HP Business Service Management

for the Windows/Linux operating system

Software Version: 9.13

Effective Modeling for BSM 9.x – Best Practices

Document Release Date: May 2012
Software Release Date: May 2012

Legal Notices

Warranty

The only warranties for HP products and services are set forth in the express warranty statements accompanying such products and services. Nothing herein should be construed as constituting an additional warranty. HP shall not be liable for technical or editorial errors or omissions contained herein. The information contained herein is subject to change without notice.

Restricted Rights Legend

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

Copyright Notices

Trademark Notices

Adobe® and Acrobat® are trademarks of Adobe Systems Incorporated.
Intel®, Pentium®, and Intel® Xeon® are trademarks of Intel Corporation in the U.S. and other countries.
Java is a registered trademark of Oracle and/or its affiliates.
Microsoft®, Windows®, Windows NT®, and Windows® XP are U.S registered trademarks of Microsoft Corporation.
Oracle is a registered trademark of Oracle Corporation and/or its affiliates.
UNIX® is a registered trademark of The Open Group.

Acknowledgements

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

Documentation Updates

The title page of this document contains the following identifying information:
Software Version number, which indicates the software version.
Document Release Date, which changes each time the document is updated.
Software Release Date, which indicates the release date of this version of the software.
To check for recent updates or to verify that you are using the most recent edition of a document, go to:
http://h20230.www2.hp.com/selfsolve/manuals
This site requires that you register for an HP Passport and sign-in. To register for an HP Passport ID, go to: http://h20229.www2.hp.com/passport-registration.html
Or click the **New users - please register** link on the HP Passport login page.
You will also receive updated or new editions if you subscribe to the appropriate product support service. Contact your HP sales representative for details.

HP Software Support

Visit the HP Software support web site at: http://www.hp.com/go/hpsoftwaresupport
The web site provides contact information and details about the products, services, and support that HP Software & Solutions offers. It provides customer self-solve capabilities, and is 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 web site to:

- Search for knowledge documents of interest
- Submit and track support cases and enhancement requests
- Download software patches
- Manage a support contract
- Look up HP support contracts
- Review information about available services
- Enter into discussions with other software customers
- Research and register for software training

To find more information about access levels, go to: http://h20230.www2.hp.com/new_access_levels.jsp

Table of Contents

1 Introduction

1.1 Scope and Motivation

The RTSM (Run Time Service Model) was introduced in BSM 9; it is the foundation of many of the BSM applications and services such as Service Health, OMi, SLM, Business Impact, TBEC and so on. In order to get maximum value out of each of the BSM applications, you need to have a good model representing the CIs in your IT environment and the relationships between them.

Information about the CIs and their relationships in RTSM can come from the following sources:

1) **CMS and DDMa.** If you have a CMDB in your IT environment (whether it is HP UCMDB or third-party), you can synchronize the service models from the CMDB into RTSM. These topologies may have been discovered by DDMa, or modeled manually by the CMDB team. (You can find detailed information on synchronizing service models from CMS into RTSM in the RTSM Best Practices white paper.) DDMa can be connected directly to RTSM to populate discovered CIs. Note that CMDB models are consumed by other BTO products such as Service Manager and Release Control.

2) **BSM data collectors and stand-alone products.** BSM data collectors can create CIs and relationships within RTSM based on monitored and/or discovered data. For example, when you define a CPU monitor in SiteScope, the remote server is a Computer CI; SiteScope creates the relevant topology in RTSM. Additional examples can be RUM creating a WebServer CI connected to a Computer based on network traffic; NNMi discovering L2 topology and enriching RTSM with it; BSM Connector for SCOM reporting topology as well as events; or Operations Manager creating service models based on its data using the TopoSync operation.

Each domain also provides different **out-of-the-box views** in order to consume the related CIs. For example, EUM has the *End User Monitors* view that shows you all the business applications and their end-user transactions; SiteScope provides the *System Hardware Monitoring* view to display all the nodes it monitors.

But what do you do when you want to see how your infrastructure affects your business? First, you need to model your business CIs correctly – you need to know which CIs your model contains, and the relationships between them. (Note that there is no significance to *which* source created each CI, as long as the CIs are in RTSM.) After you finish modeling, you can build your monitoring strategy to provide a comprehensive health view of your IT environment - to maximize the value of Service Health, OMi, SLM, SHA and more. This is when you will realize the value of RTSM.

Some data collectors can relate business CIs with their supporting infrastructure. For example, when RUM sniffs network traffic, it can find the nodes which handled requests that are part of an end-user business transaction. Similarly, Diagnostics can find which J2EE components support a business transaction based on Java calls. However, this ability is not consistent across all data collectors, and you need to bridge certain gaps in modeling.

In addition, there are some cases where a data collector cannot create the monitored CI. For example, in the case of a SiteScope Log monitor, the data monitored in the log file can be the health of several CI types (for example server, running software, application, and service). As a result, SiteScope will not create a monitored

CI for that monitor by default. In order to have a complete health overview of your model, you will want to relate this SiteScope data to a CI, and then relate that CI in your model.

1.2 What is a Model?

The concept of *modeling* was introduced with UCMDB 8, with the introduction of *Modeling Studio*. (Previously, instance views were used to create service maps or service views, to show business elements based on data from the CMDB.)

A *model* is basically a CI in CMDB. Out-of-the-box, only some CI types can act as a model, but this can be customized using the MODELING_ENABLED qualifier on the CI Type. A model is usually a logical element such as a business service or business application.

Modeling is the task of connecting a model CI to its subordinate entities, using relationships. There are two different types of relationships which are available during modeling:

a. **Containment.** Use this relationship when the subordinate entity is managed under the lifecycle of the model. These entities can be other models (for example business applications of a business service), or key components which are contained in the model (for example web servers or databases of a business application). You need to identify the elements which are *dedicated* for that model, and not shared by others.
b. **Usage.** Use this relationship when your model is using a shared resource and should be impacted by it, such as a shared DB server or LDAP authentication service.

These relationships are required to understand ownership, scope (for example the scope of a business application's downtime will include only the contained CIs and not the dependent ones), and so on.

When modeling an application, you should use Running Software CIs whenever possible, for two main reasons:

1) It is more accurate to map the model to the Running Software, because this is what the model is using from that node, and not anything else. For example, if you have an Oracle server running on a computer which is shared among multiple applications, you should use the DB Schema when you model a specific business application, because that is the only part it is using from that Oracle server and from that computer.
2) Regarding Impact calculation, the Running Software is impacted by the node and not vice-versa. This is needed for correct status calculation and propagation in BSM. (See details in Appendix A.)

Once you set up your models, you can create *views* based on your models. Different users will consume your models in different ways; for example a DBA is interested in database and storage elements, whereas a network administrator needs to know to which network devices the model is connected. These views of your model can be achieved by using *perspectives*. The CMDB will generate these views based on the relationships in CMDB, and according to the perspectives you used in your view. Perspectives are queries that use the model as their input. (A perspective is like using different sunglasses to see the model in different ways.)

2 How to Create a Model in Modeling Studio

There are two ways to create models: instance-based modeling, and pattern-based modeling.

In an **instance-based model** you need to manually find and add the CIs which are contained in your model. This approach is static; when something changes in your environment you need to update your model. This approach is typically used for high-level models such as business services since they are more static by nature. (Note that there is a semi-automatic way to receive updates when a new CI answers a pattern and can be added to the model, using *Reveal Paths and Watch Points*. For details, refer to the Modeling documentation.)

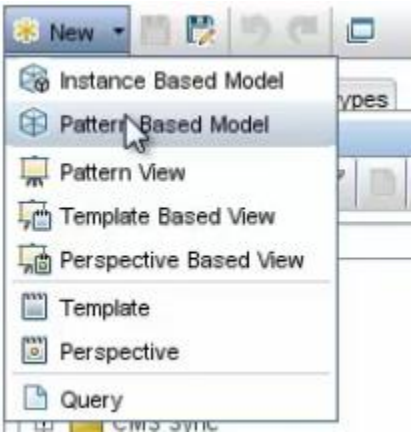
In a **pattern-based model** you need to create a query (TQL) that defines how to find the CIs which should be part of your model. This approach is useful when you can identify the dedicated CIs (for example by name or IP). When there are changes in your environment, for example if you add a web server or add a DB Instance for database high availability, your model is automatically updated.

Note that in a pattern-based model you can only create one level of modeling, whereas in an instance-based model you can create a complete top-down topology for business CIs (for example business function → business service → business application → CI collection → infrastructure).

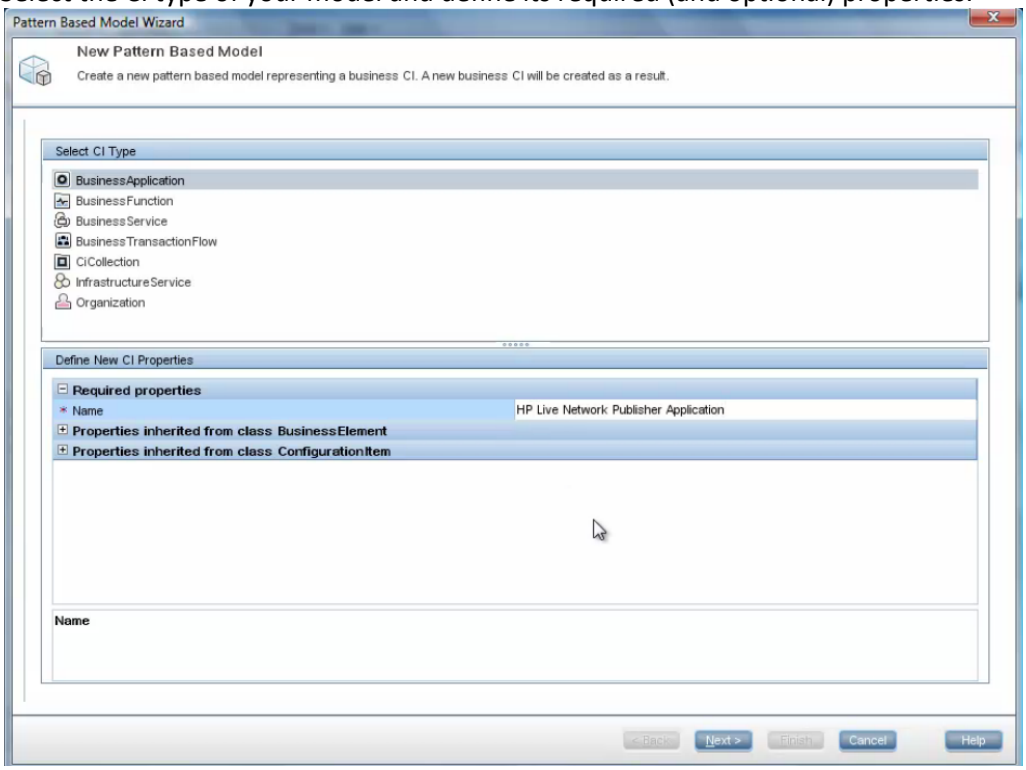
The following section provides instructions for both of these approaches.

<u>To create a pattern-based model:</u>

1) In **Admin > RTSM Administration > Modeling**, select **Modeling Studio**.
2) Select **New > Patten Based Model.**

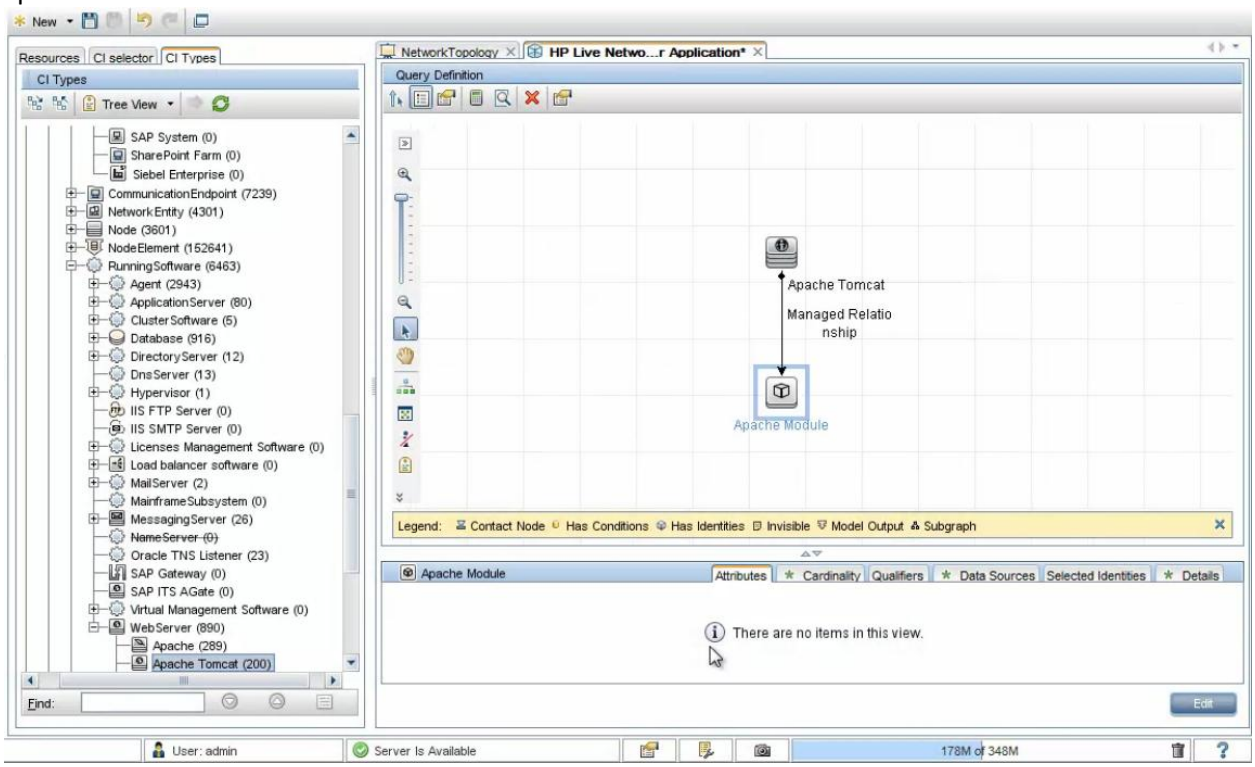


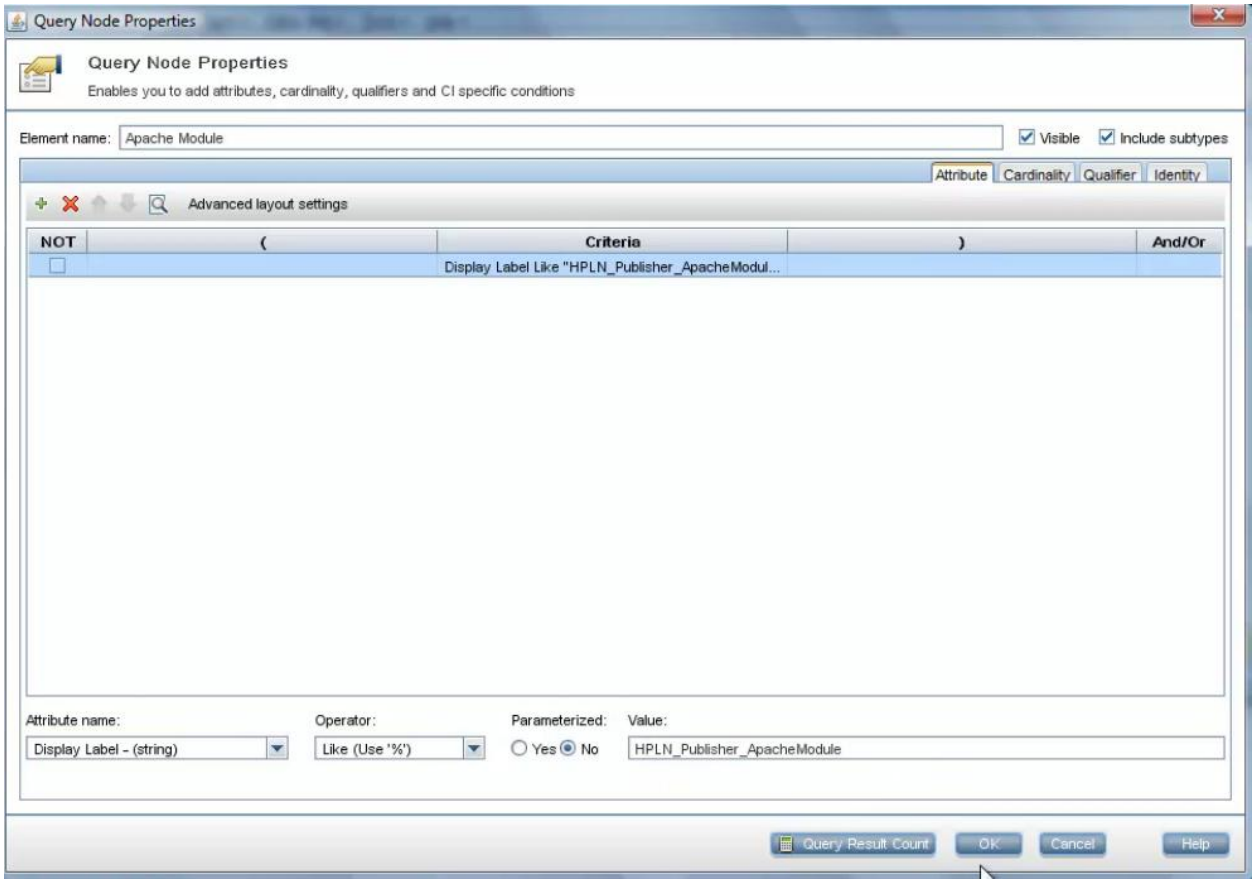
3) Select the CI type of your model and define its required (and optional) properties.



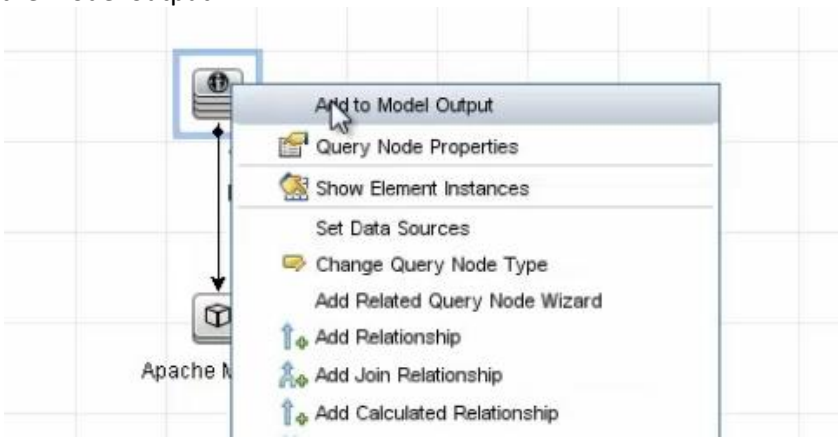
4) Click **Next**. In the next page select **Create new query**, and click **Finish**.

5) Build the TQL which will find the model's dedicated resource. In this example we will look for an Apache Tomcat that has a module with a certain name.





6) When you finished building the query, assign the dedicated resource to your model. Right-click the query node, and choose **Add to Model Output**. In this example, the Apache Tomcat will be added to the model output.



The query nodes which are marked with **Add to Model Output** will be connected to your model via **Containment** relationships.

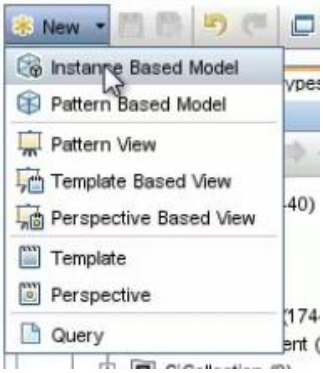
7) Save your pattern-based model.

Note:

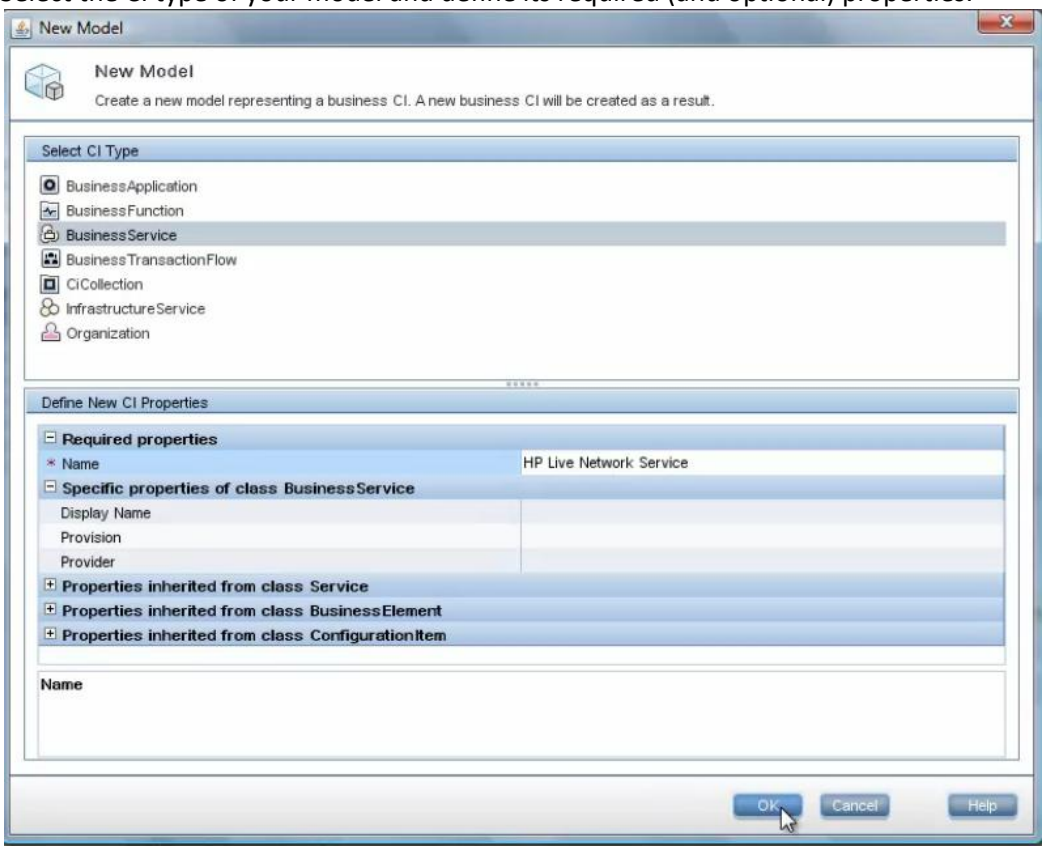
- You can have multiple query nodes in your query as part of the model output, so that (for example) you can add a database node to this TQL, with its own conditions, and include it in the model output. This reduces the number of TQLs, and makes it easier to administer your model.
- The model is updated by the query results at 12:00 AM and 12:00 PM. From CMDB 9.05 this will be configurable; at this point it is not clear in which BSM version this will be available.

To create an instance-based model:

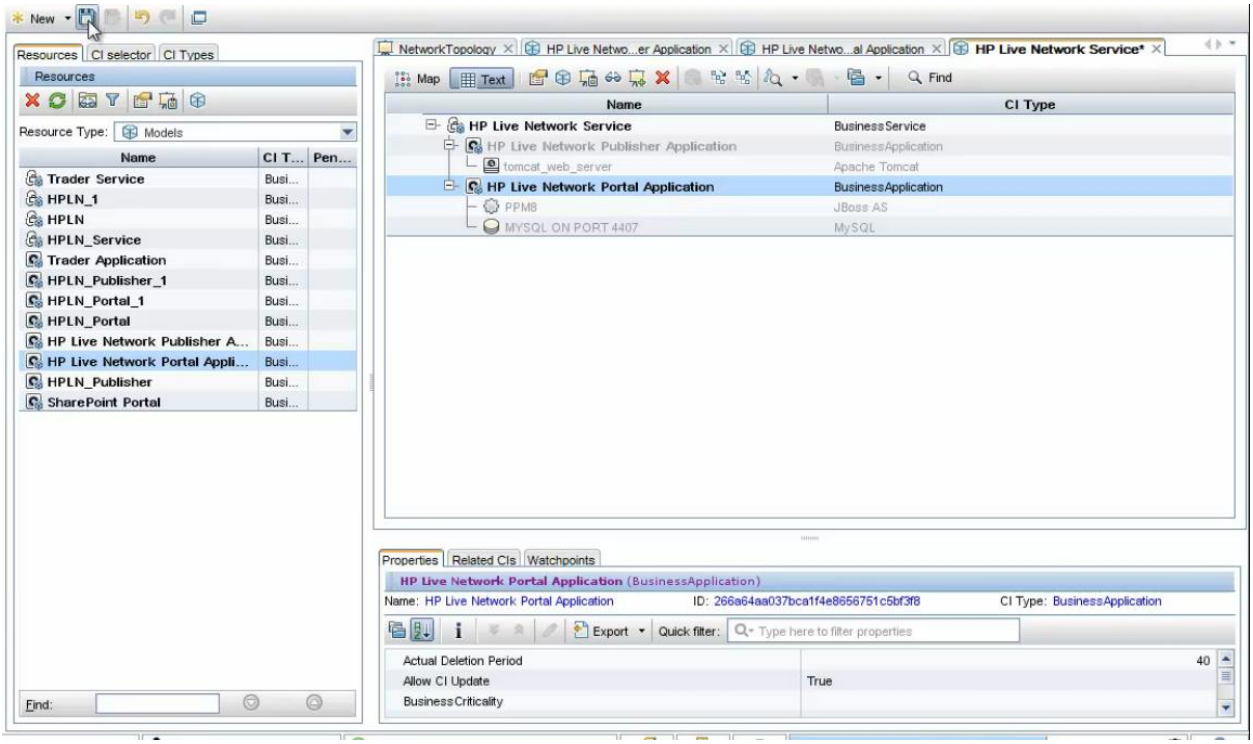
1) In **Admin > RTSM Administration > Modeling**, select **Modeling Studio**.
2) Select **New > Instance Based Model**.



3) Select the CI type of your model and define its required (and optional) properties.



4) Find the instances that you want to add to your new model. You can search for CIs in CMDB by browsing views, by CI type, or by resource.
In this case we will add two business applications. Simply drag-and-drop the instances under your model CI.



5) Click **Save.**

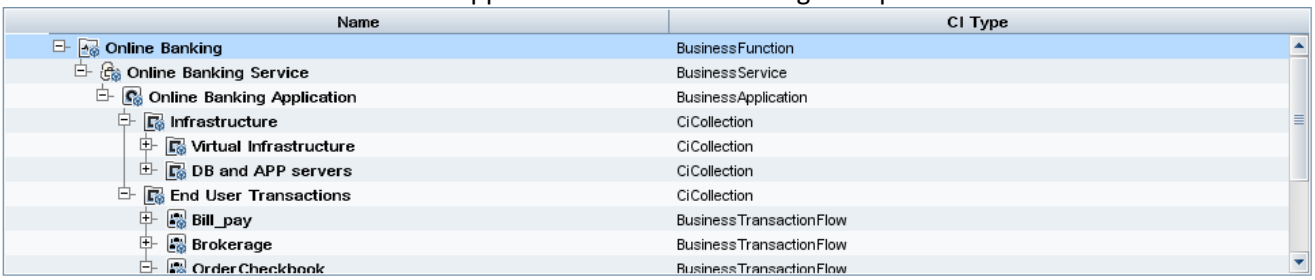
Note:

- You can insert new models or CI collections during instance-based modeling; changes in UCMDB are only submitted when you save your model.
- If your model depends on another resource (for example if a business application depends on an LDAP service for authentication, but this service is used by other applications as well), you can model this dependency by adding the CI to the **Related CIs** tab in the lower pane. This enables proper status propagation from the impacting CI to the dependant CI; for example if the LDAP service is down, the application's system availability is impaired.
- If you modify the query of your pattern-based model your model will be updated according to its scheduling definition. However, you cannot connect such a model to other CIs manually since the model is not instance-based but pattern-based. If you do so, the model will stop being pattern-based and will become instance-based.

3 Modeling Best Practices

The following examples of models illustrate modeling best practices:

1) **Top-down approach.** This example shows a business unit from a top-down perspective; the business function contains various services and applications which are managed as part of this function.



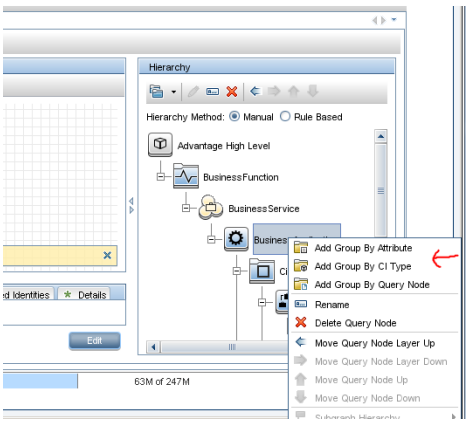
Note: If a CI name appears in **Bold** this means it is a model.

2) **Use of CI collections.** In the above example there are two CI collections below the business application. This structure is common when an application has end user monitoring; one CI collection is used to aggregate the *End User Transactions,* and another CI collection is used to aggregate the *Infrastructure* elements.

   Note that if you are working with BTM (Business Transaction Monitoring) such as Diagnostics or TransactionVision, these data collectors connect the infrastructure elements directly below the business transactions. You may therefore need to model dedicated resources which were not discovered in the *Infrastructure* CI collection; you do not need to add the infrastructure elements also in that CI Collection, since it will impact the model twice.

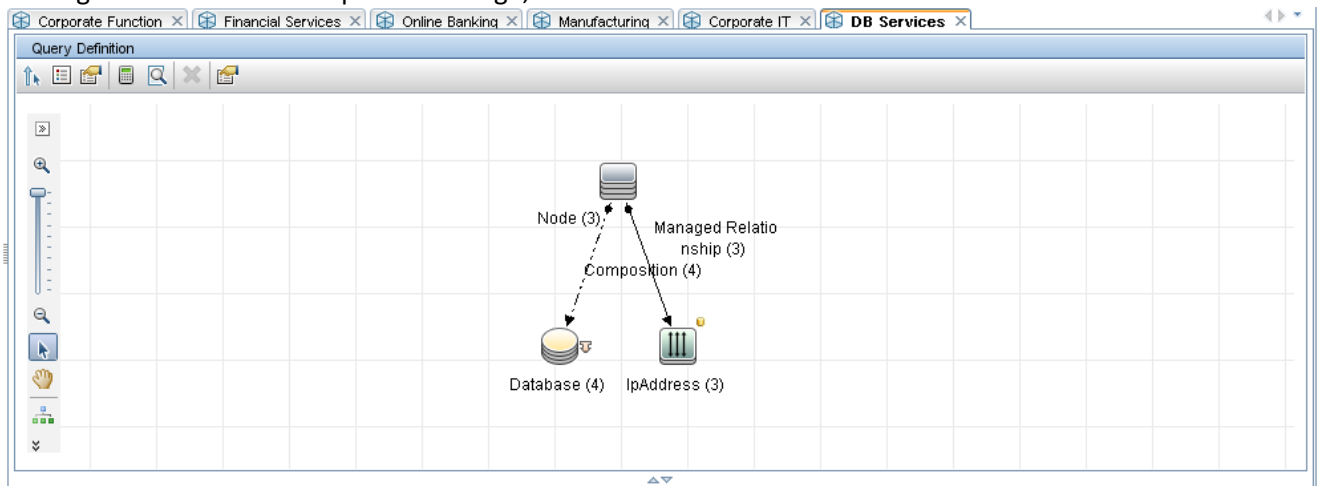
   There are two main reasons to use CI collections:

   a. **Grouping.** Grouping entities in different CI collections makes it easier to understand the model, and can be useful for logical grouping as well (databases, web servers, critical applications, and so on). Note that you can create *visualization* groupings in the view definition (see image below); this is different from using a CI Collection which is a CI. Grouping is automatic according to a rule, while CI Collections are hard-coded CI instances created and defined by the user with static content or pattern.



b. **KPI Calculation.** If you want to have a dedicated KPI calculated on a group of CIs, you need to use a CI collection; visualization grouping will not enable this. For example, you can model an application across multiple data centers, or perform Percentage Rule calculations on multiple databases.

3) **Horizontal approach.** In cases of shared services (for example mail service, storage service, authentication application, data center and so on) you can use a pattern-based model to identify all the resources of those services. In the following example we connect all the databases which are running on servers within a specific IP range, to an infrastructure service called DB Services:



4) **Models to match specific BSM requirements.** Some applications in BSM have strict model requirements. For example, some TBEC rules assume a specific topology which is usually created by BSM data collectors or DDMa; if you model your business application manually you should use the topology expected by the rule (or adjust the TBEC rules after creating the new model). Other examples are SHR and SHO which assume a direct relationship from the business service to the infrastructure (VM or node). This is not aligned with the modeling best practice described above, but for these use cases you should maintain such relationships in your model (although you do not have to display them in your views).

5) **Multiple environments.** In many cases you will have the same business entity (for example business application) in multiple environments - such as testing, staging, and production. The best practice for modeling this is as follows:

a. Model each environment instance separately; use different CIs for testing, staging, and production environments. This allows you to manage different SLAs for each service, and simplifies Impact analysis.

b. We recommended that you use different values for the **Business Criticality** attribute among different environments, in order to distinguish between production issues (high importance) and testing issues (low importance).

c. Use a CI collection in order to group all instances of model-X from its different environments. For example, you can create a CI collection called *MyApplication* and connect it to the CIs

MyApplication_Production and *MyApplication_Testing*, in order to get the overall status of MyApplication from different environments.

d. If there is a requirement to record common facts for a specific type of environment, create a sub-class for CI collection and define the necessary attributes in that sub-class.

Note: In a future release of UCMDB we will add a new attribute to Business Element CIs which will be called *environment*. This attribute will be of type "string-list" (multi-value attribute) in order to express the same CI in multiple environments.

6) **Where should you perform modeling?**

- If you have both BSM and a CMS, we recommended you perform modeling in the CMS because it contains all the relevant CIs you need to perform modeling.
- If RTSM contains additional topology information (for example, if you are using RUM or Diagnostics which provide topology from business elements to infrastructure), first synchronize this information from RTSM into CMS (like from CMS to RTSM but in the opposite direction), then perform modeling in CMS, and then synchronize all your models back into RTSM.

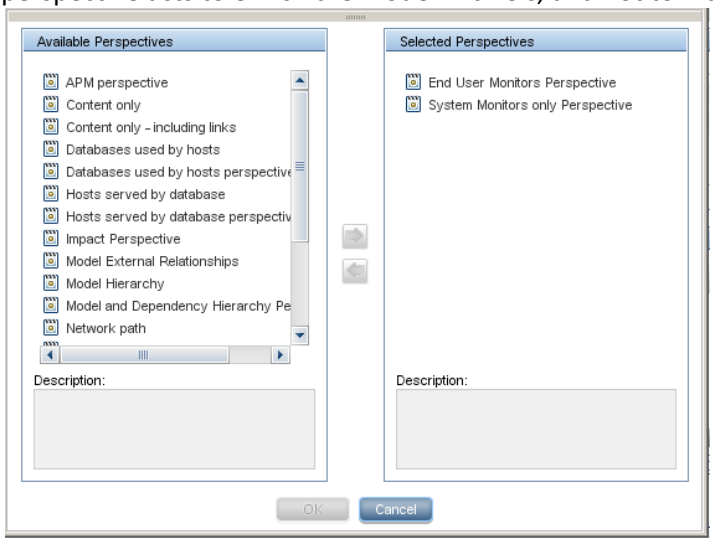
4 How to Use the Model

One of the main usages of models is to create views. The easiest way to create a view from a model is using a perspective-based view.

A **perspective** is very similar to a pattern view; it has a query which defines a pattern of CIs to search in the CMDB, and a hierarchy which defines how to fold CIs from the result into different layers. The difference between a perspective and a pattern view is the root contact for the query; unlike a pattern view, the perspective query starts from one or more specific CIs, which you define in the content section of the perspective-based view.

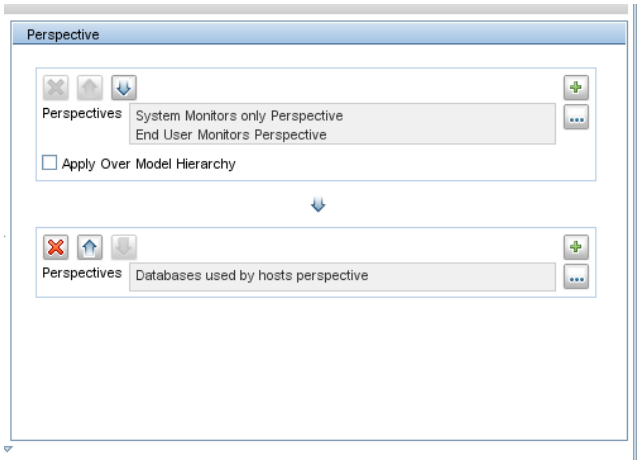
In a **perspective-based view** you can use multiple perspectives in 2 ways:

1) **Union.** You can add one or more perspectives, and then only those CIs which are part of the results of all perspectives are part of the view result. Note that the model is always part of the view - the perspective acts to enrich the model with CIs, and not to filter it.



In the above example, only those CIs which are monitored by EUM (BPM or RUM) or System (SiteScope or OM) will be part of the result of the view.

2) **Concatenation.** After applying one or more perspectives, you can add additional perspectives which will be invoked on top of the results of the previous perspectives.



In the above example, we add to the view result all the databases used by hosts which are monitored by EUM or System.

Perspective-based views allow you to choose what information related to your model you want to see. The **Apply Over Model Hierarchy** checkbox determine whether the perspectives will be applied to the CIs included in the model as well as to the model itself, or to the model itself only. The view which you create can then be used anywhere in BSM (for example in Service Health, in SLA creation, and so on).

Since perspective-based views are based on models which are dynamic, and on perspectives based on queries that are dynamic, the entire content of the view is based on dynamic structure, and is updated according to changes to the models or to related topology.

BSM provides several out-of-the-box perspectives based on monitor topology, such as the *End User* or *System Monitors Perspective*. The *Impact Perspective* is useful when creating views which will be used in Service Health; this perspective returns all the CIs which have Impacted_by relationships, and folds them according to these relationships. This perspective is very useful in Service Health since the view represents the same topology in which KPIs are propagated and calculated. (For details on the Impact layer, see Appendix A.)

5 What happens when the same CI comes from different sources?

Since RTSM gets CIs from multiple sources, what happens if the same CI is created from two sources – how are they reconciled into one CI? This is handled by the UCMDB **Reconciliation Service**. Each CI type has Identification/Reconciliation rules which define what makes a CI unique. Identification rules can be simple (for example by key attributes), or complex (for example by a combination of attributes and relationships to other CIs).

Identification rules can be seen in the **RTSM Administration > CI Type Manager > Details** tab. Although it is possible to modify Identification rules, we do not recommend doing so.

Reconciliation is ongoing; when a CI is inserted into UCMDB – whether it is added manually, or via synch from CMS, discovery, enrichment, or by a data collector – the CI goes through this reconciliation process.

6 What happens when there is no clear monitored CI?

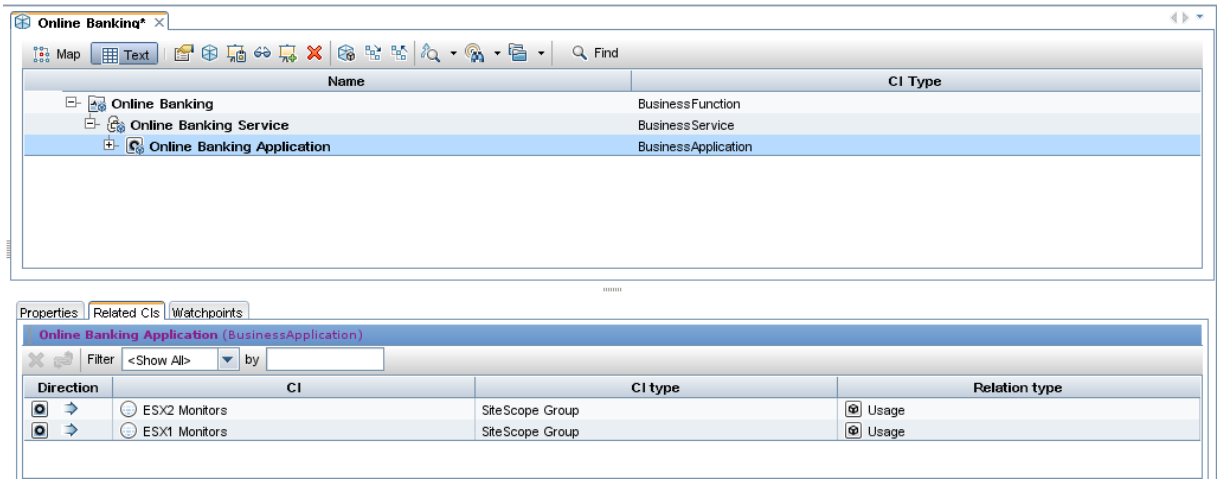
We have seen the importance of modeling and reusing the same CIs from RTSM across different data collectors; for example an application which is monitored by RUM and Diagnostics, or a server which is monitored by SiteScope and discovered by DDMa - this is where you realize the value of RTSM.

But what happens if the data collector cannot identify the monitored CI? For example, in the case of a SiteScope Log file monitor, SiteScope doesn't know by default what information is monitored in the log file. The log file can contain information on the server itself (for example free disk space); it can contain information on software running on the server (for example JBoss status); it can even contain information on logical CIs which are impacted by this server (for example BSM application services status).

You can configure the monitored CI in order to reconcile it with the topology you have in RTSM. Perform the following:

1) In the SiteScope monitor properties, open the **HP Integration Settings** panel.
2) Select the **Report monitor and related CI topology** checkbox.
3) Select the CI type that you want to report.
4) Enter the required attributes to identify the CI; make sure you fill in the correct values. For example, for a business application, the Organization Type values should match the Organization CI type's applicable values. For Running Software, the Server value is the node short name.

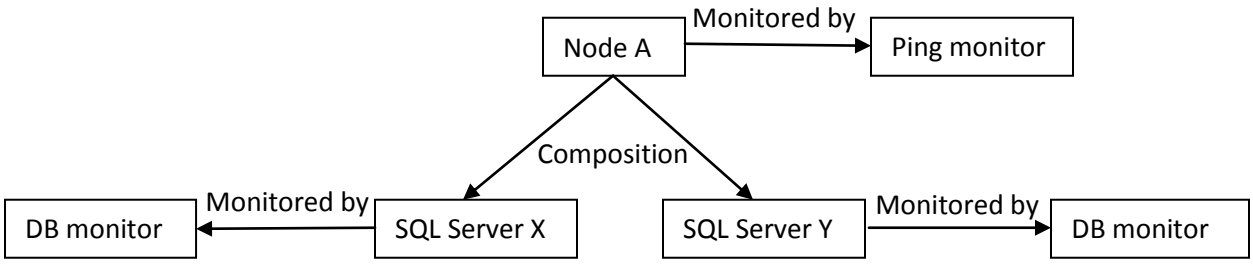
Note: Although it is possible to model the SiteScope Groups and SiteScope Monitors in your models, we recommend not to do so. This is because these CI types are only part of the BSM Class Model, and cannot be synchronized back to CMS – they will not reflect the real topology of your model. Try to use monitored CIs as much as possible; if you cannot use a monitored CI, use the SiteScope topology as **Related CIs** for your model:



Note that for EMS there is a different way to report topology; refer to the SiteScope 11.11 documentation update which explains in detail how to report topology for Technology Integration monitors.

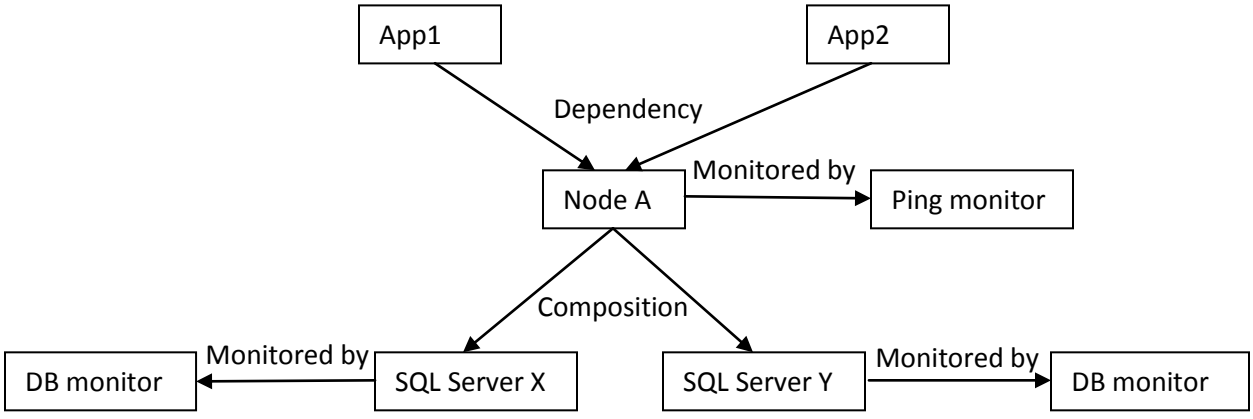
Appendix A – The Impact Layer

Before BAC 8, statuses were calculated on top of the topology in CMDB. This concept had certain limitations, such as in the following example: Node A hosts SQL servers X and Y, which are monitored by SiteScope. The node is also monitored by SiteScope for its ping availability. This is the topology created by SiteScope:



The usage of a **Composition** link between the node and SQL server is very important; this link means that if the node is deleted, the SQL server is deleted as well (recursive deletion). This behavior is not unique to the Composition link; it is defined by the RECURSIVE_DELETE qualifier. This link also means that the Running Software CI is identified by the node which hosts it, meaning you cannot move the SQL server to a different parent CI (in other words, it will be a different CI with different ID).

Suppose there are two applications which are using these SQL servers, but each application is using only one database. In order to achieve statuses propagation from both the ping monitor and the DB monitors, users needed to relate the applications this way:



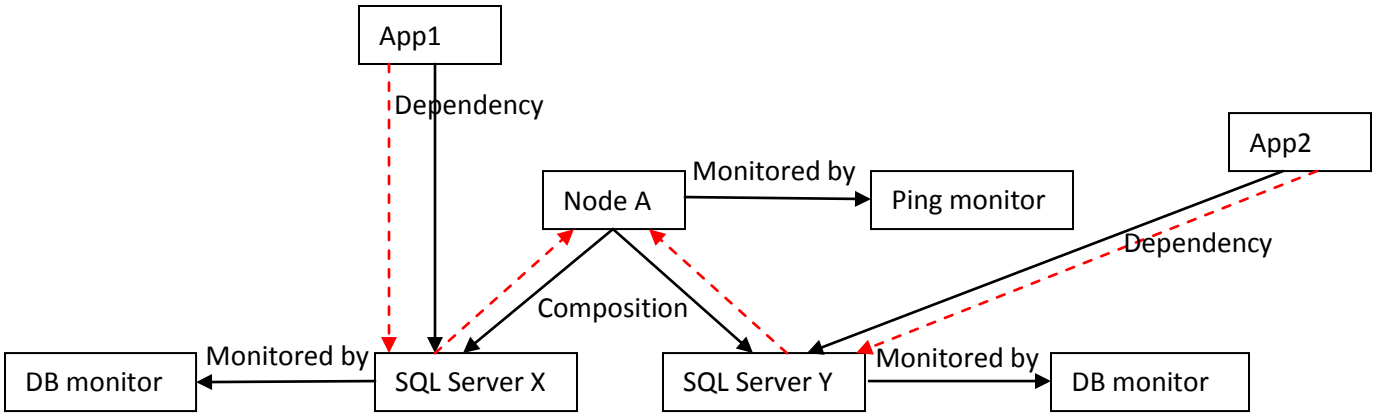
The problem with this model is that both DB monitors propagate to both applications, even though each application is only using one DB and should be impacted only from that one.

To solve this problem, UCMDB introduced the **Calculated Relationships** concept. A calculated relationship is a link which you can query from UCMDB in your TQLs, but it is not stored in the UCMDB (unlike CIs and relationships). A calculated relationship is defined by a triplet: a source CIT, a target CIT, and the link type

between them. The calculated relationship can have the same direction as the original link, or the opposite direction.

The **Impact Layer** is represented by two calculated relationships: **Impacted By (directly)** and **Impacted By (potentially)**, which both extend the **Impacted By** calculated relationship. UCMDB and BSM provide default triplets for these links which can be customized.

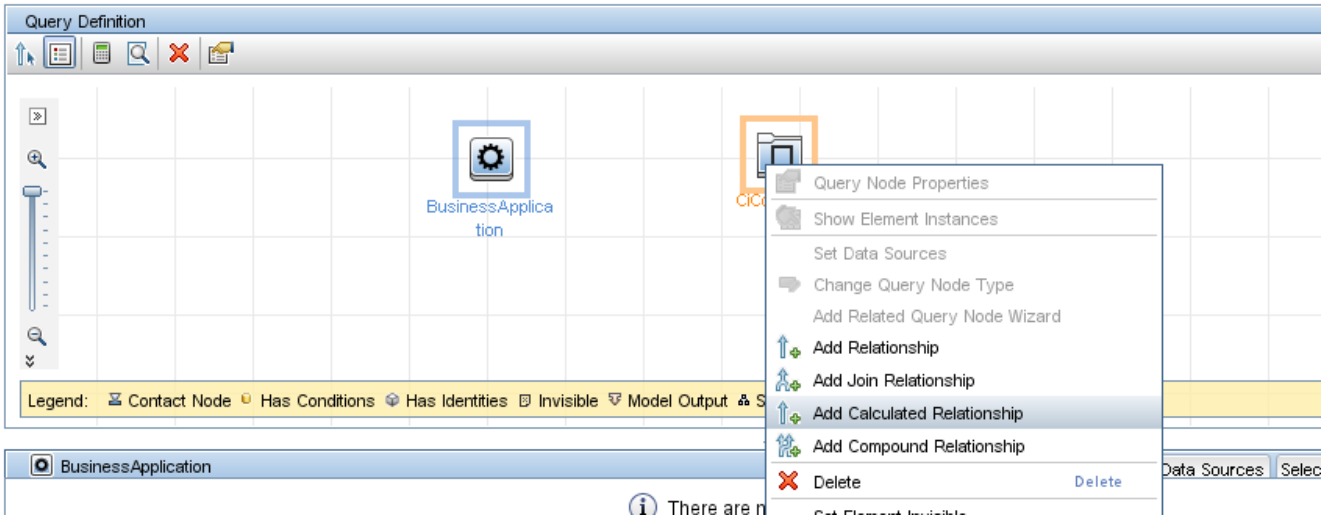
Let's look again at our example. This is the new topology with **Impacted By** links (marked in red):



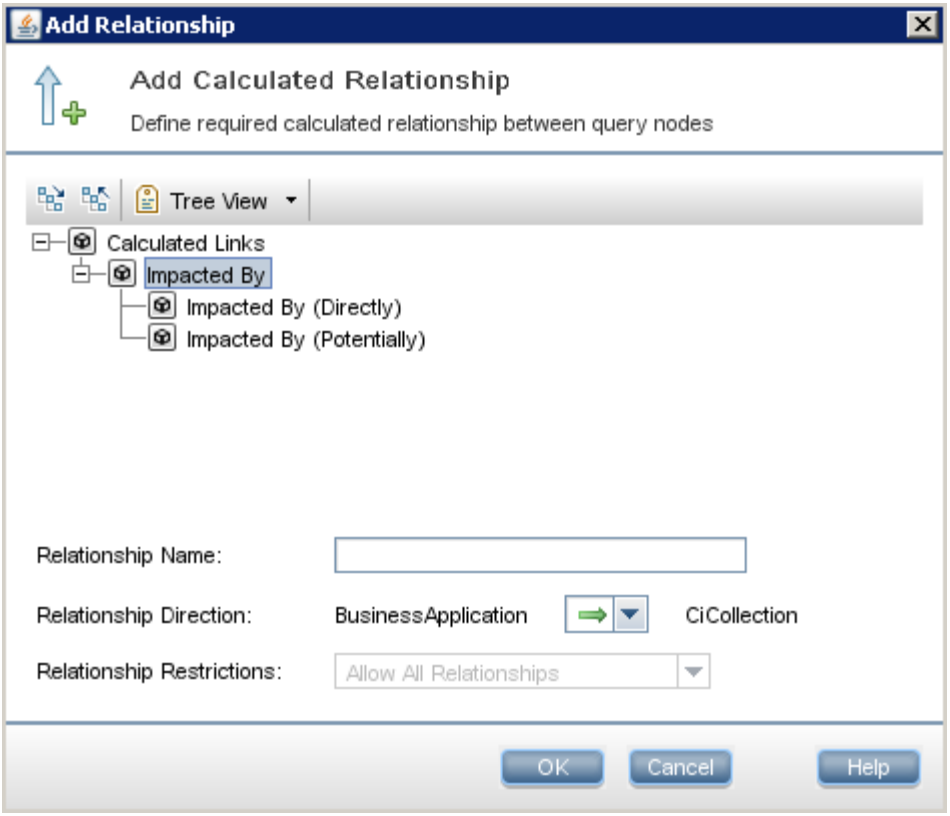
Note that there is no Impacted By link between monitored CIs and Monitor CIs, since in BSM 9 the health indicators (His) are assigned to the monitored CIs directly. (In BAC 8 there was an impact link from the monitored CI to the Monitor.)

We recommend that you use the out-of-the-box class model, but in some cases you might want to create new CI types or relationships in order to model your IT environment. In such cases we recommend that you extend the default class model CI types, and not create new branches. The reason for this is that your new CI types will inherit the Impacted_by triplets and will take part in the Impact layer, whereas if you create new branches you will need to define new triplets so that KPIs will be propagated and calculated on your new CI types.

To check whether there Is an *Impacted By* relationship between two CIs, create a TQL, add the two elements into the Query Definition, and add a calculated relationship between them:



Select **Impacted By**, and verify that the direction of the relationship is as expected.



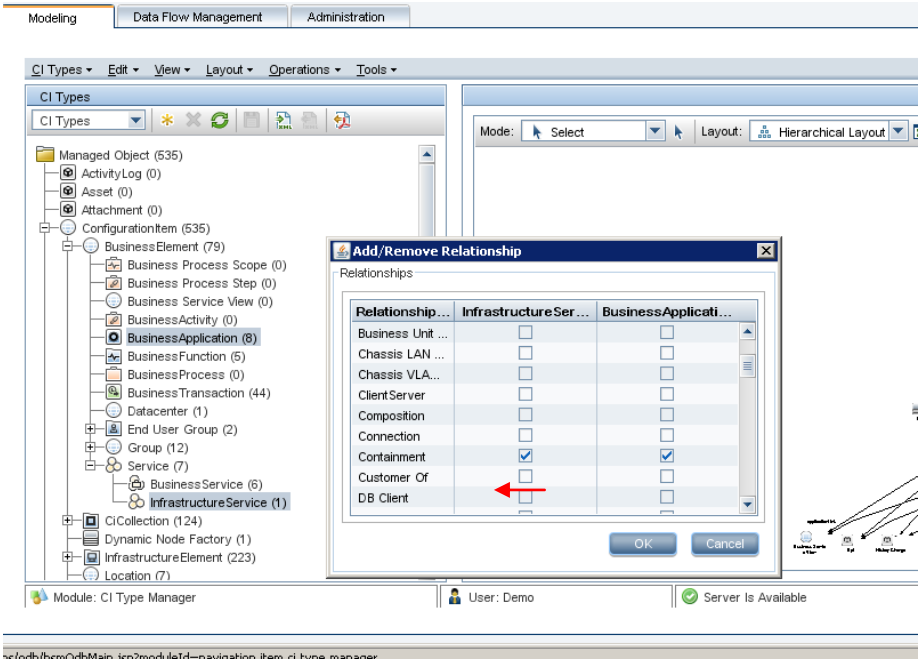
Appendix B – The Infrastructure Service Challenge

Suppose you want to monitor an LDAP Service using VuGen, and this LDAP Service is used by multiple services and applications. According to the BDM Class Model, the correct CI type to model this LDAP Service is **Infrastructure Service**.

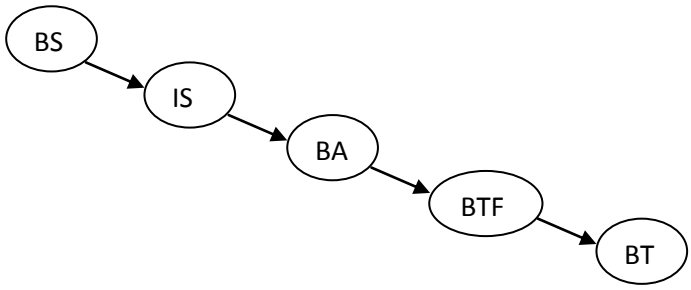
In BSM, in order to have EUM monitoring (BPM or RUM) you must use the business application CI type. In our use case, we want to use BPM. However, according to the default BDM Class Model, a **business application** cannot be connected below the infrastructure service CI type, in order for statuses to propagate bottom-up in BSM.

You can use one of the following options to deal with this:

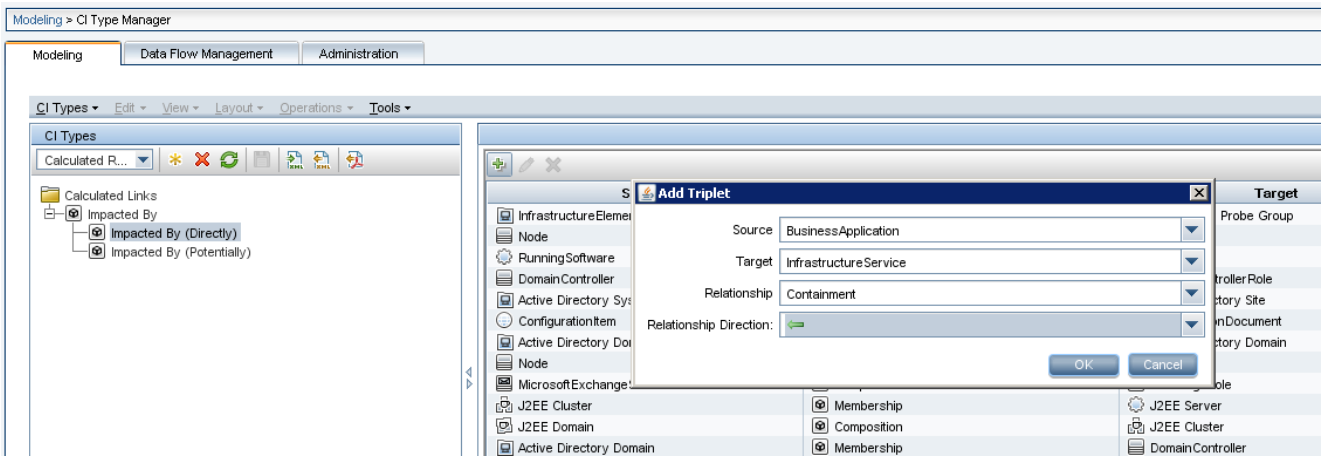
1. Add a valid link of type **Containment** from *Infrastructure Service* to *Business Application*; do this both in CMS and BSM. This allows you to model your business application below the infrastructure service.



Your model will look like this:

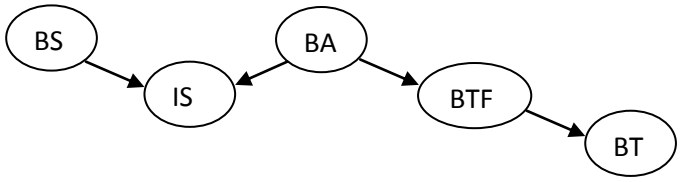


2. Add a triplet for **Impacted_by (directly)** as follows:
Source: *Business Application;* Link: *Containment;* Target: *Infrastructure Service;* Direction: *opposite*.

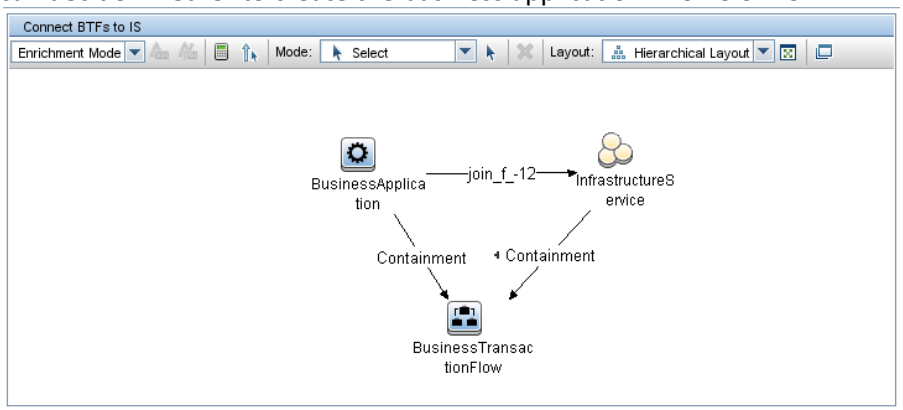


This triplet definition allows you to model the infrastructure service below the business application, but KPIs will propagate from the business application to the infrastructure service. You must do this in BSM; we recommend also doing this in CMS for aligned Impact analysis.

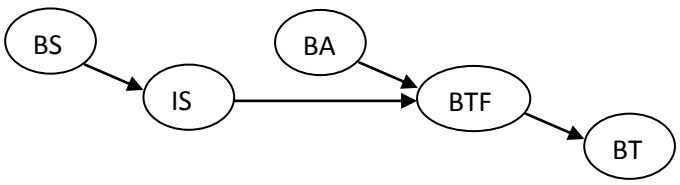
Your model will look like this (although from the Impact perspective it will look like the first option):



3. In BSM define an **Enrichment rule** which looks for all the business applications with the same name as an infrastructure service, and then connects the business transaction flows to the infrastructure service. You can decide whether to create the business application in CMS or BSM.



Your model will ultimately look like this (assuming there is a business application with the same name as the infrastructure service):



Appendix C – Additional Resources

The following additional resources can help you with Modeling:

1) BSM Documentation: There are several PDFs that can be helpful such as: bdm_business_model, the Modeling Guide, and UCMDBClassModel.

2) This is a link to a short movie demonstrating modeling: https://docs.google.com/leaf?id=0B8aDjf21B4iZY2Q3MDJhY2EtMDYyYi00ODM2LThjMzUtNDk4YWRmYWM5NGFl&hl=en_US&authkey=CMzx3skH