracle sync package service mapping file, DMA synchronization would produce something similar to the following service tree:

Figure 11. Example results from service mapping dependency rule



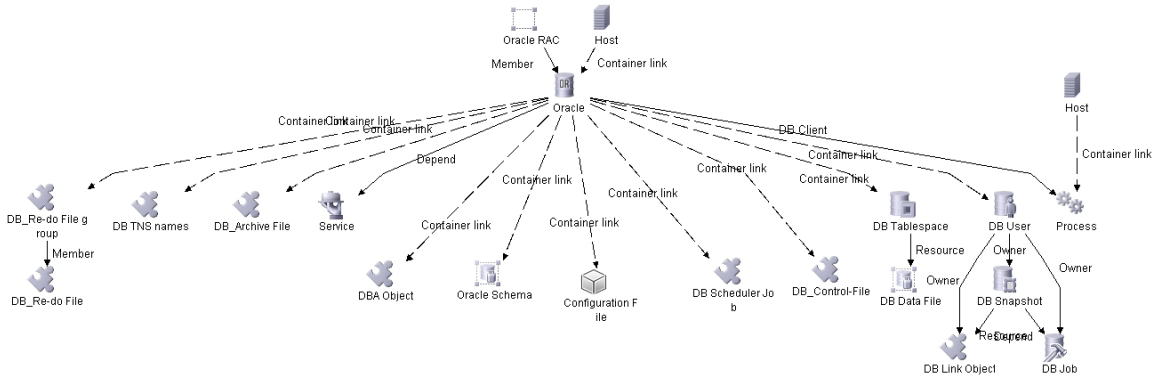
In this example, we can also see that MI6 is hosted on a node named "calderone.hpevent.local" and that a dependency from the database to the node has also been established in the service hierarchy (as a result of the data returned by the TQLs used in this synchronization). For more details on establishing dependencies between existing services and CIs being synchronized with DMA, see Chapter 8 "Service Mapping" in the **DMA Extensibility Guide**.

Creating the desired dependency and propagation in HPOM

As stated earlier, the UCMDB type model implements many different relationship types which can be used when instantiating discovered CIs. In most cases, DDM will create a model of CIs and relationships which has a top-down or "global-to-detailed" view of the world. This is largely the result of the spiral discovery design of DDM. Newly discovered CIs tend to be treated as being a component of and contained by a parent CI.

An example of this is how databases are discovered in relationship to hosts. The host is discovered first, then the database. So the typical model in UCMDB will have a "Container Link" relationship between a host and a database. The standard out-of-box Oracle TQL in UCMDB illustrates this:

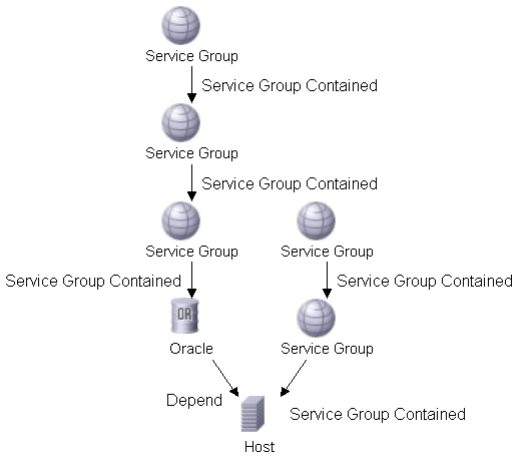
Figure 12. UCMDB "Oracle" TQL



This can be a useful way of looking at the Oracle application environment from a CI perspective, but for HPOM this view is less desirable because it doesn't reflect the service impact design of HPOM's model. In HPOM, we want to show a *dependency* from Oracle to the host, i.e. we want a problem with the host to propagate to Oracle. The TQL above does not accommodate the notion of a problem on the host having impact on the database.

Luckily we have a great deal of flexibility in UCMDB to modify and enhance the discovered CI model to suit our needs. For synchronizing Oracle CIs, DMA provides the following TQL which gives us the desired propagation and dependency relationships:

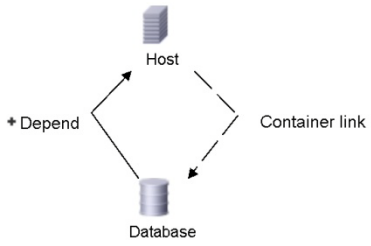
Figure 13. DMA "Oracle (Operations)" TQL



This TQL is possible as a result of DMA's UCMDB enrichment rules which modify the existing model to include the needed relationships. Enrichment is an extremely powerful concept in UCMDB and it's fair to say that without it, DMA would have greatly reduced value to HPOM customers.

Here is the DMA enrichment rule that makes the TQL possible:

Figure 14. **DataBaseDependency** Enrichment rule



It's a very simple enrichment rule, but extremely powerful: *For all instances of a host with a container link to a database, add a new "depend" relationship from the database to the host.* Once this enrichment has been applied to the UCMDB data, we can then use the DMA Oracle TQL. (We should point out that there are actually several other enrichment rules that must run to fully enable use of DMA Oracle TQL; these additional enrichments create the Service Group instances and relationships which are used to represent the virtual services, e.g. "Applications," "Systems Infrastructure," "Windows," etc. in our target HPOM model.)

The "depend" relationship is a standard relationship type in UCMDB, so we did not have to add it to the type model. However, by default the UCMDB does not allow a "depend" relationship between the database CI Type and the host CI Type, i.e. it is not a *valid link*. So even before we created the enrichment rule, it was necessary to enable this link. Valid links can be configured using the CI Type Manager, or as is the case with DMA, they can be configured into the UCMDB by importing a pre-built UCMB package with the Package Manager. DMA's "hpdmadb.zip" UCMDB package includes the necessary link for "Depend: Database → Host".

To summarize for the database example, the following sequence of UCMDB tasks was necessary in order to deliver useful data to DMA and HPOM:

1. Configure a new valid link for the depend relationship
2. Design and execute an enrichment rule to instantiate the desired depend relationship in the topology
3. Design the TQL to return the modified topology

The key thing to be learned from the above out-of-box example is that when creating TQLs and sync packages for custom applications, you may need to modify the UCMDB model to ensure that you will ultimately be able to create the desired dependencies and propagation behavior in HPOM. Furthermore, you will need to pay attention to the sequence of UCMDB modifications since they build on each other.

How to get optimum results with Smart Message Mapper

The Smart Message Mapper (SMM) is an important part of DMA since it allows the continued use of currently deployed HPOM SPI policies without having to modify them to reference the new service IDs of CIs that have been synchronized into the service model. For a good summary of SMM concepts, see Chapter 3 of the **DMA Installation and User's Guide**.

Smart Message Mapper is implemented as a MSI program on the HPOM server and exists to do essentially one thing: map incoming messages to the most appropriate service in the service tree. All messages are input to SMM whether or not they include a service ID, so it provides the added benefit of making it easier to leverage the service views in HPOM. SMM looks for hints in each incoming message to help identify the target service which the message should impact. The hints include the service name (service ID), object, application and node name attributes.

Before describing how to make SMM "smarter," we should first review the basic SMM algorithm.

SMM service matching algorithm

The service matching algorithm can be summarized by the following sequence of steps:

1. If the message includes a service ID, look for an exact match of this service ID in the service model. If there is a match, the message is sent to that service. (This first step in the algorithm ensures that as long as the service model still has the SPI-discovered services, the assignment of messages to services remains unchanged.)
2. If there is no exact match of service ID, SMM will then extract hints from the service name, object, and application attributes of the message and compare these hints with the existing *service* attributes of all services. When the best matching service is found, SMM replaces the service ID in the message with the service ID of the matching service and forwards the modified message.
3. If SMM cannot find a matching service, it will then consider the node name attribute of the message. If it can find a matching host in the service model, it forwards the message to that host (again the service ID will be modified to match the service ID of the host service in the model). If multiple matches are made on services which are connected to the same host, SMM will forward the message to the host service (and also set an explanatory CMA).
4. If SMM cannot find either a matching service or node name, it will forward the message unchanged (no changes are made to the service ID).
5. Even if a message does not contain a service ID, SMM still considers the object, application, and node name attributes as hints and attempts to find a matching service by comparing with the attributes of all services.

As an example of how the hints are extracted, consider the following attributes from a typical message from a Database SPI policy:

```
a=Oracle
o=G11
msg_t="ORASPI-0001.3: DB-SPI cannot connect to database G11, may be down; Oracle
error [ORA-12560:_TNS:protocol_adapter_error].   [Policy: DBSPI-0001]"
severity=critical
service_id=Oracle_G11_@@dbhost1.acme.com
node=dbhost1.acme.com
```

The hints used by DMA in this message are highlighted.

SMM and the "hosted_on" attribute

Regarding service attributes, there is one important guideline which needs to be followed up front to ensure SMM works effectively:

- All hosted services in the HPOM model MUST have the "hosted_on" attribute set to the fully qualified domain name (FQDN) of the host. Remember that a hosted service includes both the node itself and any services hosted on that node. For example, if we have an Oracle database service "MI6" which is hosted on the node "calderone.hpevent.local," both of these services in the model must have the hosted_on attribute set.

If the hosted_on service attribute is not properly initialized, SMM will have difficulty in matching messages to services. The good news is that DMA already provides two sync packages which help you with this requirement: the DNS Mapping package (which contains a pre-mapping script to identify DNS names of hosts) and the Default sync package (attribute mapping file) which includes a rule that sets the hosted_on attribute for appropriate services.

Attribute mapping rules

Remember from the discussion in Part I above that the attribute mapping file in a sync package is the place for mapping CI attributes to service attributes and creating new service attributes. To understand the basics of how attribute mapping rules work, let's take a look at the following rule from the Default sync package:

```
<Rule name="Set hosted_on attribute for nodes">
    <Condition>
        <IsNode />
    </Condition>
    <MapTo>
       <Attribute>
          <Name>hosted_on</Name>
          <SetValue>
               <XPathResult>./attributes/host_dnsname</XPathResult>
          </SetValue>
       </Attribute>
    </MapTo>
</Rule>
```

This is a simple rule on the surface, but it also demonstrates a more advanced capability of using XPath expressions in the mapping rules. This is the rule that sets the hosted_on attribute for all node CIs that are processed during DMA synchronization. The "Condition" element evaluates to true if the CI is a node, and the "MapTo" element sets the hosted_on attribute to the value string returned by the XPath expression. The XPath expression operates on the XML structure of the normalized "pre-mapped hierarchy" which is created during synchronization (see Figure 6 above).

Equally important to the rule above for services representing nodes, there is also a Default sync package attribute mapping rule to set the hosted_on attribute for services (i.e. CIs that are NOT nodes). (This rule also illustrates a more complex usage of XPath expressions and expanded condition logic and can serve as a good example for learning more advanced XPath techniques.)

To get a better understanding of how XPath expressions work, see Appendix A in the **DMA Extensibility Guide**; you may also want to have a look at this tutorial: http://www.w3schools.com/xpath/. A word of caution is in order regarding using XPath

expressions in your mapping rules: they can have a negative impact on performance of DMA synchronization if not used carefully!

Service attributes – The key to making SMM smarter

If you observe a message that is mapped to the wrong service (or perhaps is not mapped at all but you expected a mapping), there are several steps you need to take (we are assuming that MSI has already been properly configured in HPOM and the SMM process is running):

- Verify that the expected target service actually exists in the service model.

- If it exists, you then need to verify that SMM is properly configured for how to extract the hints from the incoming message. SMM looks for separators (e.g. ":" or "@@") in the message attributes to extract each individual hint. Starting with DMA version 8.20, you can now configure these SMM parameters in the user interface in the "Configuring DMA" section. If the separators in the message are different than what SMM expects, the hints will not be properly extracted.

- Determine if you have case issues. By default SMM considers case when comparing hints with attributes. In general, you would probably want SMM to be case-insensitive since some parts of the infrastructure (e.g. DNS servers) operate without case sensitivity. You can configure SMM to be case-insensitive in the "Configuring DMA → Smart Message Mapper" page of the DMA user interface. Note that there may be a short delay before this takes effect while SMM rebuilds its internal hash table of services.

- If the separators are correct and there are no issues with case sensitivity, and if you have just recently updated the service model with the target service, you need to ensure that SMM has updated its internal hash table which it uses for the search for a match. By default, this table is updated within 60 seconds of a change in the service model, however once the table is updated, the minimum delay for the next update is 15 minutes. This means that there is a window of up to 15 minutes where the target service may exist in the model, but NOT in the internal SMM hash table. When testing with SMM, it is recommended to always force an update to the table by changing the "minimum delay" parameter in DMA ("Configuring DMA" section) after you have synchronized new services.

- *If all of the above are OK, you will then need to go a level deeper and begin looking at the message hints and service attributes of the target service. If the service attributes have wrong or misleading values or if there aren't enough hints or attributes for comparison, SMM may not be able to identify a match or make the best match.*

Having the right service attributes with the right values is the key to getting expected results with SMM. There are several ways to go about verifying service attributes and values. The first approach is to leverage DMA's ability to create dump files (XML) of the "internal" CI hierarchy used for uploading nodes and services into HPOM (refer to Figure 6). The dump files are very useful for diagnosing the following types of problems:

- expected attributes not written to HPOM
- problems with mapping rules
    - Wrong attributes
    - Nodes in wrong node group
    - Services not built in HPOM due to non-existing STD

- Mapping rules are not executed
- Smart Message Mapping does not highlight expected service

Dumping of synchronization files is enabled by changing the "sync.dumpData" parameter in the following file:

```
<SharedDir>/conf/dma/DefaultSyncTask.settings
```

It's a good idea to enable dumping only while debugging problems; make sure you turn off dumping when DMA is in production. The dump file will show exactly which attributes and associated values will be uploaded into HPOM, but since the files are XML it may be difficult to interpret them. A good XML editor which presents the data in a more readable format can make this easier. More details on how to examine the dump files can be found in Chapter 13 "Testing and Deployment" of the **DMA Extensibility Guide**.

An alternative method for verifying service attributes would be to just examine HPOM itself to see the results of the upload. In HPOM on Windows, there is unfortunately no possibility in the GUI to see the service attributes which have been instantiated since attributes are not exposed in the service properties window. The easiest way around this is to download the free Microsoft WMI Tools package which includes the WMI CIM Studio application. This tool allows you to connect to the WMI namespace used by HPOM on Windows and enables navigation and display of the various objects. With a little bit of exploring, you should be able to find the service instances and the attributes. CIM Studio can also *modify* the WMI repository, so be extremely careful that you do not make inadvertent changes which could render your HPOM environment unusable.

On HPOM on UNIX, we have a much easier solution since it is very easy to see the service attributes from the JAVA GUI interface. All you have to do is simply right-click on a service in the Service Navigator and select Properties, and then select the Attributes tab.

Improving the odds for a match

After you've examined the attributes for the target service and compared them to the message hints from the problem message, you may then be able to conclude that SMM needs some help to ensure the algorithm succeeds in making a match. Remember that the SMM algorithm will look at ALL service attributes when it compares against the message hints. So to increase the odds, you will need to do one of two things, or maybe both:

1. Add additional service attributes to the target service and initialize with appropriate values

   You could choose to map one of the existing CI attributes which is included in the TQL results, assuming its value will match a message hint. Or you could simply create a new service attribute of your own and initialize it with a value that will match a message hint. In both cases you would accomplish this with additional rules in the attribute mapping file of the appropriate sync package.

2. Expand the application or object message attributes to include additional hints

   If you have the option, and if you think the service model contains sufficient service attributes on the target service, you might consider changing the message itself to include more hints. This would mean expanding the "application" and/or "object" attributes emitted by the monitoring policy on the HPOM agent where the message originates.

   For example, instead of `object=disk`, you might get better results with `object=disk:root` or something similar which further distinguishes the service. In this case you would have two hints

("disk" and "root") instead of just one. In any case, you would need to ensure that the additional hint(s) would actually match with a service attribute on the target service.

The bottom line for SMM is that it needs to have adequate hints and adequate service attributes (each with appropriate values) to ensure the matching algorithm works optimally. In environments with large and complex service hierarchies, it is likely that some tuning of SMM (in the form of adding hints and service attributes) will be required.

Exposing service attributes as CMAs (new in DMA 8.20!)

One of the really nice new features in DMA 8.20 is the ability to expose synchronized service attributes as Custom Message Attributes (CMAs) in the HPOM browser. This can be a really powerful way to provide additional information from the UCMDB that operators can use in the process of resolving problems. Instead of manually launching a tool from a message to drill into the UCMDB to get CI properties, the operator simply views the appropriate attributes directly in the browser as CMAs embedded in the message.

This new feature is a function of the Smart Message Mapper. If SMM is configured for CMA mapping, it retrieves the specified service attribute and its corresponding value and copies this information into the message before sending it on to HPOM. Here are the steps required to make this work:

1. Decide which UCMDB CI Type attribute you want to expose in the HPOM browser.
2. Ensure that this attribute is exposed by the appropriate DMA TQL (in Node Properties, Advanced Layout Settings)
3. Create a new attribute mapping rule in the appropriate sync package which maps the UCMDB CI attribute to a service attribute in HPOM
4. Synchronize
5. Configure SMM to attach the service attribute as a CMA
6. Configure the HPOM browser to show the desired CMA as a column
7. Test the configuration by sending an appropriate message to HPOM (ensure that SMM has rebuilt its internal hash table of services and service attributes before you send the message)

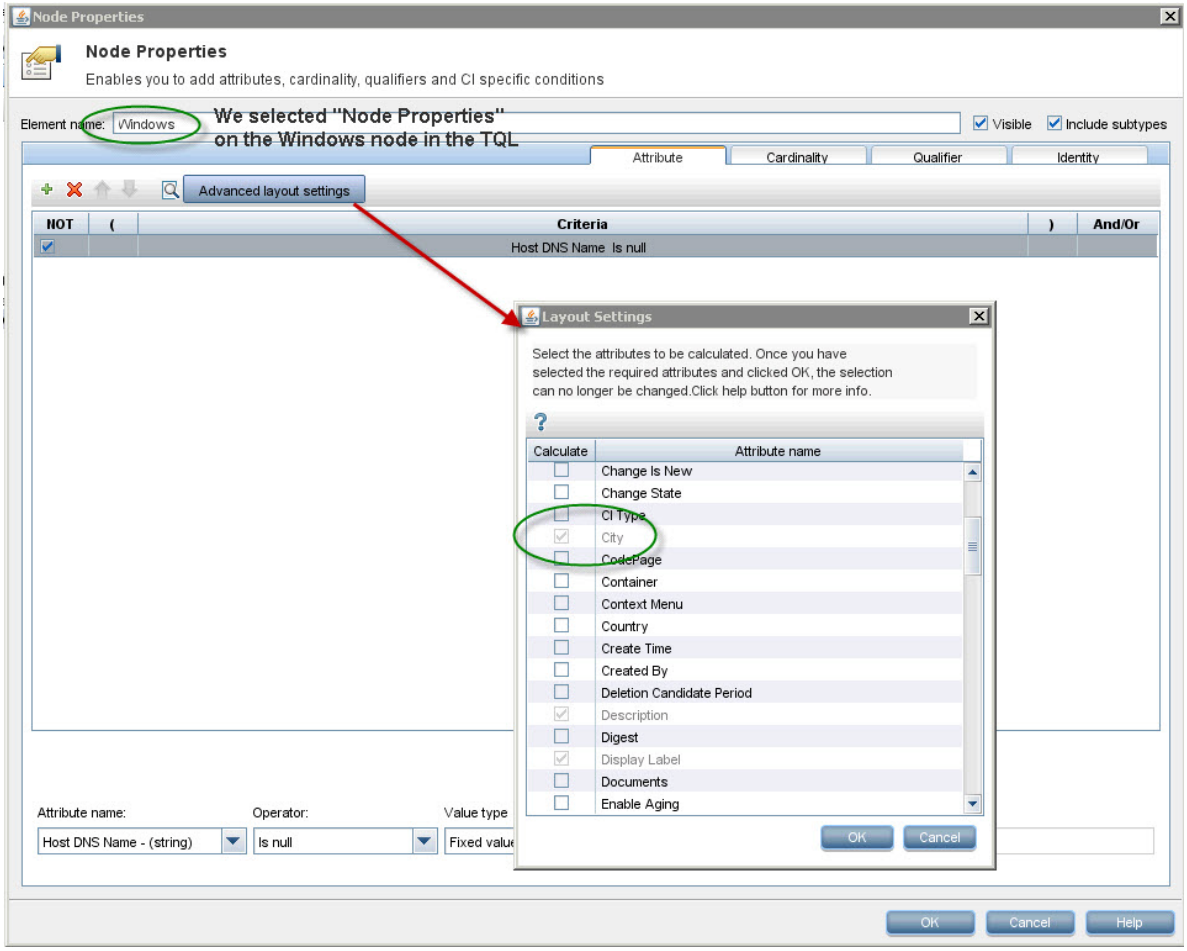
Here's a simple example scenario to illustrate the process:

UCMDB includes a "city" attribute on host CIs which can be used to track the physical location of systems. We want this attribute to be available in the HPOM message browser so that an operator can see at a glance where an affected node is physically located. (We are assuming of course that DDM or some other discovery mechanism has populated this attribute with appropriate values.)

The first step is to ensure this attribute is exposed by the appropriate DMA TQL. For this example, we'll make the required modifications to the out-of-box "dmaWinOS" sync package, which means that all synchronized Windows host CIs will get the "city" attribute.

Figure 15 below shows that we have marked the "city" attribute to be exposed in the "Windows Operating System (Operations)" TQL results.

Figure 15. Exposing CI attributes in TQL results



Next, we create a simple attribute mapping file with one rule as follows:

```
<?xml version="1.0" encoding="utf-8"?>
<Mapping xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <Rules>
    <Rule name="Set city location attribute">
      <Condition>
        <IsNode />
      </Condition>
      <MapTo>
        <Attribute>
          <Name>cmdbCity</Name>
          <SetValue>
            <Attribute>city</Attribute>
          </SetValue>
        </Attribute>
      </MapTo>
    </Rule>
  </Rules>
</Mapping>
```

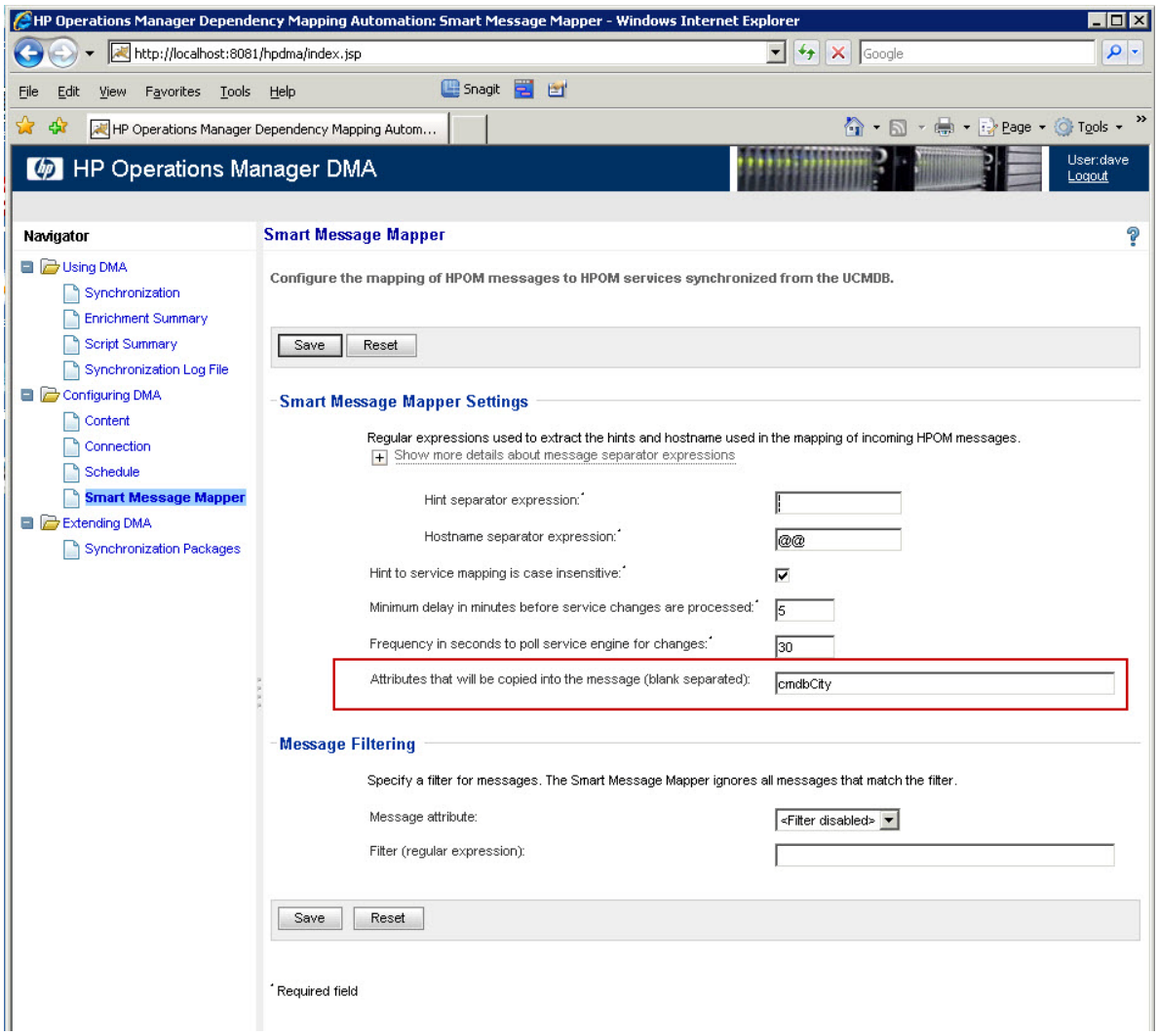
DMA translates this rule as follows:

if
    the CI is a node
then
    assign the value of the attribute “city” to the attribute “cmdbCity”

This mapping file would be saved as “attributemapping.xml” into the “dmaWinOS” sync package directory on the DMA system.

Synchronization at this point will create the “cmdbCity” attribute on Windows host services in the HPOM service model, however we aren’t quite done yet. As a final step, we need to configure SMM to include the “cmdbCity” attribute as a CMA on appropriate messages. Figure 16 illustrates how the SMM configuration is accomplished:

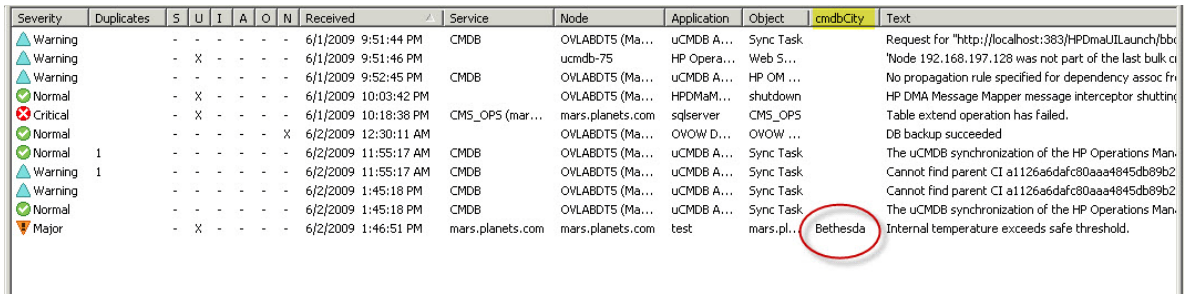
Figure 16. Configuring SMM to create CMAs



We are now ready for a test. In the figure below, we can see the results of sending a message to HPOM that is matched by SMM to the service “mars.planets.com”, a Windows node synchronized by

DMA. In the browser, the CMA "cmdbCity" has been set to the value of "Bethesda" as instantiated in the UCMDB.

Figure 17. Assignment of CMA to matched message



Severity	Duplicates	S	U	I	A	O	N	Received	Service	Node	Application	Object	cmdbCity	Text
Warning		-	-	-	-	-	-	6/1/2009 9:51:44 PM	CMDB	OVLABDT5 (Ma…	uCMDB A…	Sync Task		Request for "http://localhost:383/HPDmaUILaunch/bb
Warning		-	X	-	-	-	-	6/1/2009 9:51:46 PM		ucmdb-75	HP Opera…	Web S…		'Node 192.168.197.128 was not part of the last bulk c
Warning		-	-	-	-	-	-	6/1/2009 9:52:45 PM	CMDB	OVLABDT5 (Ma…	uCMDB A…	HP OM …		No propagation rule specified for dependency assoc fr
Normal		-	X	-	-	-	-	6/1/2009 10:03:42 PM		OVLABDT5 (Ma…	HPDMaM…	shutdown		HP DMA Message Mapper message interceptor shuttin
Critical		-	X	-	-	-	-	6/1/2009 10:18:38 PM	CMS_OPS (mar…	mars.planets.com	sqlserver	CMS_OPS		Table extend operation has failed.
Normal		-	-	-	-	-	X	6/2/2009 12:30:11 AM		OVLABDT5 (Ma…	OVOW D…	OVOW …		DB backup succeeded
Normal	1	-	-	-	-	-	-	6/2/2009 11:55:17 AM	CMDB	OVLABDT5 (Ma…	uCMDB A…	Sync Task		The uCMDB synchronization of the HP Operations Man
Warning	1	-	-	-	-	-	-	6/2/2009 11:55:17 AM	CMDB	OVLABDT5 (Ma…	uCMDB A…	Sync Task		Cannot find parent CI a1126a6dafc80aaa4845db89b2
Warning		-	-	-	-	-	-	6/2/2009 1:45:18 PM	CMDB	OVLABDT5 (Ma…	uCMDB A…	Sync Task		Cannot find parent CI a1126a6dafc80aaa4845db89b2
Normal		-	-	-	-	-	-	6/2/2009 1:45:18 PM	CMDB	OVLABDT5 (Ma…	uCMDB A…	Sync Task		The uCMDB synchronization of the HP Operations Man
Major		-	X	-	-	-	-	6/2/2009 1:46:51 PM	mars.planets.com	mars.planets.com	test	mars.pl…	Bethesda	Internal temperature exceeds safe threshold.

Like most aspects of DMA, the value of this new feature in SMM will depend on having useful information in the UCMDB to begin with. Any implementation of DMA should include a thorough review of how CIs are being discovered and instantiated in the UCMDB. For the CMA feature, pay special attention to which *attributes* are being populated. To get maximum benefit with DMA, it may be necessary to expand what goes "in" so that what comes "out" will produced the desired results.

For more information

Documentation for the following products referenced in this white paper can be found at the link below.

HP Operations Manager Dependency Mapping Automation
HP Operations Manager on UNIX
HP Operations Manager on Windows
HP Universal CMDB (Application Mapping)

http://h20230.www2.hp.com/selfsolve/manuals

Note: This site requires that you register for an HP Passport and sign in.

May 2009