

Service Modeling Guide

Operations Center 5.5

November 6, 2013

www.netiq.com/documentation

Legal Notices

THIS DOCUMENT AND THE SOFTWARE DESCRIBED IN THIS DOCUMENT ARE FURNISHED UNDER AND ARE SUBJECT TO THE TERMS OF A LICENSE AGREEMENT OR A NON-DISCLOSURE AGREEMENT. EXCEPT AS EXPRESSLY SET FORTH IN SUCH LICENSE AGREEMENT OR NON-DISCLOSURE AGREEMENT, NETIQ CORPORATION PROVIDES THIS DOCUMENT AND THE SOFTWARE DESCRIBED IN THIS DOCUMENT "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. SOME STATES DO NOT ALLOW DISCLAIMERS OF EXPRESS OR IMPLIED WARRANTIES IN CERTAIN TRANSACTIONS; THEREFORE, THIS STATEMENT MAY NOT APPLY TO YOU.

For purposes of clarity, any module, adapter or other similar material ("Module") is licensed under the terms and conditions of the End User License Agreement for the applicable version of the NetIQ product or software to which it relates or interoperates with, and by accessing, copying or using a Module you agree to be bound by such terms. If you do not agree to the terms of the End User License Agreement you are not authorized to use, access or copy a Module and you must destroy all copies of the Module and contact NetIQ for further instructions.

This document and the software described in this document may not be lent, sold, or given away without the prior written permission of NetIQ Corporation, except as otherwise permitted by law. Except as expressly set forth in such license agreement or non-disclosure agreement, no part of this document or the software described in this document may be reproduced, stored in a retrieval system, or transmitted in any form or by any means, electronic, mechanical, or otherwise, without the prior written consent of NetIQ Corporation. Some companies, names, and data in this document are used for illustration purposes and may not represent real companies, individuals, or data.

This document could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein. These changes may be incorporated in new editions of this document. NetIQ Corporation may make improvements in or changes to the software described in this document at any time.

U.S. Government Restricted Rights: If the software and documentation are being acquired by or on behalf of the U.S. Government or by a U.S. Government prime contractor or subcontractor (at any tier), in accordance with 48 C.F.R. 227.7202-4 (for Department of Defense (DOD) acquisitions) and 48 C.F.R. 2.101 and 12.212 (for non-DOD acquisitions), the government's rights in the software and documentation, including its rights to use, modify, reproduce, release, perform, display or disclose the software or documentation, will be subject in all respects to the commercial license rights and restrictions provided in the license agreement.

© 2014 NetIQ Corporation. All Rights Reserved.

For information about NetIQ trademarks, see https://www.netiq.com/company/legal/ (https://www.netiq.com/company/legal/).

All third-party trademarks are the property of their respective owners.

Contents

About This Guide 7

1 Introduction to Modeling 9

Part I Creating Service Models in the Operations Center Console 11

2 Building Service Models 13

2.1 Understanding Service Models . . . 13
2.2 Understanding Service Models Components . . . 14
2.3 Creating Service Models . . . 16
2.3.1 Creating a Service Model . . . 17
2.3.2 Adding Child Elements to an Existing Service Model . . . 19
2.4 Service Model Properties . . . 19
2.4.1 Specifying the Algorithm to Determine an Element's Condition . . . 20
2.5 Assigning Access Privileges to Service Models . . . 21
2.6 Modifying Standard Properties for Service Models . . . 21
2.7 Deleting Service Models . . . 23

3 Populating Service Models 25

3.1 Assigning Elements by Matching Properties . . . 26
3.1.1 Assigning Elements to a Service Model . . . 26
3.1.2 Matching by Specific Elements . . . 27
3.1.3 Matching by Expression . . . 28
3.1.4 Matching by Script Expression . . . 29
3.1.5 Matching by Class . . . 31
3.2 Stopping State Contributions or Alarm Propagation from Source Elements . . . 32
3.2.1 Stopping a State Contribution or Alarm Propagation . . . 32
3.2.2 Hiding or Displaying Nonsource Contributing Link Icons . . . 33
3.3 Analyzing Dependencies . . . 33
3.4 Copying, Moving and Linking Elements . . . 34
3.5 Renaming Service Model Elements . . . 34
3.6 Deleting Elements and Linked Relationships . . . 35
3.6.1 Removing a Native Element . . . 35
3.6.2 Removing a Linked Element . . . 35
3.6.3 Removing All Occurrences of an Element . . . 36

4 Defining Classes, Behavior Models, and Property Pages 37

4.1 About Custom Classes, Behavior Models, and Property Pages . . . 38
4.1.1 Metamodel Icons . . . 39
4.1.2 Metamodel Buttons . . . 40
4.1.3 Maintaining the Metamodel Hierarchy . . . 40
4.2 Creating and Deleting Property Pages and Properties . . . 41
4.2.1 Creating a Property Page . . . 42
4.2.2 Deleting Property Pages . . . 43
4.2.3 Defining Properties for a Property Page . . . 43
4.2.4 Working with a Multiple Choice Property . . . 52

4.2.5 Working with HTML Pages or Text . . . 58
4.2.6 Configuring Macro Expressions . . . 61
4.3 Creating and Deleting Behavior Models . . . 65
4.3.1 Creating a Behavior Model . . . 65
4.3.2 Creating a Matching Rule . . . 65
4.3.3 Deleting Behavior Models . . . 70
4.4 Creating and Deleting Classes . . . 70
4.4.1 Creating a Class . . . 71
4.4.2 Deleting a Class . . . 72
4.4.3 Defining Class Relationship Templates . . . 72
4.4.4 Use Case Examples . . . 75
4.5 Editing Classes, Behavior Models, Property Pages, and Properties . . . 81
4.5.1 Editing a Class . . . 82
4.5.2 Editing a Behavior Model . . . 83
4.5.3 Editing a Property Page . . . 84
4.5.4 Editing a Property . . . 84
4.6 Security on Metamodel Elements . . . 85
4.7 Creating Custom Tooltips . . . 85
4.7.1 Creating a Custom Tooltip . . . 85
4.7.2 Removing Element Class from Tooltips . . . 87
4.8 Using Classes and Properties to Enhance Finding Elements . . . 88

5 Adding Custom Properties to Service Elements 89

5.1 Editing Custom Properties . . . 89
5.2 Creating and Viewing a Custom Property Page . . . 90
5.2.1 Creating a Custom Property Page . . . 92
5.2.2 Viewing Custom Property Pages . . . 94

6 Understanding Element Relationships 95

6.1 Visualizing Relationships Using the Relationship Browser . . . 96
6.2 Navigating the Layout and Exploring Element Hierarchies . . . 97
6.3 Rendering and Exploring Relationships . . . 98
6.3.1 Understanding Relationships . . . 99
6.3.2 Viewing Additional Relationships . . . 100
6.3.3 Exploring Dependency Relationships . . . 101
6.4 Relationship Diagram Display Options . . . 101
6.4.1 Selecting Element Node Styles . . . 102
6.4.2 Relationship Display Options . . . 102
6.5 Relationship Browser Configuration Presets . . . 103
6.5.1 Updating the Relationship Browser Configuration for the Current Element . . . 103
6.5.2 Working with Global and User Presets . . . 105
6.5.3 Creating and Deleting Presets . . . 108
6.6 Using Layout Rendering Features . . . 109
6.6.1 Understanding the Different Layout Styles . . . 110
6.6.2 Using a Different Layout Style in the Relationship browser . . . 116
6.6.3 Saving the Current Layout as an Element Preset . . . 117
6.7 Adding Elements . . . 117
6.7.1 Adding a Child Element to an Existing Element . . . 117
6.7.2 Adding Existing Elements to a New Element and Defining the Dependency Relationships . . . 118
6.7.3 Adding One or More Elements to an Existing Services Element . . . 120

Part II Using the Service Configuration Manager 121

7 Introduction to the Service Configuration Manager (SCM) 123

7.1 What is Service Configuration Management? . . . 123
7.2 Key SCM Features . . . 124
7.3 SCM Methodology . . . 124

8 Creating a Service Configuration 125

8.1 Understanding Multi-Definition and Multi-Generational Configurations . . . 125
8.1.1 Understanding Multiple Configurations . . . 126
8.1.2 Understanding the Generational Models Workspace . . . 126
8.2 Understanding Element Correlation Methods . . . 126
8.2.1 Understanding Rule-Based Correlation . . . 127
8.2.2 Understanding Class-Based Correlation . . . 127
8.3 Creating Service Configuration Definitions . . . 128
8.3.1 STEP 1: Create the Service Configuration Definition . . . 129
8.3.2 STEP 2: Define Structures . . . 131
8.3.3 STEP 3: Define Sources . . . 132
8.3.4 STEP 4: Define Relationship Dependencies . . . 138
8.3.5 STEP 5: Select Generation Options . . . 143
8.3.6 STEP 6: Select Element Correlation Options . . . 145
8.3.7 STEP 7: Apply Scripts . . . 147
8.3.8 STEP 8: Configure Custom Algorithms . . . 149
8.3.9 STEP 9: Test and Generate the Configuration . . . 150
8.3.10 STEP 10: Schedule Updates (Regenerations) of the Configuration . . . 151
8.4 Enabling Auditing, Debug, and Element Locking During Generation . . . 151
8.5 Managing Multiple Definitions . . . 152
8.6 Backing Up, Restoring and Deleting Definitions . . . 152
8.6.1 Exporting a SCM Definition . . . 153
8.6.2 Importing a SCM Definition . . . 153
8.6.3 Deleting a SCM Definition . . . 153

9 Use Case: Merging Configuration Management and Asset Databases 155

9.1 Scenario: Building a View of Services and Supporting Technology . . . 156
9.2 Scenario: Building an Organization Structure Based on Discovered Items . . . 159
9.2.1 Defining a Custom Element Menu Option . . . 159
9.2.2 Defining the Service Configuration . . . 160

10 Use Case: Mapping Application Dependencies 163

10.1 Understanding Dependency Mapping . . . 163
10.2 Scenario: Displaying Network Dependencies . . . 164
10.2.1 Scenario Step 1: Creating a Parent Element in Services . . . 164
10.2.2 Scenario Step 2: Identifying an Existing Hierarchy . . . 165
10.2.3 Scenario Step 3: Defining the Sources of State Information . . . 165
10.2.4 Scenario Step 4: Defining Dependencies . . . 165
10.2.5 Scenario Step 5: Configuring Generation Under Modeling Policies . . . 166
10.2.6 Scenario Step 6: Generating a New Hierarchy & Schedule Configuration Regeneration . . . 167
10.3 Showing Impacted Services . . . 168
10.3.1 Understanding Impacted Services . . . 168
10.3.2 Excluding an Element from Impact . . . 169

11 Use Case: Correlating Network Objects 171

12 Use Case: Automating Change Management 175

12.1 Lifecycle Support: Time-Based Snapshots to Manage Changes . . . 175
12.1.1 Scenario Description: Production Environment Baseline . . . 175
12.1.2 PART 1: Set Up a Production Candidate . . . 177
12.1.3 PART 2: Set Up the Production Snapshot . . . 179
12.2 Drift Reporting to Highlight Changes . . . 182
12.2.1 Scenario Description: Shows Only Changed Elements . . . 182
12.2.2 Scenario Steps . . . 183
12.3 Use Case: Creating a Configuration Management Database . . . 186
12.3.1 Scenario Description: Creating a View from Multiple Data Sources . . . 186
12.3.2 LAYER 1: Set Up an Initial Structure to be Leveraged for the CMDB . . . 187
12.3.3 LAYER 2 – Define a Second Definition That Provides State Information and Generate the CMDB360° View . . . 189
12.4 Reconciliation . . . 194
12.4.1 Scenario Description: Show Critical Services (Outages) with No Matching Trouble Tickets . . . 194
12.4.2 Scenario Description: Show Critical Hosts Only . . . 199

13 Use Case: Reporting Service Configuration Problems 203

13.1 Identifying Unmanaged Services . . . 203
13.1.1 Scenario Description: View All Services That are Not Managed . . . 204
13.1.2 Scenario Steps . . . 205
13.2 Problem Management . . . 208
13.2.1 Scenario Description: Managing Trouble Tickets by Service . . . 209
13.2.2 Scenario Steps . . . 209

Part III Using The View Builder Repository 213

14 Editing View Builder Repository Files 215

14.1 Understanding XML Tags for Operations Center Elements and Views . . . 215
14.2 Editing XML Files . . . 217
14.3 Adding Elements or Comments . . . 218
14.4 Moving Elements . . . 219
14.5 Cutting, Copying, and Pasting Tags . . . 219
14.6 Deleting Tags . . . 219

15 Importing and Exporting Element Hierarchies 221

15.1 Importing an External XML File into the View Builder Repository . . . 221
15.2 Saving (Exporting) an Element Hierarchy Structure . . . 222
15.3 Performing a Server-Based Export . . . 223

16 Executing XML Files to Update Elements 225

A LDAP Expression Syntax for Searching and Matching 227

About This Guide

The *Service Modeling Guide* provides information for modeling your business processes.

Part I, “Creating Service Models in the Operations Center Console,” on page 11

- Chapter 1, “Introduction to Modeling,” on page 9
- Chapter 2, “Building Service Models,” on page 13
- Chapter 3, “Populating Service Models,” on page 25
- Chapter 4, “Defining Classes, Behavior Models, and Property Pages,” on page 37
- Chapter 5, “Adding Custom Properties to Service Elements,” on page 89
- Chapter 6, “Understanding Element Relationships,” on page 95

Part II, “Using the Service Configuration Manager,” on page 121

- Chapter 7, “Introduction to the Service Configuration Manager (SCM),” on page 123
- Chapter 8, “Creating a Service Configuration,” on page 125
- Chapter 9, “Use Case: Merging Configuration Management and Asset Databases,” on page 155
- Chapter 10, “Use Case: Mapping Application Dependencies,” on page 163
- Chapter 11, “Use Case: Correlating Network Objects,” on page 171
- Chapter 12, “Use Case: Automating Change Management,” on page 175
- Chapter 13, “Use Case: Reporting Service Configuration Problems,” on page 203

Part III, “Using The View Builder Repository,” on page 213

- Chapter 14, “Editing View Builder Repository Files,” on page 215
- Chapter 15, “Importing and Exporting Element Hierarchies,” on page 221
- Chapter 16, “Executing XML Files to Update Elements,” on page 225

Appendix A, “LDAP Expression Syntax for Searching and Matching,” on page 227

Audience

This guide is intended for Operations Center system administrators.

Feedback

We want to hear your comments and suggestions about this manual and the other documentation included with this product. Please use the *User Comments* feature at the bottom of each page of the online documentation.

Additional Documentation & Documentation Updates

This guide is part of the Operations Center documentation set. For the most recent version of the *Service Modeling Guide* and a complete list of publications supporting Operations Center, visit our Online Documentation Web Site at Operations Center 5.5 online documentation.

The Operations Center documentation set is also available as PDF files on the installation CD or ISO; and is delivered as part of the online help accessible from multiple locations in Operations Center depending on the product component.

Additional Resources

We encourage you to use the following additional resources on the Web:

- NetIQ User Community (https://www.netiq.com/communities/): A Web-based community with a variety of discussion topics.
- NetIQ Support Knowledgebase (https://www.netiq.com/support/kb/product.php?id=SG_XOPERATIONSCENTER_1_2): A collection of in-depth technical articles.
- NetIQ Support Forums (https://forums.netiq.com/forumdisplay.php?26-Operations-Center): A Web location where product users can discuss NetIQ product functionality and advice with other product users.

Technical Support

You can learn more about the policies and procedures of NetIQ Technical Support by accessing its Technical Support Guide (https://www.netiq.com/Support/process.asp#_Maintenance_Programs_and).

Use these resources for support specific to Operations Center:

- Telephone in Canada and the United States: 1-800-858-4000
- Telephone outside the United States: 1-801-861-4000
- E-mail: support@netiq.com (support@netiq.com)
- Submit a Service Request: http://support.novell.com/contact/ (http://support.novell.com/contact/)

Documentation Conventions

A greater-than symbol (>) is used to separate actions within a step and items in a cross-reference path. The > symbol is also used to connect consecutive links in an element tree structure where you can either click a plus symbol (+) or double-click the elements to expand them.

When a single pathname can be written with a backslash for some platforms or a forward slash for other platforms, the pathname is presented with a forward slash to preserve case considerations in the UNIX* or Linux* operating systems.

A trademark symbol (®, ™, etc.) denotes a NetIQ trademark. An asterisk (*) denotes a third-party trademark.

1 Introduction to Modeling

Operations Center provides different ways to integrate and correlate technology and business data from nearly any source, including network and systems management tools, databases, and internally developed applications. Map this data to business processes that support IT systems, and generate desktop dashboards that provide greater visibility into end-to-end IT systems value, performance, and utilization.

Modeling your business processes enables monitoring and managing availability and performance of the applications and services that you deliver to the business.

Operations Center provides a modular set of modeling solutions, so you can adopt specific applications that solve specific challenges. This guide focuses on using the Console's *Service Models* hierarchy and SCM to create service models. Table 1-1 lists methods of creating service models in Operations Center:

Table 1-1 *Methods of Creating Service Models*

Operations Center Modeling Tool	Allows you to:
Service Models Hierarchy (available in the Operations Center Console)	◆ Manully add elements to define attributes and relationships of enterprise resources using the *Service Models* element hierarchy. Enterprise resources can include any component or aspect of a business units, such as IT infrastructure components, personnel, or events and transactions. ◆ Define classes, behavior models, and custom properties and assign them to service model elements.
Service Configuration Manager (SCM)	◆ Generate new element hierarchies from multiple, varied data sources. ◆ Automatically discover dependencies and identifies scheduled and unscheduled changes. ◆ Integrate the mapping of element relationships across an enterprise. ◆ Integrate asset, configuration, and change sources into one business view. ◆ Build and maintain Service Views automatically and dynamically—alleviating the need to create, update, and maintain views (models) manually.
Layout View (available in the Operations Center Console)	◆ Create visual representations of element, service, or process information using graphics and drawing tools. ◆ Use the dynamic linking feature binds graphics in a drawing to element conditions and attributes, thus enabling automatic updates of the drawing when elements change. ◆ For information on this feature, see the *Operations Center 5.5 Custom Drawing and Layout Guide*.

Operations Center Modeling Tool	Allows you to:
Server, Configuration, Export, and Import commands (available in the Operations Center Console)	◆ Creates, edits, and shares element hierarchies from the *Service Models* hierarchy. ◆ Dynamically creates or maintains element branches in the *Service Models* hierarchy instead of manually creating them. ◆ These commands support exporting ACL permissions, custom properties, drawing attributes, and SLA properties. ◆ For information on using the Server commands, see the *Operations Center 5.5 Server Configuration Guide*.

After defining the components of a service model, populate it with elements from different sources. These elements drive the state of the service model. They are brought into Operations Center using standard adapters, custom adapters, and auto-discovery tools. It is also possible to generate elements from external data sources using the Service Configuration Manager (SCM) or the Import Configuration feature. In addition, you can drag and drop elements from anyplace in the *Elements* hierarchy to a service model.

Table 1-2 summarizes different methods of populating a service model:

Table 1-2 *Methods of Populating Service Models*

Operations Center Feature	Description
Match Elements by Criteria (available in a service model element's *Elements* property page)	◆ Assigns elements to service models using element name or DName, element property, or element class.
Standard Adapters	◆ Connects to and communicates with third-party management systems. Elements, properties, and alarms from these management systems can drive the state of a specified service model. ◆ Operations Center ships with a large number of adapters, each written for a specific network or systems management product. ◆ The complete list of standard adapters, as well as the integration and configuration steps necessary to integrate each one, are described in the *Operations Center 5.5 Adapter and Integration Guide*.
Data Integrator	◆ A custom adapter that brings in data from a defined database and builds out elements, properties, and alarms. Provides a way to access business metrics from databases not available using standard adapters. ◆ The Data Integrator is an optional component of the Operations Center platform and is licensed as a separate product. For more information, see the *Operations Center 5.5 Data Integrator Guide*.
Drag and Drop elements	◆ A method of manually fine-tuning a service model using existing elements in other hierarchies. Move or copy elements from one hierarchy to another by dragging and dropping them within the Explorer pane.

Creating Service Models in the Operations Center Console

The following sections provide instructions about building Service Models including classes, behavior models and property pages:

- Chapter 2, “Building Service Models,” on page 13
- Chapter 3, “Populating Service Models,” on page 25
- Chapter 4, “Defining Classes, Behavior Models, and Property Pages,” on page 37
- Chapter 5, “Adding Custom Properties to Service Elements,” on page 89
- Chapter 6, “Understanding Element Relationships,” on page 95

2 Building Service Models

Enterprise resources can include any aspect of a business, such as people, infrastructure components, locations, or IT resources. Enterprise resources are used to develop a service model that defines how a set of enterprise resources impacts a business function. Enterprise resources can be further defined by setting up classes and properties that are applied to an element when it is created or surfaced.

Constructing a service model begins by defining a hierarchy that represents an aspect of the enterprise. A service model can be defined for a wide range of enterprise resources including:

- Services, such as an online store or credit card history, and a subdepartmental view of e-mail servers
- Geographical or departmental divisions
- Infrastructure components, such as servers

To build service models, review the following sections:

- Section 2.1, “Understanding Service Models,” on page 13
- Section 2.2, “Understanding Service Models Components,” on page 14
- Section 2.3, “Creating Service Models,” on page 16
- Section 2.4, “Service Model Properties,” on page 19
- Section 2.5, “Assigning Access Privileges to Service Models,” on page 21
- Section 2.6, “Modifying Standard Properties for Service Models,” on page 21
- Section 2.7, “Deleting Service Models,” on page 23

2.1 Understanding Service Models

Operations Center uses service models to provide a configurable outlook of an IT environment from a business perspective. A typical IT environment has multiple system management products that provide tools for monitoring the performance of networks, applications, and services. Service models are used to analyze, monitor and manage the impact of the business interactions that exist among these systems. A service model is a unique set of enterprise resources that pertain to an outlook. For example, service models can be created for a business function, department, or geographical location.

A service model presents a set of elements in a hierarchy that represents the relationships among the elements. Operations Center tools create element hierarchies based on criteria to provide a business view of an environment. Operations Center can associate information from various management platforms into these service models to provide an end-to-end view of the current state of the elements across an enterprise.

Condition algorithms are used to refine system reporting. For example, if one of two network routers is down, the network can be shown in a degraded state rather than completely down, as most system monitors might report. In a more complex situation, Operations Center software can greatly reduce

the number of extraneous messages generated by a network outage and allow you to focus on the issues causing the network problem. To go one step further, condition algorithms can be used to create intelligent automated responses to changes across the enterprise.

The process of building a meaningful service model begins with a clear understanding of enterprise resources and a plan for organizing a business view. Table 2-1 lists IT resources that are commonly represented in a service model:

Table 2-1 *Service Resource Categories*

Category	Monitored Objects	Explanation
Network	Routers, switches, Virtual Private Networks (VPNs), ports, and so on.	The paths that exist between the enterprise management platforms and the servers where the application components exist.
Users	Workstations, routers, switches, Virtual Private Networks, ports, and so on.	The devices and paths that allow users to connect to an application front-end (such as a Web server).
Servers	CPU, memory, and file system space utilization, and so on.	Includes the servers on which application processes run, as well as other servers on which an application might be dependent.
Application	Running processes, services, tasks, and so on.	Processes and services that must run to drive an application.
Databases	Database tables, locks, database processes and services, and so on.	Also monitor the processes, services, and servers that support the database itself.

Not all business services use all of these categories. Additional categories can be added, if necessary, to support more complex requirements, such as performance and messaging.

2.2 Understanding Service Models Components

Use service models to group elements in an enterprise that represent the various management activities or services related to a business. For example, an *Order Processing* service can consist of the following elements: *Order Capture*, *Billing*, *Manufacturing*, and *Delivery Services*. These elements can be subdivided into functional or physical groupings. For example, *Order Capture* can consist of a *Web Farm*, *Database Farm*, *Network Communications*, and an *Order Capture application*.

Service models are presented in element hierarchies that can consist of elements (*Elements*), geographical sites (*Locations*), or other business resources. Each parent element has a state that is determined by the most severe state of its child. For example, if the state of the Billing system is CRITICAL, then the state of its parent, Order Processing Service, is also CRITICAL. The color of the *Service Model* icon identifies its state.

Elements can be grouped based on their geographical location under *Location*. For example, if technicians are responsible for particular geographical locations, create a site for each location, then link to the corresponding technology in the *Elements* tree. Figure 2-1 shows an example of a service model.

There are different ways to populate a service model:

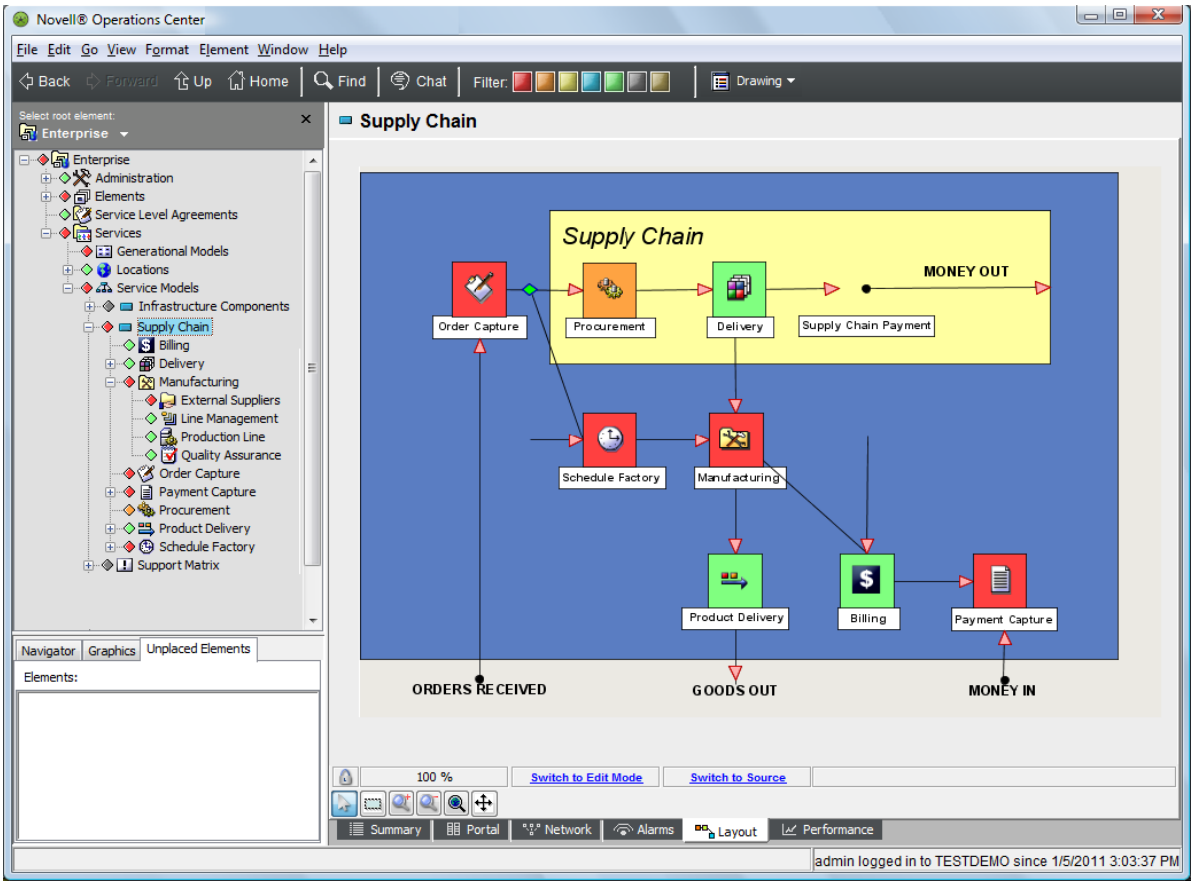
- Manually create each element.

- Import hierarchies using the *Server* > *Configuration* > *Import* option.
- Use SCM to generate new element hierarchies from multiple, varied data sources.

In Figure 2-1, all the first-level elements under *Supply Chain* were created manually using the *Add Element* option. Child elements with state information were added under these first-level elements using the *Match By* criteria feature, which is explained in Section 3.1.1, "Assigning Elements to a Service Model," on page 26.

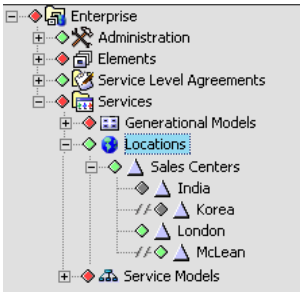
The *Layout* view was created to graphically depict the business process. A quick glance at the element states shows the *Manufacturing*, *Order Capture*, *Payment Capture*, and *Schedule Factory* are CRITICAL and require attention. For more information about creating custom Layout views, see the *Operations Center 5.5 Custom Drawing and Layout Guide*.

Figure 2-1 *Service Model Element Hierarchy and Associated Layout Drawing*



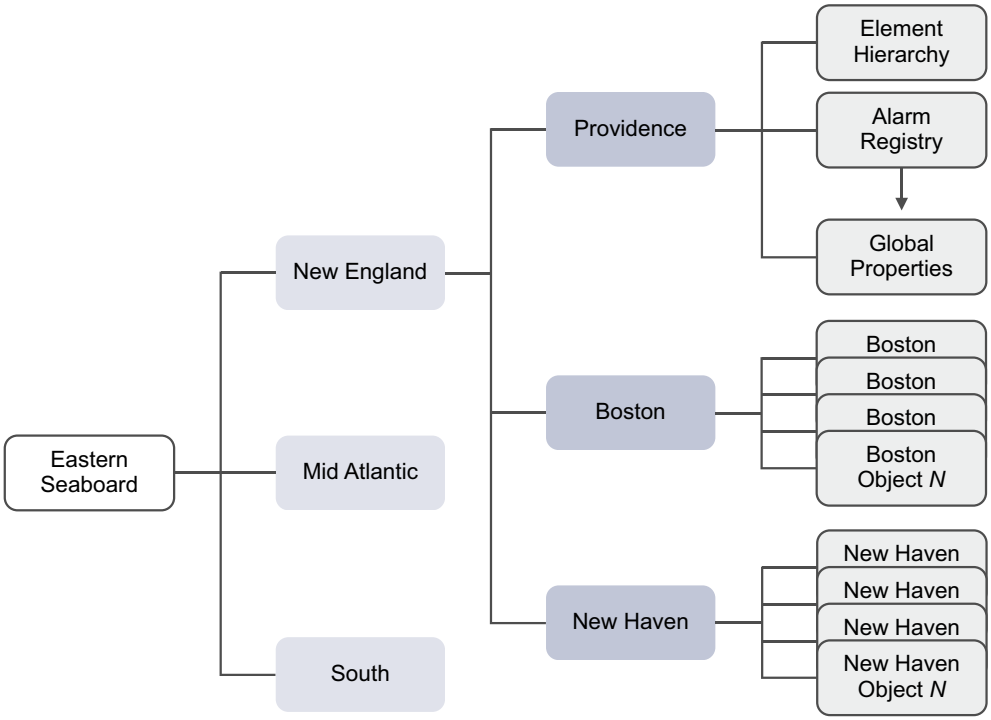
Locations elements can consist of *Service Models*, elements (such as *Elements*), or other geographical sites. Each site has a state that is based on the states of its underlying sites, or elements. The state of a site is identified by the color of the site icon.

Figure 2-2 *Service Model by Location*



Before defining service models, determine how your company organizes network management activities, business processes, or its information. The configuration should reflect the way your company works. The following illustration shows a hierarchy set up by geographic location:

Figure 2-3 *Sample Service Model Hierarchy*



2.3 Creating Service Models

The process of creating a service model involves the following general steps:

1. To include geographic sites in service models, first define them in the *Services* > *Locations* hierarchy.
2. Use the *Add Element* option to define the major components of a *Service Models* element.

3. Use a variety of sources to populate the *Service Model* elements.

   These source elements determine the service model state and can be used for root cause and impact analysis.

   This is documented in Chapter 3, "Populating Service Models," on page 25.

4. Service model elements can have custom property pages. Fill in the custom property values if applicable.

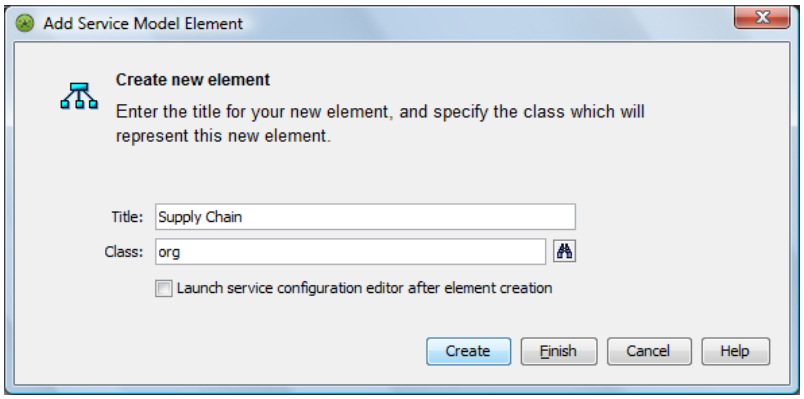
   This is also documented in Chapter 3, "Populating Service Models," on page 25.

To create a service model, do the following:

- Section 2.3.1, "Creating a Service Model," on page 17
- Section 2.3.2, "Adding Child Elements to an Existing Service Model," on page 19

2.3.1 Creating a Service Model

1. In the Explorer pane, expand *Administration* > *Services*.
2. Right-click *Service Models*, *Locations*, or any subfolder, then click *Add Element* to open the Add Service Model Element dialog box.

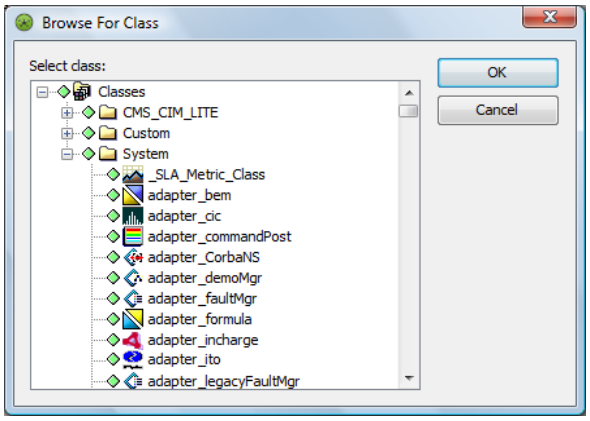


3. Specify the name of the new service model in the *Title* field.

   Element names in Operations Center are case-sensitive. For example, the elements *MyORG* and *myorg* are not the same. If *MyORG* is imported into a system that already has *myorg*, it does not replace *myorg*, but is added as a new element. Element names are unique, based on case sensitivity, whether they are created using *Add Element* or are imported.

4 To select a class other than `org` for the service model, click Browse .

The Browse for Class dialog box opens, displaying available classes:



5 Navigate to the class to be assigned to the service model, then click *OK*.

The icon associated with the selected class is assigned to the new element.

Class names are not required to be unique in the *Metamodel* hierarchy. Multiple instances of a class name can exist. Use the fully qualified class name to select the correct class.

To learn more about creating custom classes, see Chapter 4, "Defining Classes, Behavior Models, and Property Pages," on page 37.

6 Do one of the following:

- To view and edit the new element's property pages, click *Create*.

  The element is created and the element's property pages open.
- Click *Finish* to create the element and close the dialog box.

7 Do the following:

7a Continue defining service model components by right-clicking the top-level element you just created, then selecting the *Add Element* option, as described in "Adding Child Elements to an Existing Service Model" on page 19.

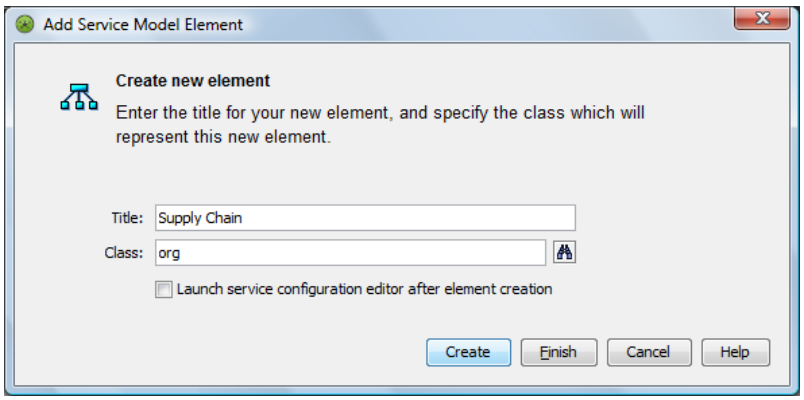
7b Populate these components using the methods described in Chapter 3, "Populating Service Models," on page 25.

2.3.2 Adding Child Elements to an Existing Service Model

After defining the major components of a service model, define the subcomponents by using the *Add Element* option to add child elements.

To add a child element to a service model element:

1. In the Explorer pane, expand *Service Models*.
2. Right-click an element, then select *Add Element* to open its dialog box:



3. Specify a name for the element in the *Title* field.
4. To assign the element to a class, do one of the following:
   - Click Browse and navigate to the class for the element.
   - Specify a fully qualified class name in the *Class* field.
   - Specify a new class name in the *Class* field.

   The new class displays in the *Custom* folder under *Classes*.
5. Click *Finish*.

   The new element is added as a child of the selected service model element.

2.4 Service Model Properties

When service model elements are created, they have the standard element property pages, such as *Status*, *Condition*, and *Comments*. However, they also have a unique *Elements* property page, which is used to select elements that drive the service model element's conditions and alarms. For more information, see "Assigning Elements by Matching Properties" on page 26.

The *Condition* property page has special considerations. For more information, see the following section.

Custom property pages can be added to a service model element in two ways:

1. The element's class can determine which custom property pages are automatically assigned to it.

   For more information, see "About Custom Classes, Behavior Models, and Property Pages" on page 38.
2. Custom property pages can be added to individual service model elements using the Manage Properties feature.

   For more information, see "Editing Custom Properties" on page 89.

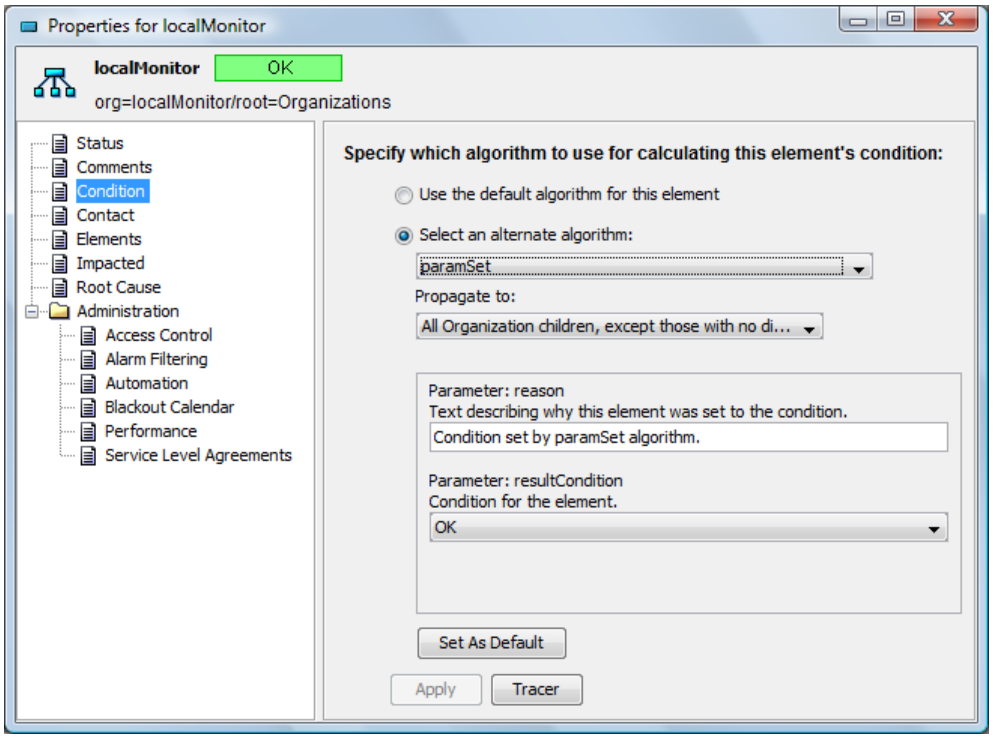
2.4.1 Specifying the Algorithm to Determine an Element's Condition

Service model and location elements are defined in terms of other elements, such as routers, systems, devices, applications, or even other locations or sites. In the simplest scenarios, these objects derive their condition (often thought of as the state of the object) from the highest severity of all other objects on which they depend. Operations Center software provides customizable algorithms that determine how service model and location elements calculate their conditions.

For additional information about using algorithms to change the state of elements, see "Using Algorithms to Calculate Element State" in the *Operations Center 5.5 Server Configuration Guide*.

To specify an algorithm for a service model or location:

1. Right-click a service model element, then select *Properties*.
2. Click *Condition* in the left pane to display the *Condition* property page:



3. To specify an algorithm, select one of the following radio buttons:

   Use the default algorithm for this element: Takes the highest condition level from all children.

   Select an alternate algorithm: Uses a custom algorithm.

4. Click the drop-down list under the radio buttons, then select the algorithm to use.
5. Click the *Propagate To* drop-down list, then select the inheritance method:

   No Children: Applies the algorithm to the element only, and not to any children.

   All children: Applies the algorithm to the element and all children created within the element and linked to the element.

   All Organization children, except those with no children (might force discovery): Applies the algorithm to the element and only the children created under a *Services* element, but not to element children linked to the element. A discovery process determines if the child element has children.

All Organization children, except those with no discovered children (discovery is not forced):

Applies the algorithm to a *Services* element and only those children currently known to have children created under the element. No discovery process is run to determine the child element's status.

6 If the algorithm requires parameters, supply them in the *Parameter* section.

7 (Optional) Click *Set As Default* to save selections as the default algorithm settings for new elements.

8 Click *Apply*.

For more information about algorithms, see "Using Algorithms to Calculate Element State" in the *Operations Center 5.5 Server Configuration Guide*.

2.5 Assigning Access Privileges to Service Models

Every service model inherits the security of its parent unless additional security is specified. Modify access privileges for service models using the *Access Control* panel in the *Portal* view or the *Access Control* property page. For more information, see the *Operations Center 5.5 Security Management Guide*.

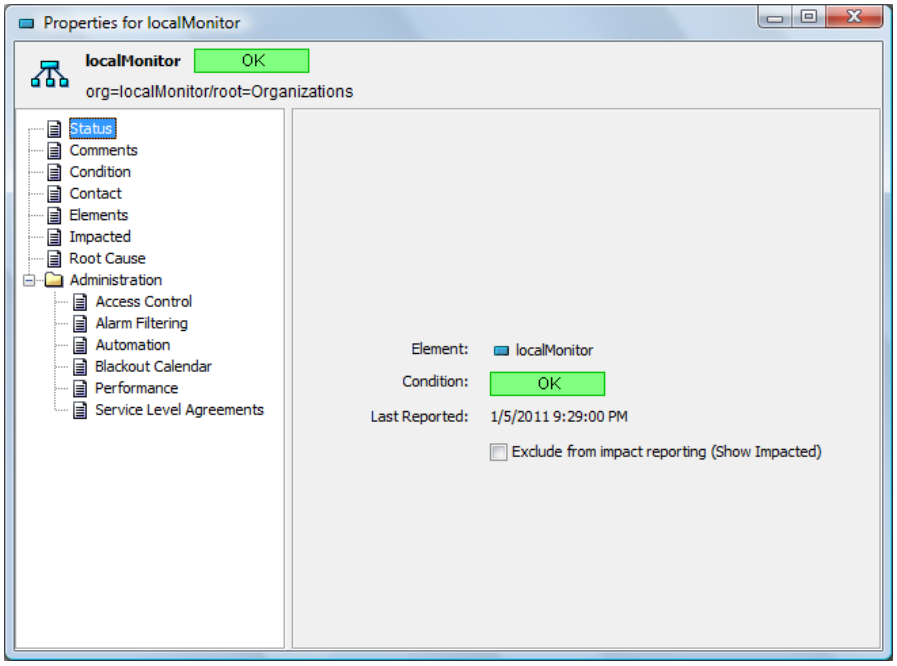
2.6 Modifying Standard Properties for Service Models

Contact information, access control, element selection and other information that defines a service model element can be modified through the property pages.

To modify information using the property pages:

1 In the Explorer pane, expand *Services* > *Service Models*.

2 Right-click the element you want to edit, then select *Properties* to open the Status property page:



3 In the left pane, click one of the following:

Status: Displays the label, condition and date and time of the last reported condition for the service model. Cannot be modified, except for the *Exclude from Impact Reporting (Show Impacted)* check box. Select this check box to exclude the element from impact reporting. For information on the root cause and impacted elements features, see "Viewing Root Cause and Impacted Elements" in the *Operations Center 5.5 User Guide*.

Comments: Add text comments regarding the element. Historical comments also display.

Condition: Select the algorithm used to calculate the element's condition. For instructions, see "Specifying the Algorithm to Determine an Element's Condition" on page 20.

Contact: User contact information, such as full name, phone numbers and e-mail addresses.

Elements: Specify the elements that drive this element's conditions and alarms. For more information on selecting elements, see "Creating a Matching Rule" on page 65.

Impacted: The Show Impacted feature identifies higher-level elements that are affected by the condition of the selected element. For more information, see "Viewing Root Cause and Impacted Elements" in the *Operations Center 5.5 User Guide*

Root Cause: Use the Root Cause feature to identify lower-level elements that caused a change in the condition of the selected (higher-level) element. For more information, see "Viewing Root Cause and Impacted Elements" in the *Operations Center 5.5 User Guide*.

Administration: Only administrators should access the following property pages.

Access Control: Maintain access privileges of users and groups associated with this service model element. For more information, see the *Operations Center 5.5 Security Management Guide*.

Automation: Maintain the types of automation events, actions, and descriptions related to this service model element. For more information on the Automation feature, see "Defining and Managing Automation Events" in the *Operations Center 5.5 Server Configuration Guide*.

Blackout Calendar: Define a blackout calendar requirement, which halts data capture at times when the element has planned or unplanned maintenance or down time that does not affect any other elements in the hierarchy. For information on blackout calendars, see "Setting Blackout Calendars on Elements" in the *Operations Center 5.5 Server Configuration Guide*.

Performance: Displays a list of all available performance profiles. Select a check box to add the element to a profile. The Data Warehouse uses profiles to select the elements for which alarm history is collected. For information on profiles, see "Capturing Alarm and Performance History" in the *Operations Center 5.5 Server Configuration Guide*.

Service Level Agreements: Identifies Service Level Agreements or objectives that are defined for the element. To learn more about Service Level Agreements, see the *Operations Center 5.5 Service Level Agreement Guide*.

4 Modify the settings as needed.

5 To save the updates do one of the following:

- Click *Apply* to apply the edits to the currently displayed element.
- Click *Apply to All* to apply all edits made to all elements in the current session.

2.7 Deleting Service Models

Deleting a service model deletes only the elements displayed in the *Explorer* pane. It does not delete the classes, behavior models, property pages, or properties linked within the service model.

To delete a service model:

1. In the Explorer pane, expand *Services* > *Service Models*, then navigate to the service model element that should be deleted.
2. Right-click the element, then select *Remove* to open a confirmation dialog box.

   Multiple elements could be selected for removal.
3. Click *Yes* to delete the service model.

   The service model is removed from the *Services* hierarchy.

3 Populating Service Models

After defining the components of a service model, populate it with elements from different sources. These elements drive the state of the service model. They are brought into Operations Center using standard adapters, and custom adapters. It is also possible to generate elements from external data sources using the Service Configuration Manager (SCM) or the Import Configuration feature. In addition, you can drag and drop elements from anyplace in the *Elements* hierarchy to a service model.

This section focuses on using the property matching criteria to assign elements, and the options used to move or copy elements within the Explorer pane. Table 3-1 summarizes the different methods for populating a service model:

Table 3-1 *Methods for Populating Service Models*

Operations Center Feature	Description
Match Elements by Criteria (available in a service model element's Elements property page)	◆ Assigns elements to a service model using the following criteria: element name or DName, element property, or element class
Standard Adapters	◆ Connects to and communicates with third-party management systems. Operations Center ships with a large number of adapters, each written for a specific network or systems management product. ◆ The complete list of standard adapters, as well as the integration and configuration steps necessary to integrate each one, are described in the *Operations Center 5.5 Adapter and Integration Guide*.
Data Integrator	◆ A custom adapter that brings in data from a defined database and builds out all the elements, properties, and alarms. Provides a way to access business metrics from databases not available using standard adapters. ◆ The Data Integrator is an optional component of the Operations Center platform and is licensed as a separate product. For more information, see the *Operations Center 5.5 Data Integrator Guide*.
Drag and Drop elements	◆ A method of manually fine-tuning a service model. Simply drag and drop elements from anyplace in the *Elements* hierarchy to a service model component. There are additional menu options to copy, cut, paste, and link elements within the Explorer pane.

To populate service models, review the following sections:

- Section 3.1, "Assigning Elements by Matching Properties," on page 26
- Section 3.2, "Stopping State Contributions or Alarm Propagation from Source Elements," on page 32
- Section 3.3, "Analyzing Dependencies," on page 33
- Section 3.4, "Copying, Moving and Linking Elements," on page 34

- Section 3.5, “Renaming Service Model Elements,” on page 34
- Section 3.6, “Deleting Elements and Linked Relationships,” on page 35

3.1 Assigning Elements by Matching Properties

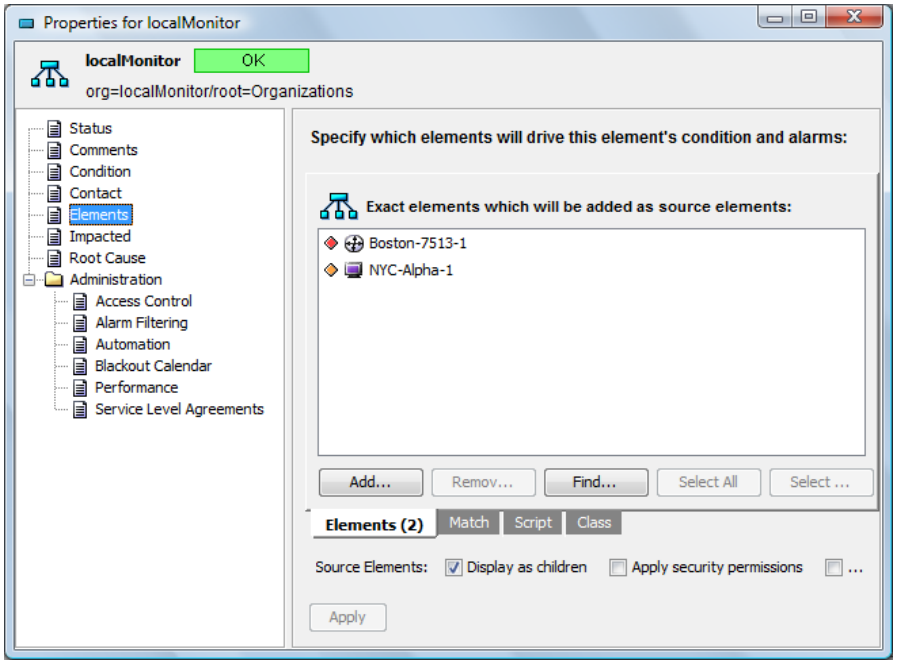
This section explains how to assign elements to a service model using criteria, such as element or class names. The last four sections are optional procedures that are referenced from within the first section’s steps:

- Section 3.1.1, “Assigning Elements to a Service Model,” on page 26
- Section 3.1.2, “Matching by Specific Elements,” on page 27
- Section 3.1.3, “Matching by Expression,” on page 28
- Section 3.1.4, “Matching by Script Expression,” on page 29
- Section 3.1.5, “Matching by Class,” on page 31

3.1.1 Assigning Elements to a Service Model

To assign elements to a service model using the *Elements* property page:

1. In the Explorer pane, right-click a service model element, then select *Properties*.
2. In the left pane, click *Elements* to update the property page:



3. Click one of the following tabs to specify a method to select elements (each one requires a different set of criteria to locate matching elements):

   Elements: Selects specified elements.

   For instructions, see “Matching by Specific Elements” on page 27, then return here to Step 4.

Match: Selects elements based on a regular expression or LDAP-style syntax. The match is based on the element's full DName.

For instructions, see "Matching by Expression" on page 28 or Appendix A, "LDAP Expression Syntax for Searching and Matching," on page 227, then return here to Step 4.

Script: Selects elements based on a script written using the NOC Script language. The entire text in the window must evaluate to either True or False for a given object. Only administrators with programming experience should use this feature. It is reserved for situations that require statements that exceed the complexity of an element expression, or that require validation of a property other than an element's DName.

For instructions, see "Matching by Script Expression" on page 29, then return here to Step 4.

For more information about NOC Script, see the *Operations Center 5.5 Scripting Guide*.

Class: Selects elements based on element class name.

For instructions, see "Matching by Class" on page 31, then return here to Step 4.

4 Select the following check boxes as appropriate to determine the behavior for elements created for this service model:

Display as children: Displays source elements as linked children under the service model.

Apply security permissions: Applies the source element's security permissions to the element created for the service model.

Halt alarm contributions: Stops the propagation of alarms from the source element up to the service model.

5 Specify an algorithm to determine the element's state.

For instructions, see "Specifying the Algorithm to Determine an Element's Condition" on page 20. Then, return here to Step 6.

6 Click *Apply* to add the new element to the service model hierarchy.

7 Click *New* to add another element to update the Add Service Model Element dialog box.

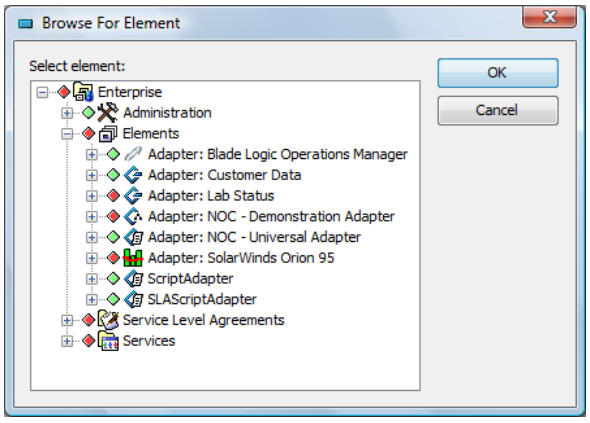
3.1.2 Matching by Specific Elements

Matching by elements enables you to select specific elements as source elements for the service model.

To match specific elements:

1 From the Add Service Model Element dialog box, click the *Elements* tab to update the dialog box.

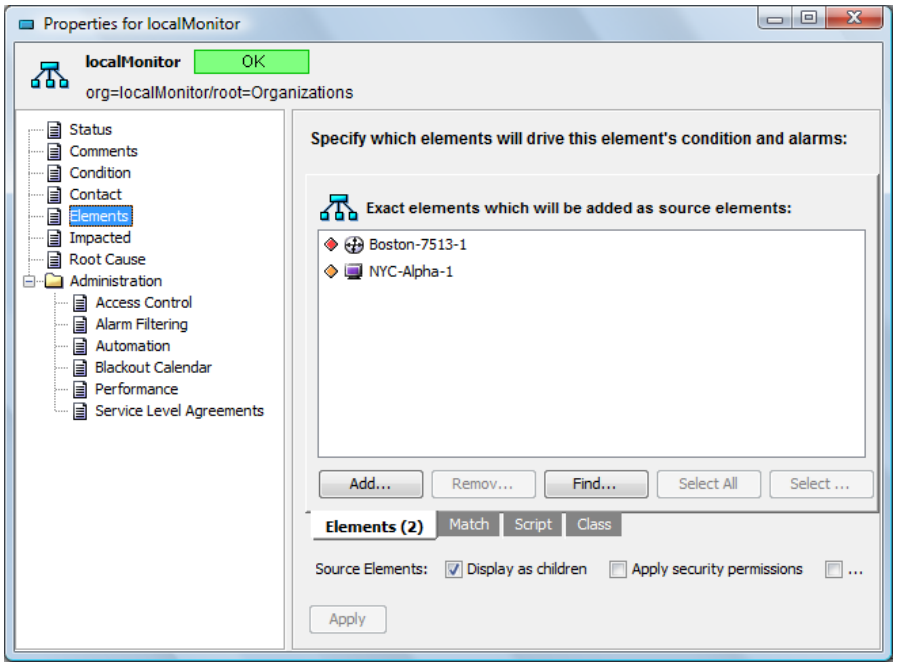
2 Click *Add* to navigate to an element to open the Browse for Element dialog box:



3 In the Browse for Element dialog box, navigate to the elements you want to use, then click *OK*.

These are exact element selections. Child elements are not selected automatically when their parents are selected.

Use Ctrl+click or Shift+click to select multiple elements at once. The selected elements display in the Add Element dialog box:



4 To search for an element, click *Find* on the Add Service Model Element dialog box to open the Find dialog box.

For more information about finding elements, see "Using Find to Search for Elements" in the *Operations Center 5.5 User Guide*.

5 Return to Step 4 on page 27.

3.1.3 Matching by Expression

Match by expression uses regular expressions or LDAP-style syntax. All element expression matches are based on an element's DName. Each element has a unique DName, which displays by right-clicking an element, then selecting *Properties*. The DName displays at the top of the property page.

For information about using LDAP expressions, see Appendix A, "LDAP Expression Syntax for Searching and Matching," on page 227.

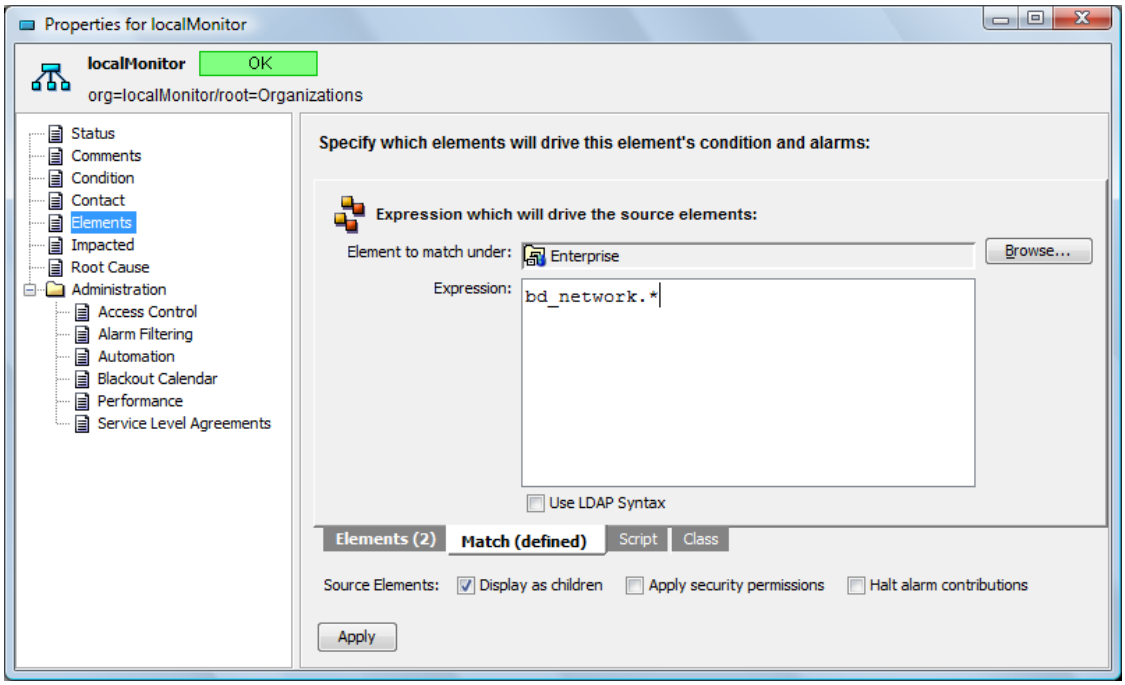
To match elements using an expression:

1 From the Add Service Model Element dialog box, click the *Match* tab to update the Add Service Model Element dialog box.

2 Click *Browse* and select the element for the match search.

Only the children of the selected element are eligible for matching.

3 Enter the expression in the *Expression* field.



4 To use LDAP syntax, select the *Use LDAP Syntax* check box.

5 To add the matching element to the service model, click *Apply*.

6 Click *New* to add another element.

IMPORTANT: Deleting an adapter deletes all regular expressions used to match elements associated with the adapter. (These expressions can be viewed in the *Match* tab on the *Elements* property page for a service model element.) Before deleting an adapter, you might want to copy these regular expressions if it is necessary to re-create them later.

---

7 Return to Step 4 on page 27.

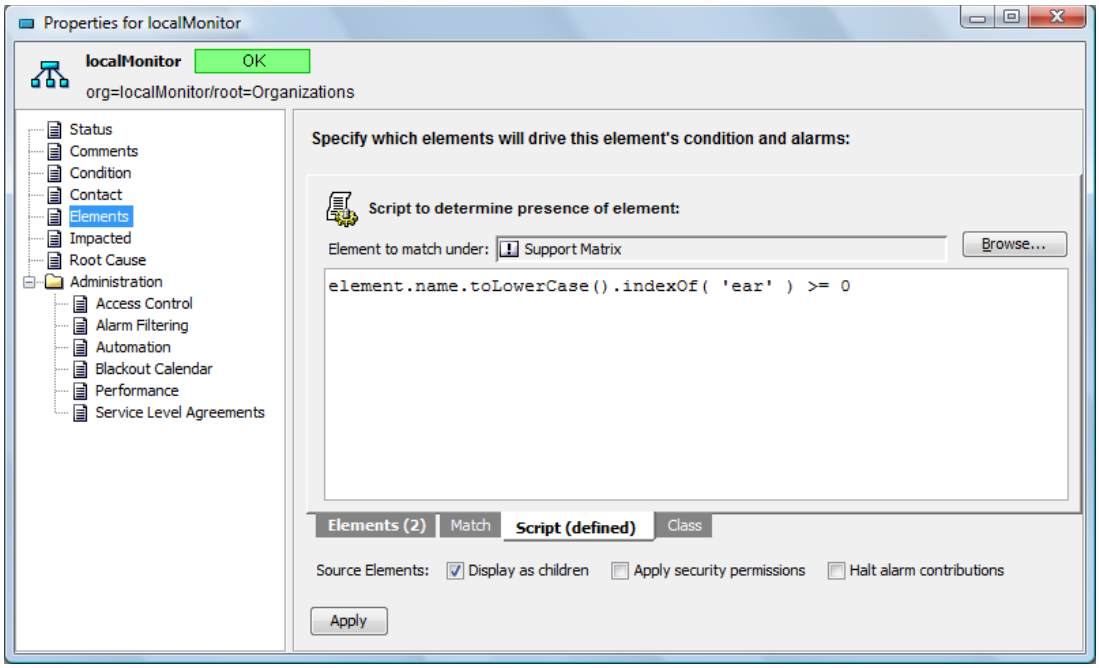
3.1.4 Matching by Script Expression

Script expressions should be reserved for situations that require statements that exceed the complexity of an element expression or that require validation of a property other than an object's DName. Only administrators with a programming background should use script expressions because they are very powerful. An improperly written script can consume enough resources to severely impact the Operations Center server.

The script language expected is NOC Script and the entire text within the window must evaluate to True or False. For more information about creating scripts and scripts available in the Script Library, see the *Operations Center 5.5 Scripting Guide*.

To match elements using a script expression:

1 From the Add Service Model Element dialog box, click the *Script* tab to update the Add Service Mode Element dialog box:



2 Click *Browse* and select the parent element for the match search.

Only the children of the selected element are eligible for matching.

3 Paste or enter the script code in the main field.

The script language expected is NOC Script.

The script must result in a True or False output for elements in the selected element branch.

4 To add the matching element to the service model, click *Apply*.

5 Click *New* to add another element.

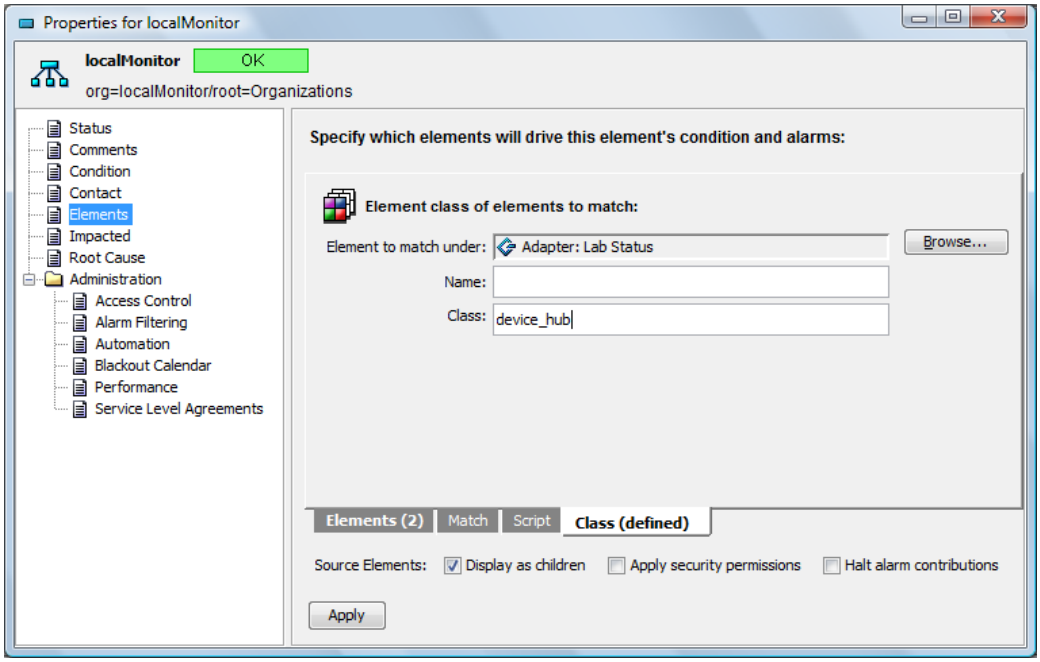
6 Return to Step 4 on page 27.

3.1.5 Matching by Class

Match by class allows you to search for elements based on their class name. For ways to determine an element's class name, see "Determining an Element's Class Name"the *Operations Center 5.5 Server Configuration Guide*.

To match elements by class:

1 From the Add Service Model Element dialog box, click the *Class* tab to update the Add Service Model Element dialog box:



2 Click *Browse* and select the parent element for the match search.

Only the children of the selected element are eligible for matching.

3 Specify the name of the specific element to be found in the *Name* field.

Leave the *Name* field blank to find all elements in the specified class.

4 In the *Class* field, specify the class name to match.

Org is the default class name for a service model.

TIP: The element to match defaults to Enterprise. Selecting a more specific element reduces the processing requirements.

---

5 To add the matching element to the service model, click *Apply*.

6 Click *New* to add another element.

7 Return to Step 4 on page 27.

3.2 Stopping State Contributions or Alarm Propagation from Source Elements

By default, the state of child elements contributes to the state of their parents. In some situations, you might want to prevent an element from contributing the state of a service model. The following methods are available:

- A parent service model can represent a logical grouping within a BSV as a type of container. This service model container organizes data about an service model, but does not contribute to the state of the service model.
- State contributions from a service model can be broken. If a service model is a natural child of a service model, the affect of the native child on the parent service model can be stopped without deleting the child from the hierarchy.
- By default, alarms for child elements contribute to the overall alarm count of their parents. However, it is possible to halt alarm propagation from source elements using the *Halt Alarm Contributions* option, which disables alarms from a specific element. This can break the state-contributing link to a service model container.

Because the default algorithm for service models only includes child service models that have a state-contributing relationship, the algorithm does not normally include children with broken state relationships. A custom algorithm can be created for the parent service model that considers all children regardless of whether there is a broken state relationship to the parent.

- Section 3.2.1, "Stopping a State Contribution or Alarm Propagation," on page 32
- Section 3.2.2, "Hiding or Displaying Nonsource Contributing Link Icons," on page 33

3.2.1 Stopping a State Contribution or Alarm Propagation

To stop a state contribution or alarm propagation from a child element in the service model:

1. Right-click the parent service model for which you wish to stop the state contribution or alarm propagation, then select *Properties* to open the Status property page.
2. In the left pane, click *Administration* > *Elements* to open the *Elements* property page.
3. Select one or more service model elements.
4. Do one of the following:
   - Click *Remove Link* to stop a state contribution from a child element.

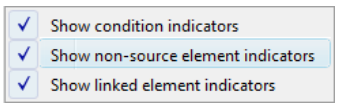
     The element is removed from the list.
   - Select the *Halt Alarm Contributions* check box to stop the roll up of alarm information to the service model.
5. Close the Property page, then click *Yes* when prompted to save the changes.

   In the Explorer pane, the element remains a child of the parent, but displays a nonstate-contributing link indicator (if this features is enabled in the Explorer pane). The element remains in the hierarchy, but no longer contributes to the state of the parent.



3.2.2 Hiding or Displaying Nonsource Contributing Link Icons

Right-click the Explorer pane background, then select *Show Non-Source Element Indicators* to enable or disable the feature:



3.3 Analyzing Dependencies

Dependencies define points in a hierarchy that rely on elements from other branches, or indicate an implied relationship. Based on selections described in previous section (such as *Halt Alarm Contributions*) these relationships could contribute to state.

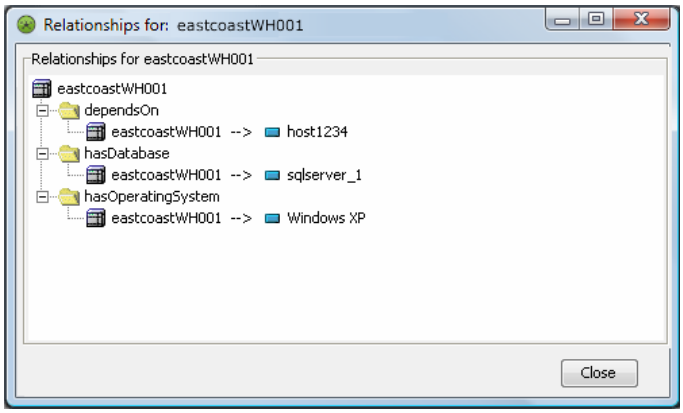
Dependencies are relationships that display in the Operations Center Relationship browser. To open the Relationship Browser, right-click an element, then select *Show Relationships*. For more information about using the Relationship Browser, see Chapter 6, "Understanding Element Relationships," on page 95.

Use an element's *Navigate Relationships* option to quickly view an element's dependency relationships and navigate to one of the end-point elements named in a relationship.

To view an element's nonparent/child dependencies:

1 In the *Explorer* pane, right-click an element in the *Service Models* hierarchy, then select *Navigate Relationships*.

The Relationships dialog box displays the element, and its dependent relationships:



2 To navigate to one of the elements listed as an end-point in the Relationships dialog box, double-click the element.

The element is selected in the *Explorer* pane. For example, in the illustration in Step 1, double-click the operating system relationship to navigate to the *Windows XP* element in the *Explorer* pane.

3.4 Copying, Moving and Linking Elements

Reorganize elements in the *Services* hierarchy using copy, paste, move, and link options from a menu or through drag and drop actions.

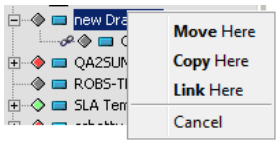
Dragging and dropping an element from the *Elements* hierarchy to a service model element automatically creates a link. This pastes a copy of the element as a child of the target element and retains a link to the original element. Changes made to the original element affect the linked element. Using the right-click menu *Copy* and *Paste* options also creates a link.

However, within the *Services* hierarchy, it is possible to copy, cut, paste or link elements.

To copy, move or link an element:

1. In the *Element* or *Services* hierarchy in the *Explorer* pane, navigate to the element to copy, move, or link.
2. Expand the hierarchy to display the destination element in the *Services* hierarchy.
3. Use the right mouse button to drag and drop an element to the target element.

   The following options display:



4. Select one of the following options:

   Move Here: Cuts and pastes the element as a child of the target element. Pressing Shift and dragging the element also moves it.

   Copy Here: Pastes a copy of the element (and its children) as a child of the target element. Pressing Ctrl and dragging the element also copies it.

   Link Here: Pastes a copy of the element as a child of the target element and retains a link to the original element. Changes made to the original element affect the linked element. Pressing Shift+Ctrl and dragging the element also creates a linked copy.

3.5 Renaming Service Model Elements

Use caution when renaming service model elements. In particular, if there are any explicit element name matching rules defined for SCM, SLA, element children, behavior models, and so on, and a matched element is renamed, you must manually update the matching rules. Otherwise, the renamed element is not selected when the matching rules are executed.

For example, assume a behavior model and custom property pages use a rule that matches a service model element named *ABC*. If you then rename *ABC* to *123*, modify the behavior model and apply the original match rule, element *123* is not updated. You must modify the match rule to find element *123*.

To rename a service model element:

1. Right-click a service model element, then select *Rename*.
2. In the Rename dialog box, specify the new name, then click *Finish*.

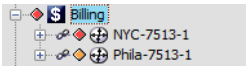
3.6 Deleting Elements and Linked Relationships

Service model elements and their linked relationships can be removed using the right-click *Element* menu. The three types of elements found in the *Service Models* hierarchy as follows:

- Native elements (elements that were created directly under a service model)
- Elements linked from the *Elements* hierarchy (cannot be deleted)
- Elements linked from within the *Locations* or *Service Models* hierarchy

Table 3-2 provides linked element examples:

Table 3-2 *Linked Element Examples*

Link Representation	Description
	Linked elements are identified by a linked symbol in the Explorer pane. This includes any elements created using the *Paste Link* option.
	Linked elements that do not contribute to state are identified by a broken link symbol in the Explorer pane. This is only available for a linked location.

Removing service model elements involves different options, depending on the element type and whether the link relationship or the element should be deleted.

- Section 3.6.1, "Removing a Native Element," on page 35
- Section 3.6.2, "Removing a Linked Element," on page 35
- Section 3.6.3, "Removing All Occurrences of an Element," on page 36

3.6.1 Removing a Native Element

1 In the Explorer pane, right-click the element, then select *Remove*.

A confirmation dialog box lists the parent from which the selected element is removed.

WARNING: Deleting an element also deletes any links to it that have been created elsewhere in the *Service Models* hierarchy.

2 Click *Yes*.

The element is removed along with any links to it that exist elsewhere.

3.6.2 Removing a Linked Element

1 In the Explorer pane, right-click the linked element (displays with a linked icon as shown above), then select *Remove*.

The element is removed from only this section of the *Service Model* hierarchy. The original element and other links to the element remain intact.

3.6.3 Removing All Occurrences of an Element

To remove all occurrences of an element, including all links to it in the hierarchy:

1. In the Explorer pane, right-click a linked element, then select *Remove All Occurrences*.
   A confirmation dialog box lists all the parent service models to which the element is linked.
2. Click *Yes*.
   The element and all its linked references are removed.

4 Defining Classes, Behavior Models, and Property Pages

Classes, behavior models, and property pages define attributes and relationships for enterprise resources in the *Service Models* hierarchy. Enterprise resources include any component or aspect of a business units, such as IT infrastructure components, personnel, or events and transactions.

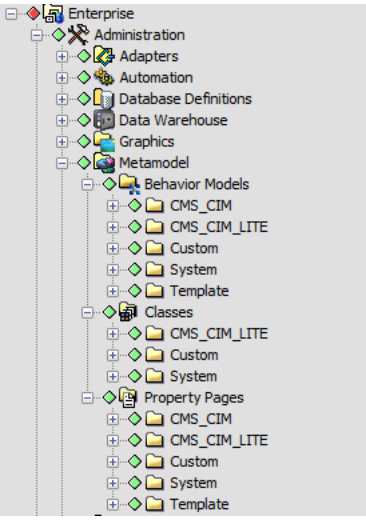
This section explains how to define classes, behavior models and custom properties using the Operations Center software. You can assign these components to service model elements.

- Section 4.1, “About Custom Classes, Behavior Models, and Property Pages,” on page 38
- Section 4.2, “Creating and Deleting Property Pages and Properties,” on page 41
- Section 4.3, “Creating and Deleting Behavior Models,” on page 65
- Section 4.4, “Creating and Deleting Classes,” on page 70
- Section 4.5, “Editing Classes, Behavior Models, Property Pages, and Properties,” on page 81
- Section 4.6, “Security on Metamodel Elements,” on page 85
- Section 4.7, “Creating Custom Tooltips,” on page 85
- Section 4.8, “Using Classes and Properties to Enhance Finding Elements,” on page 88

4.1 About Custom Classes, Behavior Models, and Property Pages

Custom classes, behavior models and property pages are used to define the enterprise resources that are used to create a service model. The *Metamodel* hierarchy, located under *Administration*, allows administrators to define new classes, behavior models, and property pages. *Property* pages are associated with elements through behavior models via class or directly to behavior models via match expressions.

Figure 4-1 *Metamodel Hierarchy*



The *Metamodel* hierarchy includes the following main features:

Table 4-1 *Metamodel Features*

Name	Default Subfolders	Description
Behavior Models	*System* *Template*	*System* and *Template* folders contain system-defined behavior models that can be modified. Behavior models under *System* folders can be modified, but they cannot be renamed or deleted. Define additional behavior models to map one or many property pages to elements based on class or other criteria defined by match expressions. Behavior models expand any service model element to display a list of property pages defined for that element. Behavior models do not inherit property pages from other behavior model definitions. Additional folders can be created to accommodate behavior models for your environment.

Name	Default Subfolders	Description
Classes	*System* *Custom*	The *System* folder contains system-defined classes that cannot be modified or deleted. The *Custom* folder contains classes that you create or that are converted from a prior version of Operations Center. Classes are presented in a hierarchical structure and subclasses inherit class properties from their parent class. Custom classes and subclasses can be created in any folder, including the *System* folder. Class names are not required to be unique in the *Metamodel* hierarchy. Use the fully qualified class name when selecting a class.
Property Pages	*System* *Template*	The *System* folder contains system-defined property pages that cannot be modified or deleted. The *Template* folder contains system-defined and custom property pages that can be modified and deleted. Define a set of property pages and their data fields. Expand any property page element to show a list of properties defined for that page. Property pages are assigned to elements using behavior models.

- Section 4.1.1, "Metamodel Icons," on page 39
- Section 4.1.2, "Metamodel Buttons," on page 40
- Section 4.1.3, "Maintaining the Metamodel Hierarchy," on page 40

4.1.1 Metamodel Icons

Table 4-2 describes the icons available on dialogs used to create or maintain classes, behavior models, or property pages. The icons that display depend on the dialog box from which they are selected.

Table 4-2 *Metamodel Definition List: Icons*

Icon Name	Icon	Description
Create		Opens the Create Behavior Model, Create Class, Create Property Page, or Add Match dialog box to allow the user to create a definition.
Browse		Browses a list of all behavior models, classes, properties, or property pages to add to a definition. This creates an explicit match to class name for the behavior model.
Edit		Opens the Edit Class, Edit Match, Edit Model, or Edit Property Page dialog box to view and edit definitions.
Copy		Copies a selected property, property page, or matching rule.
Delete		Removes a matching rule or the relationship to a model definition, property definition, or property page definition. Removing a relationship to a definition does not delete the actual model, property, or property page from the system.

4.1.2 Metamodel Buttons

The following describe the buttons available on dialogs used to create and maintain classes, behavior models, and property pages. The specific buttons that display depend on the dialog box open.

- **Back:** Opens the previous metamodel element.
- **Create:** Creates the metamodel element.
- **Forward:** Opens the next dialog box in a series of dialogs.
- **Finish:** Completes a task, such as creating a class and closes the dialog box.
- **Close:** Closes the dialog box without saving any changes on the current dialog box.
- **Apply:** Saves changes made on the open dialog box.
- **Apply All:** Saves all changes made on all dialogs opened.

4.1.3 Maintaining the Metamodel Hierarchy

The *Metamodel* hierarchy provides a structure that contains classes, behavior models, and property pages. It allows new folders to be created and elements to be copied and pasted from one folder to another within the associated structure.

Classes, behavior models, and property pages contain two default types of definitions:

- **System:** Product standard metamodel objects appear under the *System* folders. They can be copied, but cannot be modified or deleted. Custom metamodel objects created under the *System* folder can be modified or deleted.
- **Template:** Product standard template metamodel objects appear under the *Templates* folders. They can be copied, modified and deleted. Custom metamodel objects created under the *Template* folder can be modified or deleted.

To maintain the metamodel hierarchy, review the following sections:

- "Adding Folders in the Metamodel Hierarchy" on page 40
- "Deleting a Folder in the Metamodel Hierarchy" on page 41
- "Copying Behavior Models and Properties" on page 41
- "Renaming a Class, Behavior Model, or Property Page" on page 41

Adding Folders in the Metamodel Hierarchy

1. In the Explorer pane, expand *Administration* > *Metamodel*.
2. To add a new folder, right-click *Classes*, *Behavior Models*, or *Property Pages*, or the folder where you want to add a new folder, then select *Add Folder* to open its dialog box.

   Folders can be added to any existing folder including those you created.
3. Specify the name of the folder in the *Enter a Name for the Folder* field, then click *OK*.

   The folder is added to the hierarchy.

Deleting a Folder in the Metamodel Hierarchy

1. To delete a folder, right-click the folder, then select *Delete* to display a confirmation dialog box.
2. Click *Yes* to delete the folder.

   The folder is removed from the hierarchy.

   All contents of a folder and all subfolders are deleted when a folder is deleted.

Copying Behavior Models and Properties

To copy behavior models and properties within the *Metamodel* hierarchy:

1. In the Explorer pane, expand *Administration* > *Metamodel*, then select either *Behavior Models* or *Property Pages*.
2. Right-click a behavior model or property page, then select *Copy*.
3. Right-click the destination folder, then select *Paste*.

   The element displays under the folder.

Renaming a Class, Behavior Model, or Property Page

Deleting or renaming a class does not change the original class name already associated with service model elements. This protects an administrator from making mistakes and losing stored historical data that is tied to the element's full DName, which includes the class and service element name.

WARNING: Think carefully before renaming an established class. In particular, if there are any explicit class name matching rules defined for SCM, SLA, element children, behavior models, and so on, and a class is renamed, you must manually update associated elements and any matching rules. Otherwise, the elements formerly associated with the renamed class are not selected when the matching rules are executed.

To rename a class, behavior model or property page:

1. In the Explorer pane, expand *Administration* > *Metamodel*.
2. Navigate to the class, behavior model, or property page you want to rename.
3. Right-click the class, behavior model, or property page you want to rename, then select *Rename* to open its dialog box.
4. Specify the new name in the *Enter New Name* field.
5. Click *OK*.

   The class, behavior model, or property page displays with the new name.

4.2 Creating and Deleting Property Pages and Properties

Property pages define a list of properties available as a tab in the Properties dialog box for the elements defined by matching rules in behavior models. Property pages act as a library of definitions that can be linked to one or more behavior models.

A specific type of property page called the SLA Metric Computed property page defines properties for the SLA Metric Catalog and is automatically associated with every element with the class SLA Metric Class that belongs to the behavior model called SLA Metric Model. The SLA Metric Computed property page is designed to generate an SQL query that retrieves custom properties for an element

based on data in an external data source. Properties can be added to SLA Metric Computed property page but the existing properties cannot be edited or deleted and the page itself cannot be deleted. For more information about the SLA Metric Catalog, see the *Operations Center 5.5 Service Level Agreement Guide*.

To manage properties, review the following sections:

- Section 4.2.1, "Creating a Property Page," on page 42
- Section 4.2.2, "Deleting Property Pages," on page 43
- Section 4.2.3, "Defining Properties for a Property Page," on page 43
- Section 4.2.4, "Working with a Multiple Choice Property," on page 52
- Section 4.2.5, "Working with HTML Pages or Text," on page 58
- Section 4.2.6, "Configuring Macro Expressions," on page 61

4.2.1 Creating a Property Page

1. In the Explorer pane, expand *Metamodel* > *Property Pages*, then select the folder where the new property page should be added.
2. Right-click the folder where you want to create a property page, then select *Create Property Page* to open its dialog box.
3. Specify the name of the property page in the *Name* field.
4. Specify the name used for the property page in the Properties dialog box in the *Display Name* field.
5. (Optional) To specify a folder to be used in the Properties dialog box, specify folderName|pageName.

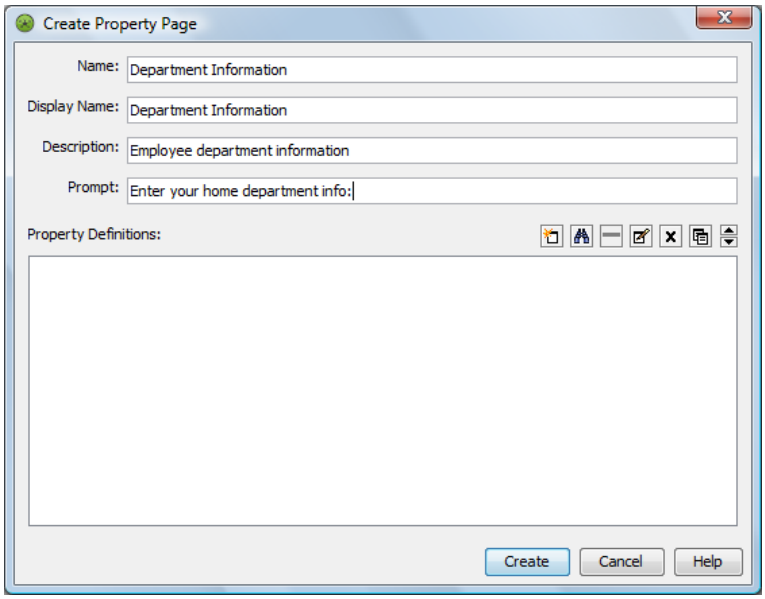
   For example, Boston|Headquarters|Contacts shows a Contacts property page under a *Boston/Headquarters Hierarchical* list in the Properties dialog box for associated elements.
6. Enter a description for the property page in the *Description* field.
7. (Optional) In the *Prompt* field, enter the text to display at the top of the Properties dialog box before property fields to provide additional information to the user or prompt a user for the information to be given.

   If null, no prompt displays before the property fields.
8. Define properties as required.

   For more information, see "Defining Properties for a Property Page" on page 43.

9 Click *Create* to create the property page.

A new element displays in the *Property Pages* hierarchy for the page definition:



4.2.2 Deleting Property Pages

1 Right-click the property page you want to delete, then select *Delete Page* to display a confirmation dialog box.

2 Click *OK* to delete the property page.

The property page is deleted from the *Metamodel* hierarchy.

Properties related to a property page are not deleted when a property page is deleted.

4.2.3 Defining Properties for a Property Page

One or more properties can be defined for a property page. Properties are defined when creating the property page.

- "Understanding Property Types" on page 44
- "Creating a Property" on page 47
- "Adding Separator Lines to Group Sets of Properties in the Property Page" on page 50
- "Defining Validation or Computational Rules for a Property" on page 51

Understanding Property Types

Table 4-3 describes the different types of custom properties. The property types are listed in alphabetical order.

Table 4-3 *Property Types*

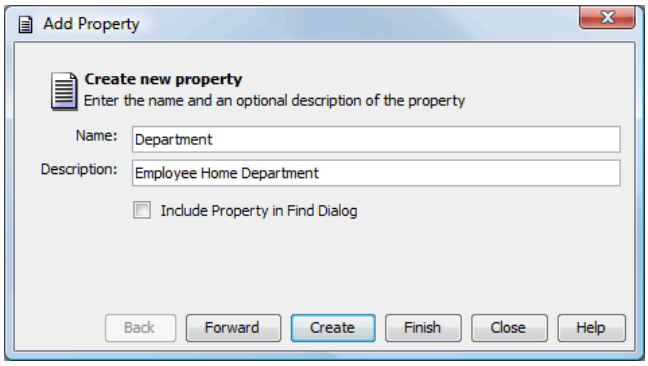
Property Types	Mask	Attribute Options	Automatic Validation Rules
Choice (combo box)	N/A	◆ **Fixed Choice:** Define the combo box list options. ◆ **Dynamic Choice:** Combo box options are derived from the element display names of children of a selected element.	Validation rules can be specified optionally, if the combo box is designated as editable. For examples, see "Working with a Multiple Choice Property" on page 52.
Computed Property	N/A	◆ Advanced rules only	
Computed Property (2 fields)	N/A	◆ Property1 ◆ Property2 ◆ Function (+,–,/,*,%)	◆ Verify property1 and 2 are number types ◆ Calculate using mathematical operators , which can be used to compare two properties(property1 * property2)
Currency	$####.##	◆ Default Value ◆ Input Mask ◆ Number Range ◆ Required Property	◆ None ◆ Mask enforces data entry on the input component ◆ Value range sets the number range rule ◆ Required property sets text length nonzero rule (ignore mask special characters)
Date	MM/DD/YY (and 12 other allowable)	◆ Default Value ◆ Input Mask (only for input and property page display) ◆ Required Property	◆ None ◆ Mask enforces data entry on the input component ◆ Required property sets text length nonzero rule (ignore mask special characters)
Duration/Elapsed Time (computed property)	N/A	◆ Property1 ◆ Property2	◆ Verify property1 and 2 are date types ◆ Calculate date difference (Property2–Property1)

Property Types	Mask	Attribute Options	Automatic Validation Rules
Email Address	None	◆ Default Value ◆ Email Address ◆ Required Property ◆ Minimum and Maximum text length	◆ None ◆ Required property sets text length nonzero rule ◆ Email Address sets Email address regular expression (variabletext@ variabletext.variabletext) ◆ Minimum and Maximum text length sets text length range regular expression (ignore mask special characters)
Function	N/A	◆ Required Property ◆ Property1 ◆ Property2 ◆ Function (+,–,/,*,%)	◆ Verify property1 and 2 are number types ◆ Calculate using function (property1 and property2)
HTML Page	N/A	◆ Width ◆ Height ◆ Open Links in External Browser	◆ None ◆ For examples, see “Working with HTML Pages or Text” on page 58.
HTML Text	N/A	◆ Width ◆ Height ◆ Open Links in External Browser	◆ None ◆ For examples, see “Working with HTML Pages or Text” on page 58.
ID	###-##-####	◆ Default Value ◆ Input Mask ◆ Required Property	◆ None ◆ Mask enforces data entry on the input component ◆ Required property sets text length nonzero rule (ignore mask special characters)
IP Address	###.###.###.###	◆ Default Value ◆ Input Mask ◆ Required Property	◆ None ◆ Mask enforces data entry on the input component ◆ Required property sets text length nonzero rule (ignore mask special characters)
Number	N/A	◆ Default Value ◆ Required Property ◆ Number Range ◆ Decimal Places	◆ A Number regular expression rule to enforce the allowable decimal places ◆ Required property sets text length nonzero rule ◆ Value range sets the number range rule

Property Types	Mask	Attribute Options	Automatic Validation Rules
Text	None	◆ Default Value ◆ Input Mask ◆ Required Property ◆ Minimum and maximum text length	◆ None ◆ Mask enforces data entry on the input component ◆ Required property sets text length nonzero rule (ignore mask special characters) ◆ Minimum and Maximum length set text length range rule (ignore mask special characters)
Text Area	N/A	◆ Default Value ◆ Required Property ◆ Minimum and maximum text length	◆ Required property sets text length nonzero rule ◆ Minimum and maximum length set text length range rule (ignore mask special characters)
Telephone	(###)###-####	◆ Default Value ◆ Input Mask ◆ Required Property	◆ None ◆ Mask enforces data entry on the input component ◆ Required property sets text length nonzero rule (ignore mask special characters)
URL	None	◆ Default Value ◆ URL Format ◆ Required Property ◆ Minimum and Maximum text length	◆ URL Format sets regular expression to check for ftp:// varialbletext or http/s:// variabletext ◆ Required property sets text length nonzero rule ◆ Minimum and Maximum text length sets text length range regular expression (ignore mask special characters) ◆ The URL displays only in the Property page; it does not display as a column in the *Alarms* view.
Zip Code	#####-####	◆ Default Value ◆ Input Mask ◆ Required Property ◆ Minimum and Maximum text length	◆ None ◆ Mask enforces data entry on the input component ◆ Required property sets text length nonzero rule (ignore mask special characters) ◆ Minimum and Maximum text length sets text length range regular expression (ignore mask special characters)

Creating a Property

1 From the Create Property Page dialog box, click Create Property to open the Add Property dialog box:



2 Specify a name and description for the property in the *Name* and *Description* fields.

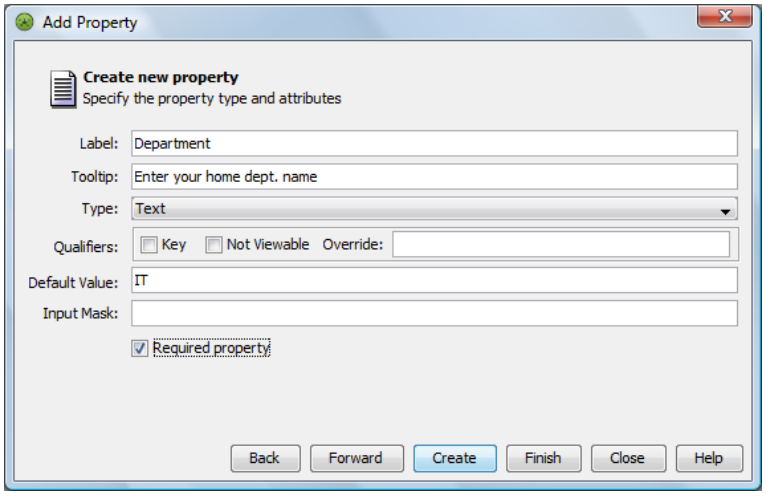
To make the property available as search criteria when finding elements, select the *Include Property in Find Dialog* check box. For more information, see "Using Classes and Properties to Enhance Finding Elements" on page 88.

Property names cannot include the space character or the following special characters:

< > # % " ' { } ( ) [ ] | \ ^ `)

3 Click *Forward*.

The dialog box to define the property type opens:



4 Specify a name to display for the property in the *Label* field.

5 Click the *Type* drop-down list, then select the type of property.

Select from the list described in "Understanding Property Types" on page 44.

6 (Optional) Do any of the following to set property options in the *Qualifiers* section:

- To indicate that this property is a key of a CIM class, select the *Key* check box.

  This option is not used by Operations Center but can be referenced by user scripts or applications.

- To hide this property on the property page, select the *Not Viewable* check box.
- To allow this property to override another property in a parent class, specify the property name to override in the *Override* field.

7 (Optional) Do one of the following:

- Specify a default value for the property in the *Default Value* field.

  The default property is not entered automatically on an element's property page. It is necessary to edit the property by placing the cursor in the property field.
- To enter the Operations Center default value in the property, click *Set to Default*.

8 Specify the appropriate symbols or edit the default symbols to define a formatting and validation rule for the property value in the *Input Mask* field as defined in the following table:

Symbol	Description
#	Any valid number.
'	Escape character, used to escape any of the special formatting characters.
U	Any character. All lowercase letters are mapped to upper case.
L	Any character. All upper case letters are mapped to lower case.
A	Any character or number.
?	Any character.
*	Anything.
#H	Any hex character (0–9, a–f or A–F).

When a mask is defined, you are automatically specifying the exact number of characters expected for the property value.

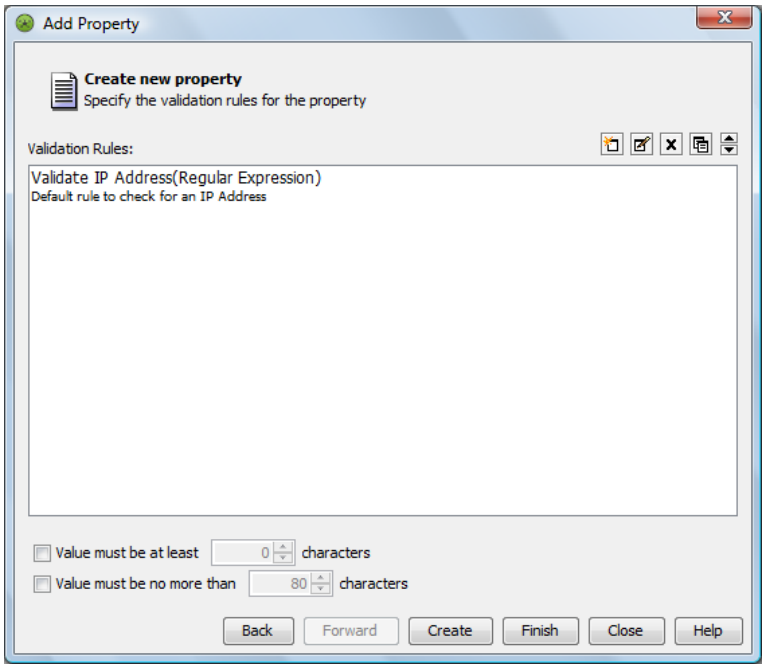
Leave the *Mask* field empty if you do not want to use a mask for the property value.

9 To make the property a required field, click the *Required Property* check box.

10 Click *Forward*.

The dialog box to specify validation rules opens. Validation options are based on the type selected.

In the following example, the rule verifies a valid IP address is entered:



11 Define validation and computational rules as required in the *Validation and Computational Rules* list.

Rules are applied to the property value in the order they are listed in the *Validation and Computational Rules* list. Use the Up and Down arrows to sort the rules.

For more information, see "Defining Validation or Computational Rules for a Property" on page 51.

12 To create a rule based on another rule, click (*Duplicate*), select the rule, then click (*Edit*) to specify new criteria.

13 To edit a rule, select the rule, then click (*Edit*) to open the Edit Rule wizard.

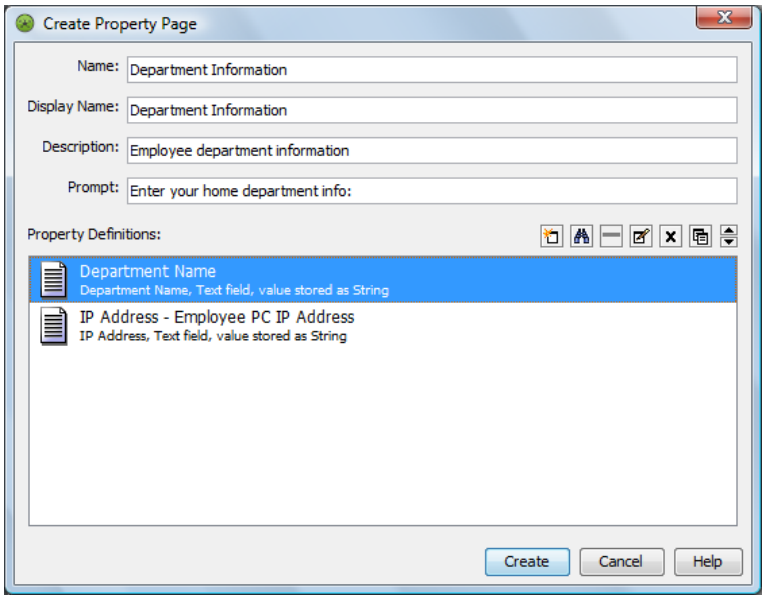
For more information, see "Defining Validation or Computational Rules for a Property" on page 51.

14 To delete a rule, select the rule, then click (*Delete*).

The rule is removed from the list.

15 Click *Finish* to save the property page.

The property displays in the *Property Definitions* section of the Create Property dialog box:



Adding Separator Lines to Group Sets of Properties in the Property Page

- "Adding a Separator Between Two Properties" on page 50
- "Adding a Separator to an Existing Property Page" on page 50

Adding a Separator Between Two Properties

1 In the Create Property Page dialog box, select a property in the *Property Definitions* section.

2 Click (*Add Separator*).

A line displays after the property selected in the previous step.

Adding a Separator to an Existing Property Page

1 Right-click a Property page in the Explorer pane, then select *Properties* to open the Status property page.

2 Click Property Page in the left pane to update the page.

3 Select a property in the *Property Definitions* section.

4 Click (*Add Separator*).

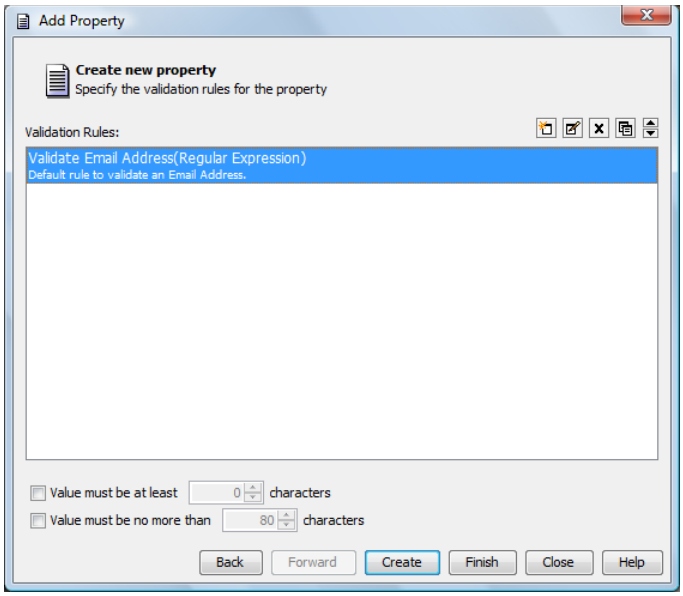
A line displays after the property selected in the previous step.

Defining Validation or Computational Rules for a Property

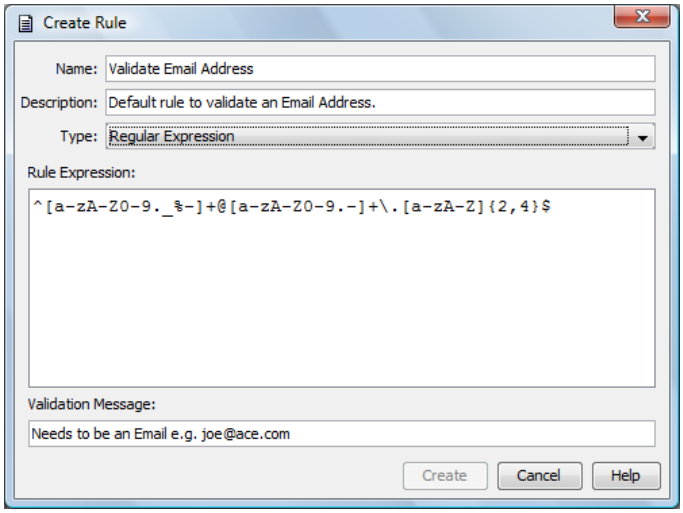
Each property definition can have one or more validation rules but only one or computational rules. These can be used to further restrict acceptable values for a property or to perform a mathematical calculation to produce the value.

To define validation rules for a property:

1 From the Edit Property dialog box, click *Forward* until you reach the Validation Rules dialog box:



2 Click Create New Validation Rule to create a new validation rule:



3 Click the *Type* drop-down list, then select the syntax type for the validation rule:

Regular Expression: Specifies how to look for a specified pattern in text and how what processing occurs when a pattern match is found.

Macro Expression: Specifies how to look for a pattern in text using a statement that contains some variable parameter information and what processing occurs when a pattern match is found.

Script Expression Specifies how to validate using a script expression.

Date Range: Validates the selected date matches the rule parameters (*Is Before*, *After*, or *Between Specific Dates*).

After you select the syntax type, the dialog box options update to match it.

4 Complete the rule parameters as defined in the following table:

Validation Rule Types	Input Types	Rule Parameters
Date Range	Text Date	◆ Lower Bound ◆ Upper Bound ◆ Date Format (used if Text input type)
Regular Expression	Any	◆ Regular Expression ◆ Invalid Value Message
Script Expression	Any	◆ Script Expression ◆ Invalid Value Message
Macro Expression (Velocity)	Any	◆ Macro Expression ◆ Invalid Value Message

5 Enter the error message to display when the data does not meet the validation rule in the *Validation Message* field.

6 Click *Create*.

4.2.4 Working with a Multiple Choice Property

The Choice property type enables creating a multiple choice property that displays in a service model element property page. Only one value can be selected from the *Property* drop-down list. When configuring the Choice property, it is necessary to specify the values to display in the drop-down list.

When configuring the drop-down list, specify the following:

- "Specifying a Default Value" on page 53

  Displays as the property value until a different value is selected or entered.

- "Configuring Fixed vs. Dynamic Values" on page 53

  Fixed values that display in the drop-down list, or Dynamic values that are generated using the names of child elements related to a specified parent element. These values are generated when an element's property page displays.

- "Storing and Displaying Different Property Values" on page 56

  The Display Name, the part of a property value that displays in the property page, and the Stored Value, the part of a value that is stored in the configuration storage database.

- "Requiring a Property Value" on page 57

  Whether a value for the property is required (Required property).

- "User Entry of a Multiple Choice Property Value" on page 58

  Whether a property value can be entered directly in an element's property page (*Editable*).

Specifying a Default Value

When configuring a property, specifying a default value is optional. This value is used for the property until it is changed.

A default value can provide guidance for an expected property value. For example, specify <operating system> as the default value for a property named Operating System. For a multiple Choice property type, the default value should be a choice that displays in the drop-down list. Note that default values satisfy the Required property restriction when a user saves a property page.

To set a default value, enter a string value in the *Default Value* field in the Add Property dialog box.

Configuring Fixed vs. Dynamic Values

When defining the values for the multiple Choice property type, you can select one of the following:

- **Fixed:** Specify a fixed list of values that display in the drop-down list for the property in the property page. Fixed is the default setting, and One, Two, Three, and Four are the default choices that appear in the drop-down list unless you change them.

  Use the *Fixed* option when there is a predefined list of possible values for a property that rarely changes, such as computer operating systems, usernames, or corporate locations. For example, assume there is a finite list of lab locations for your company. Create a multiple Choice property named Lab Name that includes all of the lab locations.
- **Dynamic:** Use *Browse* to select a parent element whose first-level child element names display in the drop-down list for the property. The dynamic list ensures that all new child elements related to the selected parent automatically display in the drop-down list.

  Each multiple choice value appears in a separate line in the drop-down list and only one value can be selected.

To configure fixed values or a drop-down list:

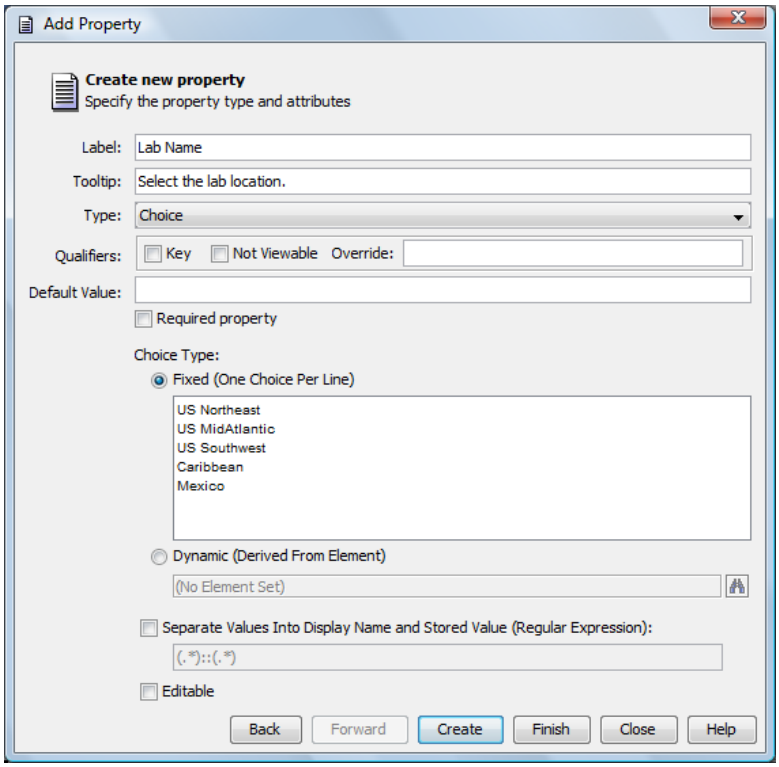
- "Creating a List of Fixed Values for the Property" on page 54
- "Generating a Drop-down List" on page 54

Creating a List of Fixed Values for the Property

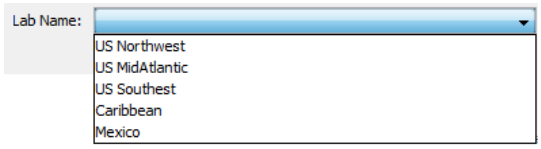
1. In the Add Property dialog box, select the *Fixed* radio button.
2. Replace the default `One, Two, Three, Four` text with the values that should appear in a drop-down list for this property.

   When you edit the property, these fixed values display as choices in the drop-down list.
3. Click *Create*.

   The following illustrates configuring a new Multiple Choice property containing a fixed list of values:



   The following illustrates selecting from a fixed list of values in a Service Model element property page:



Generating a Drop-down List

If the list of values for a property can change over time, consider defining a Dynamic multiple Choice property based on element names that exist in the *Operations Center* hierarchy. As an example of an element with dynamic children, assume the business processes associated with the *Supply Chain* element are expected to change over a few months as new procedures are put in place. You could

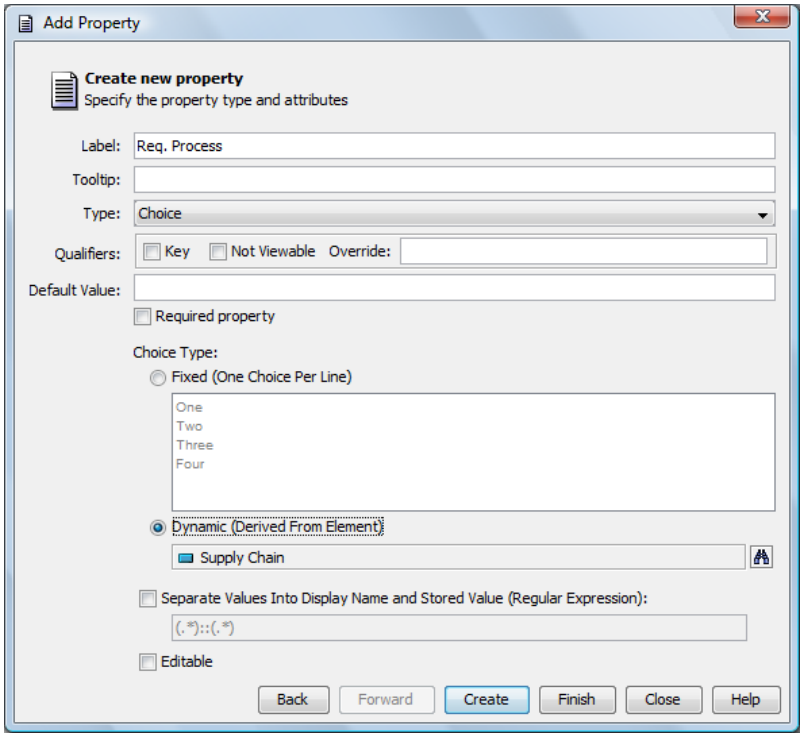
define the property Req. Process to display all the child elements of the *Supply Chain* element. When a new process is added, it automatically displays in the drop-down list for the multiple choice property.

To generate a drop-down list using the children of a specific parent element:

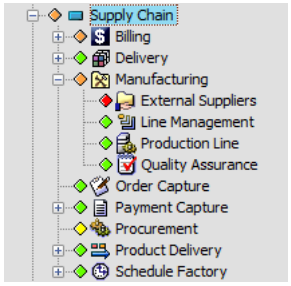
1. In the Add Property dialog box, select the *Dynamic (Derived from Element)* radio button.
2. Click *Browse*, then select a parent element (such as Supply Chain).

   When users edit the property, the selected element's children display as choices in the drop-down list.

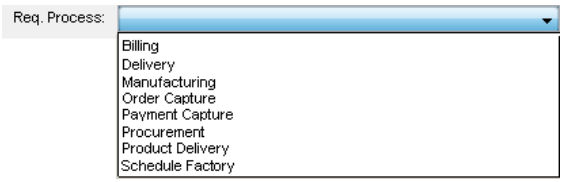
   The following illustrates configuring a new Multiple Choice property containing a dynamic list of the child elements of Supply Chain:



The following illustrates viewing child elements under Supply Chain:



The following illustrates selecting from a dynamic list of values generated from the child elements under Supply Chain in a Service Model element property page:



Storing and Displaying Different Property Values

Typically, the value displayed for a property is the same as the value stored in the configuration storage database. For each value defined for a multiple Choice property, you can define a displayed value and a corresponding different value that is stored in the configuration storage database.

- "Displaying One Set of Values" on page 56
- "Defining the Values Displayed" on page 56

Displaying One Set of Values

To display one set of values in the property page and store a different set of values in the configuration storage database:

1. In the Add Property dialog box, select the *Separate Values into Display Name and Stored Value (Regular Expression)* check box.
2. To specify the displayed value and the stored value, enter an expression in the field.

   The expression should be in the form: (.*)symbol(.*). The data before the symbol displays in the property page and the data after the symbol is the stored value. The default is (.*)::(.*) where (.*) indicates all data and :: is the separator. This option is available when using both fixed values and dynamic values.

   As an example, a list of system usernames might change over time, but the associated user IDs that are stored by the system remain constant. Assume Lisa Smith married and changed her name to Lisa Jones. The Username property value displayed in the property page might change from Lisa Smith to Lisa Jones, but her fixed, unique user ID, 123456, should remain the same.

Defining the Values Displayed

To define the values displayed in the property page and the stored values:

1. In the *Choice Type* section, under *Fixed*, specify a value consisting of two parts: 1) the display value and 2) the stored value.

   Separate the values using the default regular expression string of "::" (double colon string).

   Using the previous example, enter:

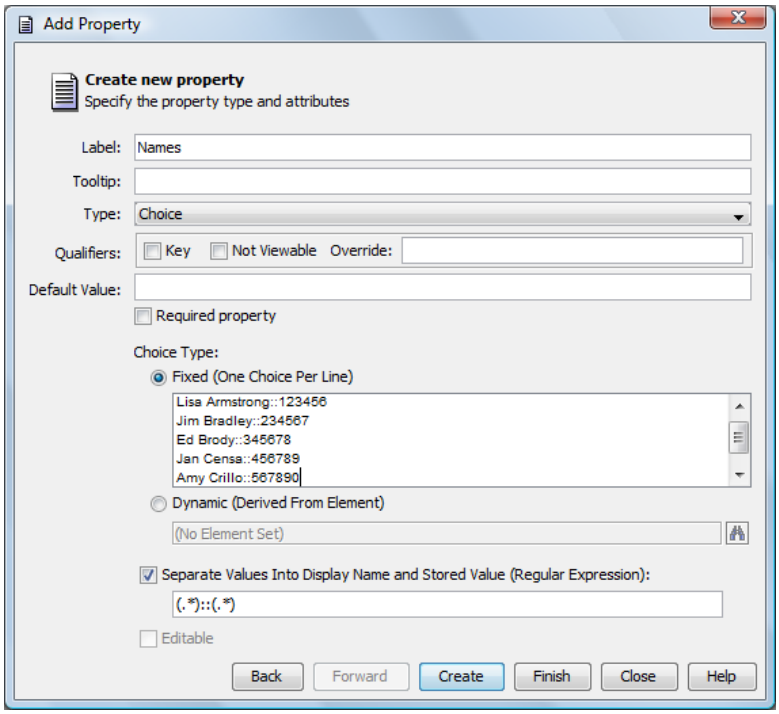
   ```
   Lisa Jones::123456
   ```

   Lisa Jones displays in the property's drop-down list, but the stored value is 123456.

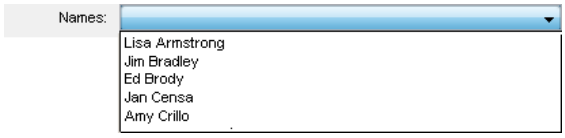
2 Select the *Separate Values into Display Name and Stored Value* check box.

This ensures only the display names are shown in the drop-down list. If this check box is deselected, both the display names and stored values are shown in the drop-down list.

The following illustrates configuring a new multiple Choice property to use a fixed value pair in which the values stored differ from those displayed in the property page:



The following illustrates selecting the values for a property that is configured to display one value in the property page and to store a different value in the configuration storage database:



Requiring a Property Value

When defining a multiple Choice property, decide whether the property is required, which requires selecting a property value prior to saving property value changes entered on the property page. Note that if a Default value is specified, it satisfies the *Required* property option.

To specify a required property:

1 In the Add Property dialog box, select the *Required Property* check box.

By default, the check box is not selected, so a property value is not required.

User Entry of a Multiple Choice Property Value

Another feature of a multiple Choice property is allowing users to directly enter a property value in the property page. This allows greater flexibility in determining a property value for an element, but it can have a negative impact on standardization of data.

To allow users to enter a value for a multiple Choice property:

1. In the Add Property dialog box, select the *Editable* check box.

   By default, this option is not selected, meaning users cannot enter property values; they must select a value from the drop-down list.

   The option to make the property editable is applicable for Fixed and Dynamic values. The exception is if you choose to display and store different values, then the property cannot be edited in the property page.

   When the Separate values into display name and stored value check box is selected, the *Editable* check box is not available.

4.2.5 Working with HTML Pages or Text

To create custom properties that display HTML pages or text, create two custom property pages. One page contains a HTML Page (or HTML Text) property type that contains a specific Web page URL (or HTML text). The other page contains a TextArea property that displays the Web page (or HTML text). Use a common property name to link the two pages. If you enter a different URL in the first property, the displayed Web page updates automatically in the second property.

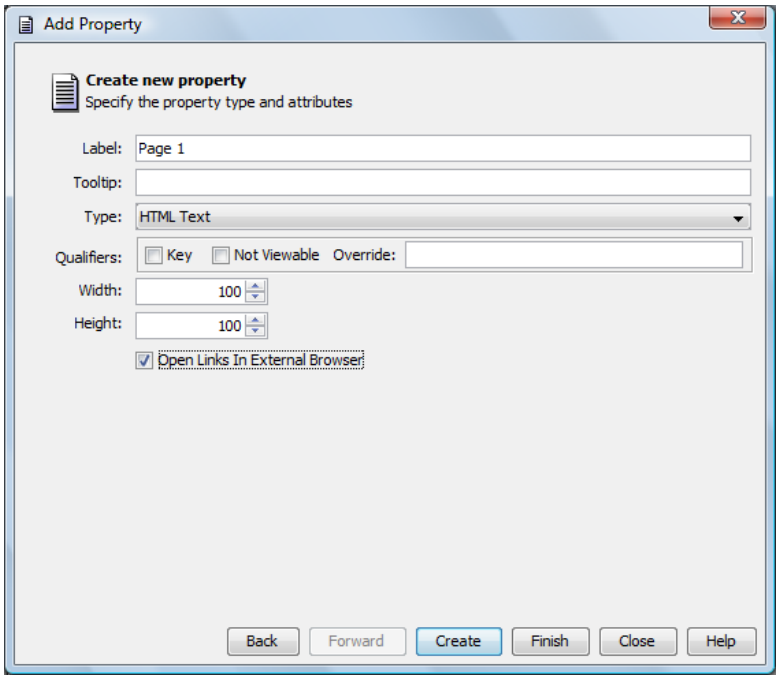
A subset of HTML is supported for this feature, so some HTML code might not be rendered in the page displayed using the HTML custom properties.

The following example steps through the process of creating two custom properties to display a Web page. Change the Web page content by changing the URL.

To display a Web page as a custom property:

1. Create a property page named `Web Page`.
2. In this page, create and name a property named Page1 that is the HTML Page type.

   If creating HTML text, select the *HTML Text* type.

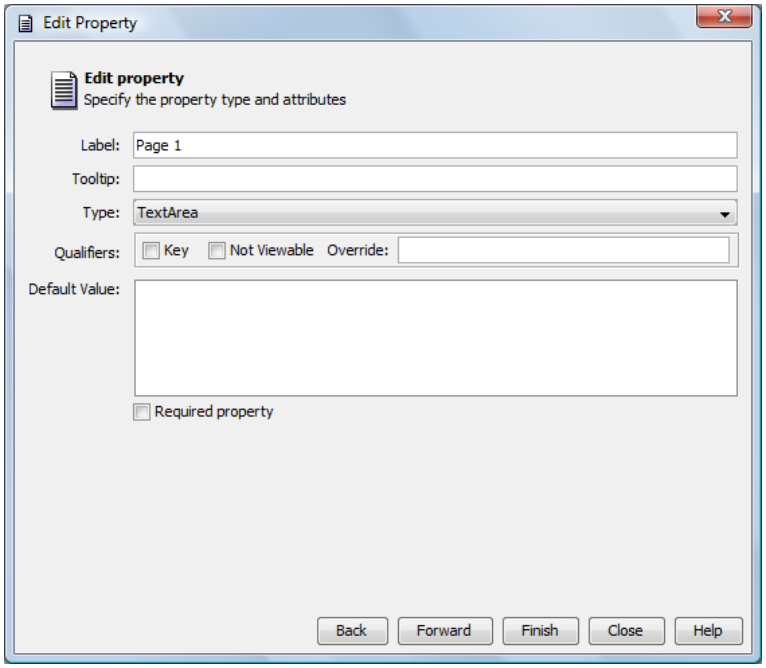


3 Specify the page height=500 and width=100.

4 Select *Open Links in External Browser*.

5 Copy and rename the Web Page property page to Web Page Edit.

6 Edit Web Page Edit and change the Page1 property type to TextArea:



7 To attach both new custom property pages to an element, right-click a service model element, then select *Manage Properties*.

8 Click the Browse icon, then select both *Web Page* and *Web Page Edit*.

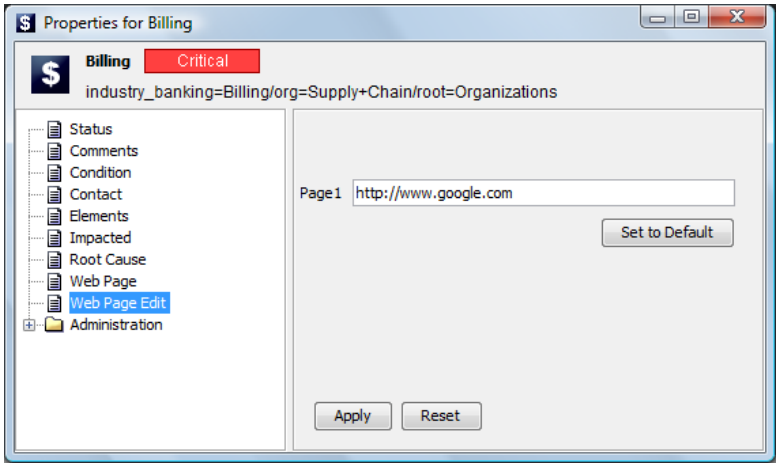
9 Click *Apply*.

10 Right-click the same service model element, then select *Properties*.

11 Click *Web Page Edit*.

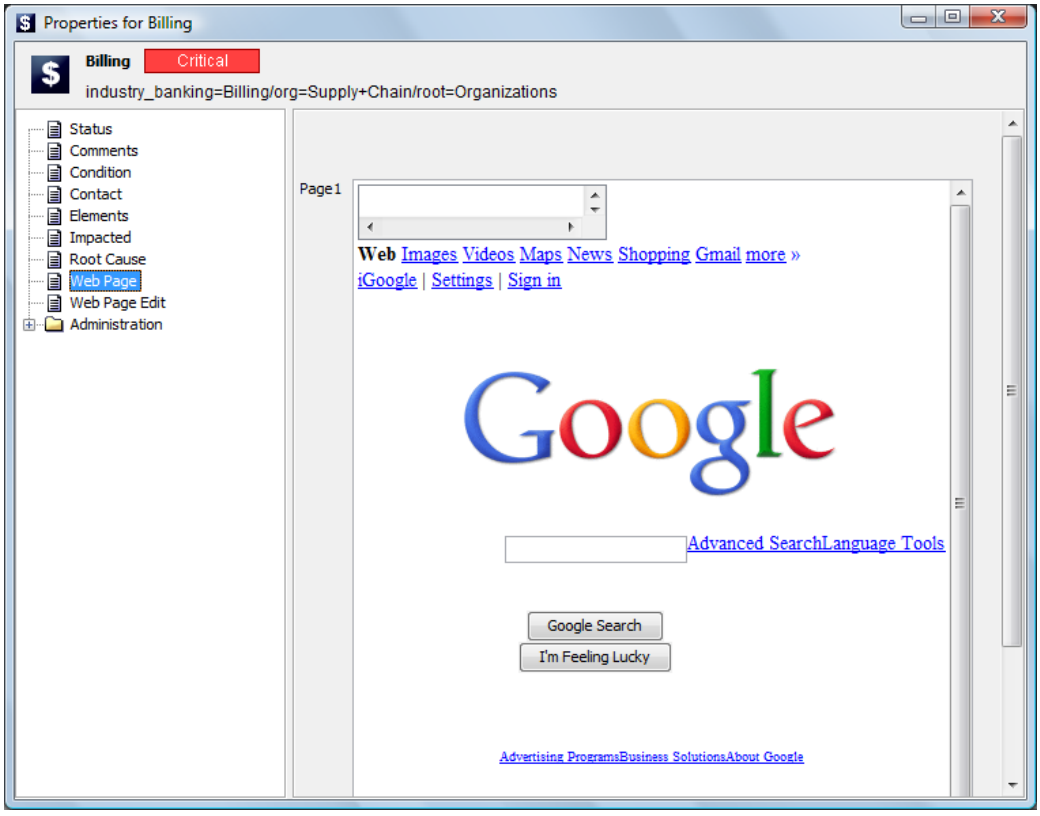
12 Enter the URL, and then click *Apply*.

Alternatively, if displaying HTML text, enter the HTML text here. It displays in the corresponding HTML Text type property:



13 To verify the setting, click the *Web Page* property.

For the example in the above illustration, the Google home page displays:



4.2.6 Configuring Macro Expressions

Operations Center macro expressions use Velocity, which is a third party package provided by the Apache Organization and bundled into Operations Center for macro expressions. Documentation for Velocity is available at http://jakarta.apache.org/velocity (http://jakarta.apache.org/velocity).

Macro expressions use the following syntax rules:

- No spaces are used in macro expressions.
- The notation for a variable consists of a leading "$" character following by a VTL identifier. A VTL identifier must start with an alphabetic character.
- Characters in a macro expression are limited to:
  - Alphabetic characters (a–z or A–Z)
  - Numeric characters (0–9)
  - Hyphen (–)
  - Underscore (_)

- "Example of a Macro Expression" on page 61
- "Using Script Expressions" on page 61

Example of a Macro Expression

The following is an example of a macro expression for a computed property that generates an e-mail address by concatenating the first initial of the ContactFirstName property with the ContactLastName property and uses the CompanyDomain property after the @ symbol:

```
#if( ${ContactFirstName} && ${ContactFirstName.length()} > 0 )
${ContactFirstName.substring(0,1).toLowerCase()}${ContactLastName.toLowerCase()}@$
{CompanyDomain}
#else
E-mail address not computed
#end
```

ContactFirstName and ContactLastName are both text properties. CompanyDomain is a text property such as NetIQ.com.

If no ContactFirstName is provided, the e-mail address displays the message: "E-mail address not computed."

Using Script Expressions

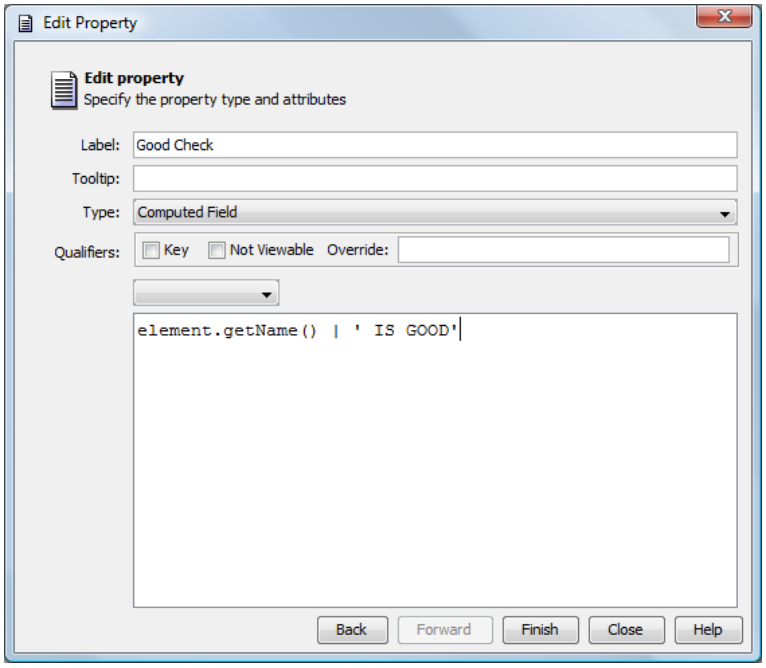
Script expressions can be used within macro expressions for validation rules and computed fields for custom properties. Also, predefined scripts can be saved and used in macro expressions. This is useful if you are more familiar with NOC Script than with macro expressions.

The following example creates a computed field script expression consisting of an element name concatenated with the text "IS GOOD!":

- "Creating the Custom Property" on page 62
- "Testing the Custom Property" on page 62
- "Creating the Validation Rule" on page 63
- "Testing the Validation Rule" on page 64

Creating the Custom Property

1 Create a custom property named Good Check.



2 Select *Computed Field* as the property type.

3 Select *Script Expression* from the drop-down list.

4 In the text area enter:

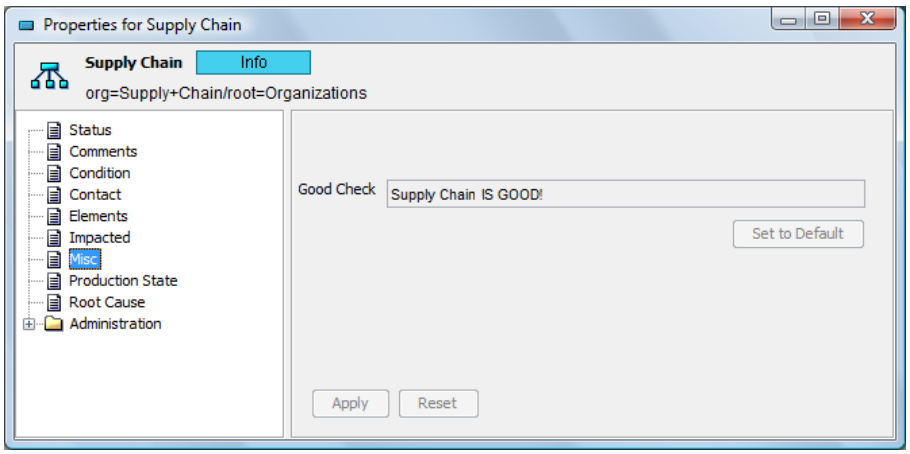
```
element.getName() + ' IS GOOD! '
```

5 Click *Create*.

Testing the Custom Property

1 Attach the custom property page to a service model element.

2 Open the custom property page and the *Good Check* field should display the element name and the words "IS GOOD!":

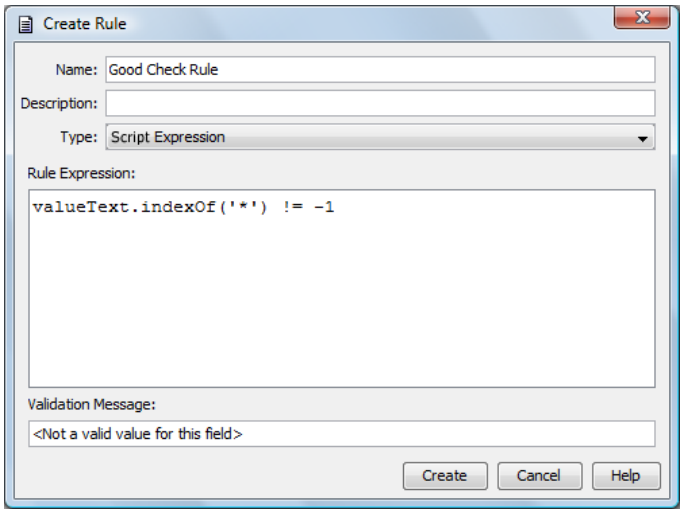


Creating the Validation Rule

Assume a property value is valid only if it contains an asterisk (*). Create a validation rule that uses a script expression to check the property value entered by a user.

To create a validation rule:

1. In the same custom property page as above, define a new property named Validation Check.
2. Select *Text* for the property type.
3. Click *Forward* to display the validation rules.



4. Create a validation rule, then select *Script Expression* for the validation rule type.

5 In the text area, enter:

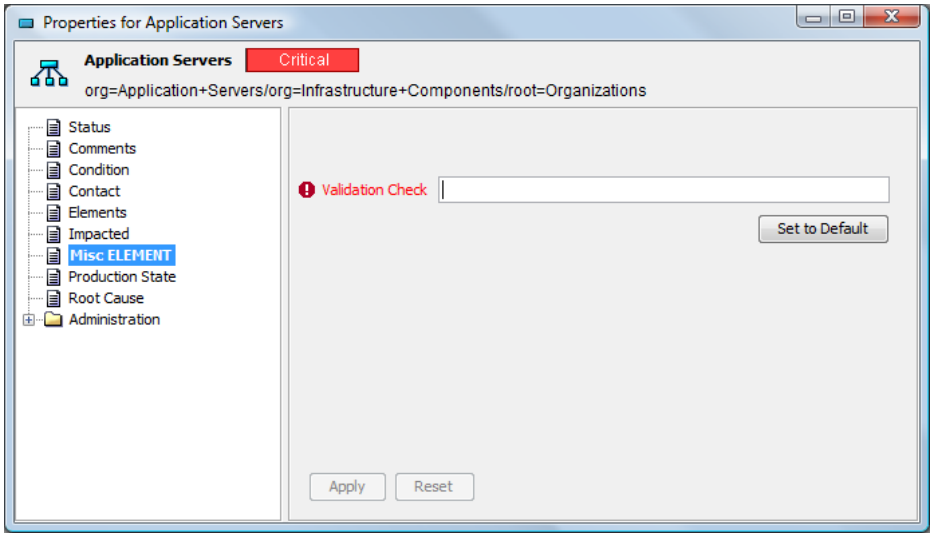
```
valueText.indexOf( '*' ) != -1
```

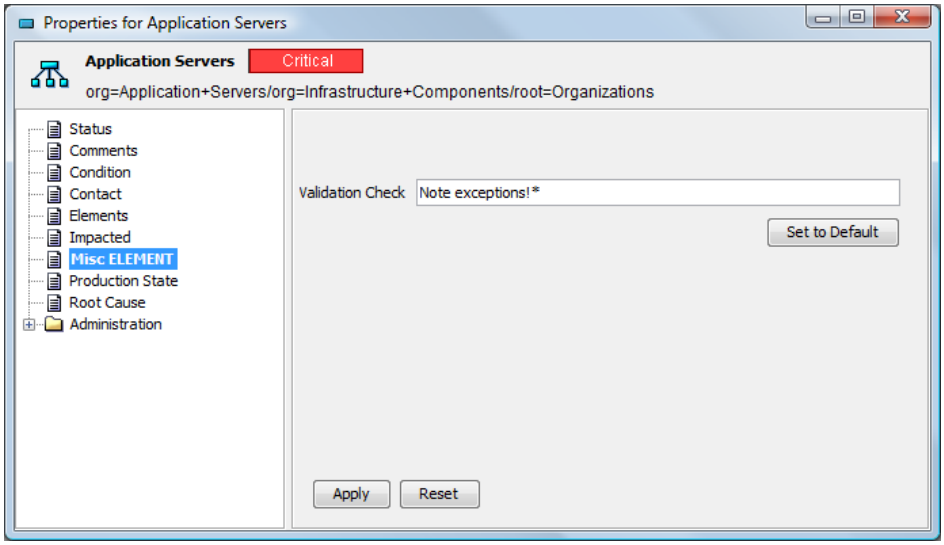
6 Click *Update* and save the new property.

Testing the Validation Rule

1 Apply the new custom property page to a service model element.

The *Validation Check* property is red and marked with an exclamation point until a valid value (containing an asterisk in this example) is entered.





4.3 Creating and Deleting Behavior Models

Behavior models provide a convenient way to organize custom property pages by business function and attach them to service model elements. For example, define a group of behavior models that represent various IT components required to operate business center branches. Attach custom property pages to these behavior models, then use behavior model matching rules to select elements. The behavior models (and the property pages) are attached to the selected elements.

Create behavior models and then attach custom property pages to them. Attach behavior models to selected elements based on element class or other matching rules.

- Section 4.3.1, "Creating a Behavior Model," on page 65
- Section 4.3.2, "Creating a Matching Rule," on page 65
- Section 4.3.3, "Deleting Behavior Models," on page 70

4.3.1 Creating a Behavior Model

1. In the Explorer pane, expand *Metamodel* > *Behavior Models*.
2. Right-click the *Behavior Models* folder (or a subfolder), then select *Create Behavior Model* to open its dialog box.
3. Specify the behavior model name and description in the fields.
4. To define matching rules for selecting elements to attach the behavior model, click New in the *Matching Rules* section to open the Match Rules wizard.

   For more information, see "Creating a Matching Rule" on page 65.
5. To attach a property page to a behavior model, do one of the following:
   - To select an existing property page for the behavior model, click Browse in the *Property Pages* section and navigate to the property page you want to attach.
   - To define a new property page, click Create a Property Page in the *Property Pages* section.

     For more information, see "Creating a Property Page" on page 42.
6. Click *Create*.

   The new behavior model name displays in the *Metamodel* hierarchy in the Explorer pane.

4.3.2 Creating a Matching Rule

The Match Rules wizard is used to create matching rules that determine which elements are assigned a specific behavior model (and the custom property pages assigned to that behavior model).

Define only one match rule type per behavior model, per element. For example, you can set only one class matcher to element *X*. You can set one class matcher to element *Y* and another to element *Z*, and so on. If multiple class matches are needed for the same element, select the *Match by Expression* option and write an LDAP expression that uses an OR statement.

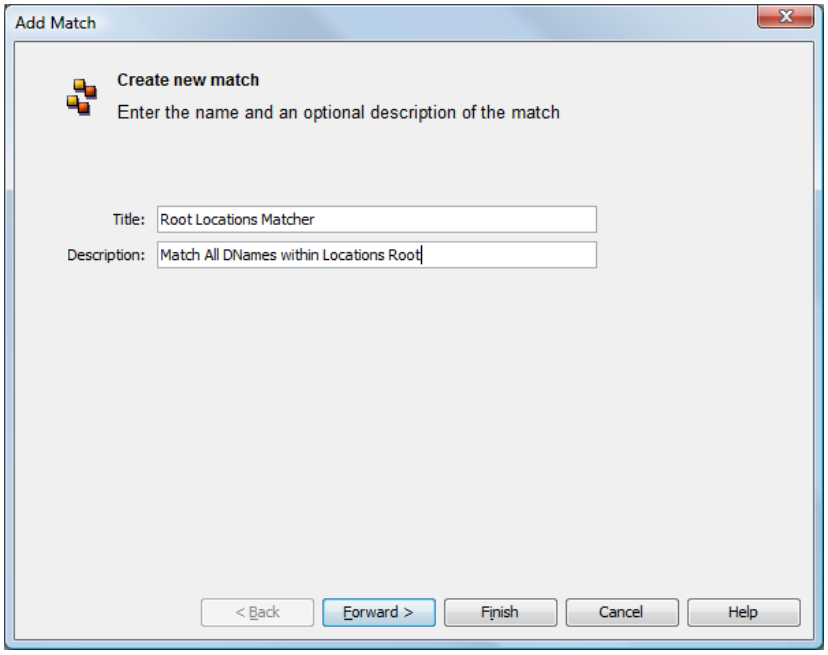
The last four sections are optional steps that are referenced from with the first section. To create matching rules, follow the instructions in the first section:

- "Creating a Matching Rule for a Behavior Model" on page 66
- "Matching by Element" on page 68
- "Matching by Expression" on page 69

- "Matching by Script" on page 69
- "Matching by Class" on page 70

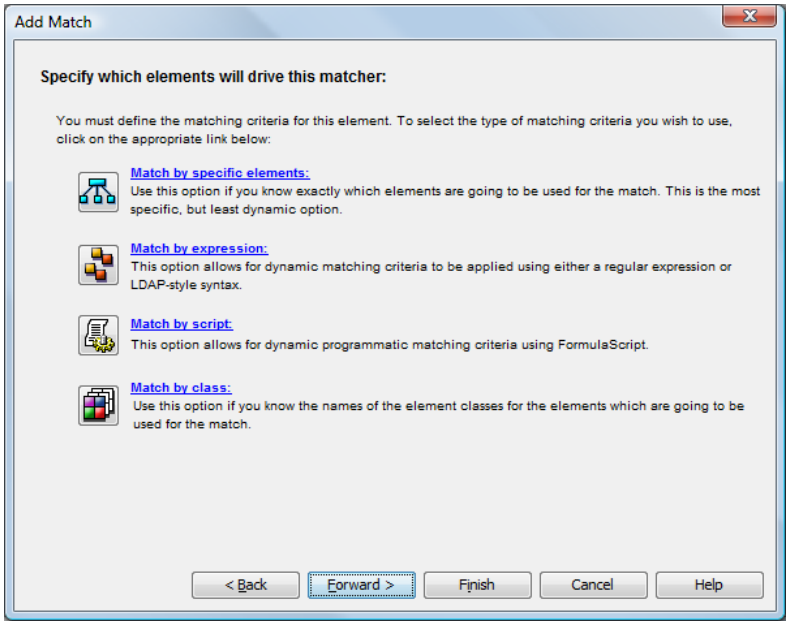
Creating a Matching Rule for a Behavior Model

1 From the Create Model dialog box (or the *Model* property page for an existing behavior model), click (*Create New*) in the *Element Matching Rules* section to open the Add Match dialog box:



2 Specify a name and description for the matching rule in the *Title* and *Description* fields, then click *Forward*.

3 To specify the matching criteria used to select elements to associate with this behavior model, select one of the following options:



The Add Match dialog box updates to include four tabs along the bottom that allow you to create a matching rule, which are described in the following table:

Icon	Description
	**Match by Specific Elements:** Allows you to search for and select an element for the match criteria. For instructions, see “Matching by Element” on page 68.
	**Match by Expression:** Selects elements based on a regular expression or LDAP-style syntax. The match is based on the element’s full DName. For instructions, see “Matching by Expression” on page 69.
	**Match by Script:** Selects elements based on a script written using the NOC Script language. The entire text in the window must evaluate to either True or False for a given object. Only administrators with programming experience should use this feature. It is reserved for situations that require statements that exceed the complexity of an element expression, or that require validation of a property other than an element’s DName. For instructions, see “Matching by Script” on page 69.
	**Match by Class:** Selects elements based on element class name. Results are based on the fully qualified class name, not just the base name. For instructions, see “Matching by Class” on page 70. It is not possible to use multiple matchers with the same starting element. Instead, use an Expression matcher with the LDAP syntax. For example, to match the classes city and state, use the following expression: (|(objectClass=city)(objectClass=state))

4 After completing a matching rule definition, do one of the following:

- Click *Forward* to create a matching rule to open the Create Rule dialog box.

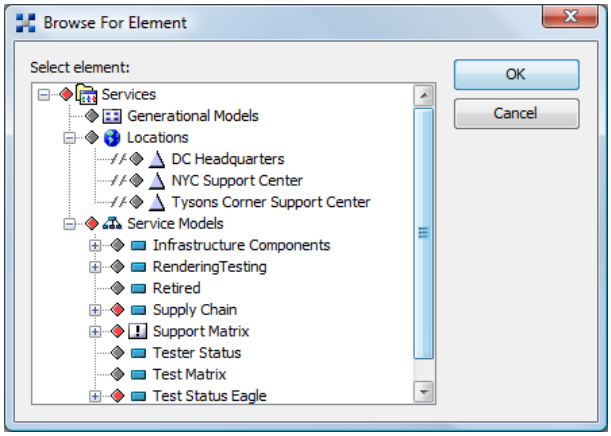
  The matching rule displays in the Create Model dialog box.
- Click *Finish* to save the matching rule.

Matching by Element

To assign the behavior model to specific elements:

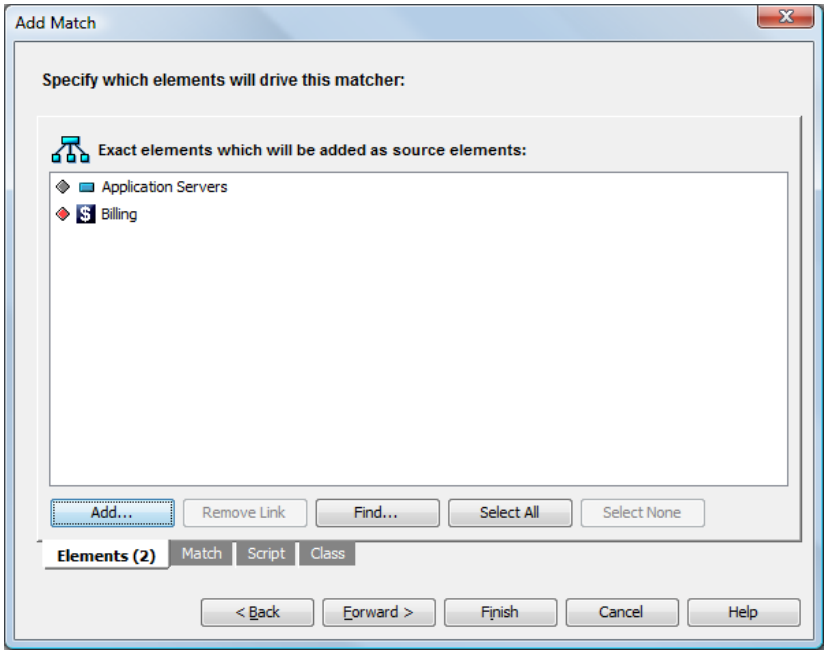
1 On the Add Match dialog box, click  (*Match by Specific Elements*).

2 If you know the location of the element to use as a match, click *Add* to open the Browse for Element dialog box.



3 Navigate to the element that you want to select, then click *OK*.

The selected elements display in the following dialog box:



4 If you do not know the exact element name or its location in the hierarchy, click *Find* to open its dialog box, then search for a specific element.

For more information on using this feature, see the *Monitoring Guide*.

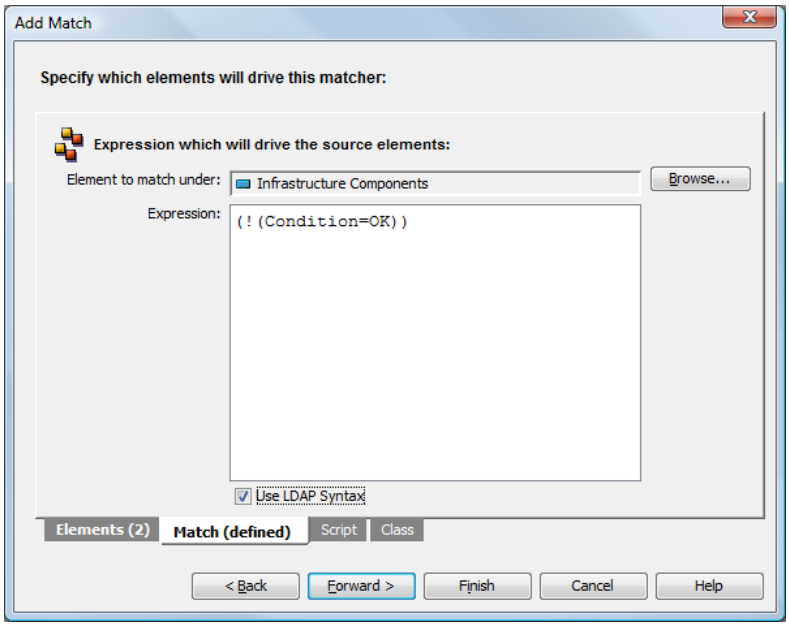
Matching by Expression

To use an expression to select matching elements:

1. On the Add Match dialog box, click  (*Match by Expression*).
2. To specify the parent element whose children should be used to test the matching expression, click *Browse* to open the Browse for Elements dialog box, navigate to the element, then click *OK*.

   The element displays in the *Element to Match Under* field.
3. Enter the expression in the *Expression* field.
4. To identify the expression as LDAP, select the *Use LDAP Syntax* check box.

   The following example uses and LDAP expression that selects elements with any condition except OK:



Matching by Script

To define a match by script:

1. On the Add Match dialog box, click  (*Match by Script*).
2. To specify the element to match, click *Browse* to open the Browse for Elements dialog box, navigate to the element you want to use, then click *OK*.

   The element displays in the *Element to Match Under* field.
3. Enter the script code in the field.

   The script language expected is NOC Script.

   The script must result in a True or False output for elements in the selected element branch.

Matching by Class

To define a match by class:

1. On the Add Match dialog box, click  (*Match by Class*).
2. To specify the parent element to match, click *Browse* to open the Browse for Elements dialog box, navigate to the appropriate parent element, then click *OK*.

   The element displays in the *Element to Match Under* field.
3. Specify the element name and/or class in the *Name* and *Class* fields.

   Use the fully qualified class name to accurately identify the exact element to use as a match.

   If you do not use the fully qualified class name, you can match against an element that has the same name, but is in a different class.

4.3.3 Deleting Behavior Models

Deleting a behavior model deletes the behavior model, but property pages and classes linked to a behavior model are not deleted. Properties related to property pages are not deleted when a behavior model is deleted.

To delete a behavior model:

1. In the Explorer pane, expand *Administration* > *Metamodel* > *Behavior Models*, then navigate to the element that you want to delete.
2. Right-click the behavior model, then select *Delete Model* to open a confirmation dialog box.
3. Click *Yes* to delete the behavior model.

   The behavior model is removed from the hierarchy.

4.4 Creating and Deleting Classes

Create classes that can be assigned to new elements defined in the *Service Models* hierarchy. Also attach behavior models to a class, to determine which property pages display for elements in the class.

The element class can be used to automatically correlate different types of elements. Define Class Relationship templates to generate relationships between elements of different classes and automatically associate element conditions and alarms. For more information, see "Defining Class Relationship Templates" on page 72.

Class names in the *Metamodel* hierarchy are not required to be unique if the classes are retained in different folders in the hierarchy. A fully qualified class name refer to a single specific instance of the class within the hierarchy (such as Classes > Custom > *router*), but a class of router refers to all instances of the class name router (such as Classes > Custom > *router* and Classes > MyClasses > *router*).

Review the following sections to manage classes. The fourth section provides examples of three use cases:

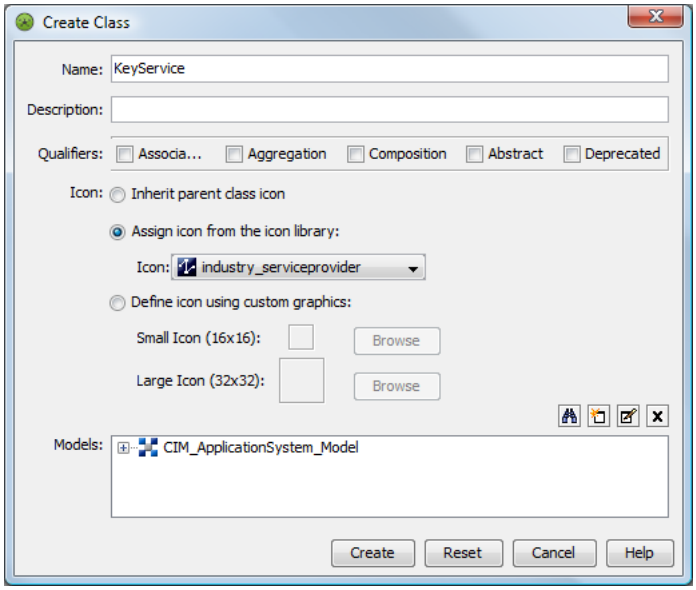
- Section 4.4.1, "Creating a Class," on page 71
- Section 4.4.2, "Deleting a Class," on page 72

- Section 4.4.3, "Defining Class Relationship Templates," on page 72
- Section 4.4.4, "Use Case Examples," on page 75

4.4.1 Creating a Class

To create a class:

1. In the Explorer pane, expand *Metamodel* > *Classes*.
2. Right-click the folder where the new class is created, then select *Create Class* to open the Create Class dialog:



3. Specify a name and description of the class in the *Name* and *Description* fields.
4. Select any options in the *Qualifiers* section to apply CIM model qualifier attributes as appropriate:

   Association: The class defines a relationship between two objects.

   Aggregation: The class defines a relationship between parent and child objects.

   Composition: The class defines a relationship where the child object is a part of the parent. In this case, the child can only have one parent.

   Abstract: The class serves as a base to define new classes.

   Deprecated: The class is obsolete and is only used for backwards support only.

   These options apply to the Operations Center Configuration Management System (CMS) only. For more information about CMS, see the *Operations Center 5.5 Configuration Management System (CMS) Guide*.

5. To assign an icon to the class, select one of the following options:

   Inherit parent class icon: Reuses the class' parent icon. This assumes the class is created as a child of another class. If the parent is not a class, but a folder, then the default formula_org icon is used.

   Assign icon from the icon library: Select this radio button, click the *Icon* drop-down list, then select the existing icon to display for the class.

Define icon using custom graphics: Select this radio button, *Small Icon* and *Large Icon*, click *Browse*, navigate to the icon, then click *Open*. The selected icon displays on the Create Class dialog box.

If the icon graphics are later edited or replaced, it is necessary to restart the server in order to display the updated icons.

6 To make the class available as search criteria when finding elements, select the *Include in Find* check box.

For more information, see "Using Classes and Properties to Enhance Finding Elements" on page 88.

7 Select one or more behavior model definitions to apply to elements of this class using one of the following steps:

- Click Browse to select one or more behavior model definitions.

  Behavior model definitions contain a list of property pages that show for elements of this class.

- Click Create a Class Model to create a behavior model to open the Create Model dialog box.

  For more information, see "Creating and Deleting Behavior Models" on page 65.

- Click Edit to display the behavior model and to edit it.

8 Click *Finish*.

The new class displays in the *Metamodel* hierarchy.

4.4.2 Deleting a Class

When a class is deleted, elements of the class revert to the default org or map class. Data associated with property pages is retained.

It is possible to associate a *Layout* view component with a class. All elements created using the class inherit the associated drawing, graphic, nodestyle or drawing template. For more information, see the *Operations Center 5.5 Custom Drawing and Layout Guide*.

To delete a class:

1 In the Explorer pane, expand *Administration* > *Metamodel* > *Classes*.

2 Right-click a class element, then select *Delete Class* to open a confirmation dialog box.

3 Click *Yes* to delete the class.

The class and all subclasses are removed from the *Metamodel* hierarchy.

4.4.3 Defining Class Relationship Templates

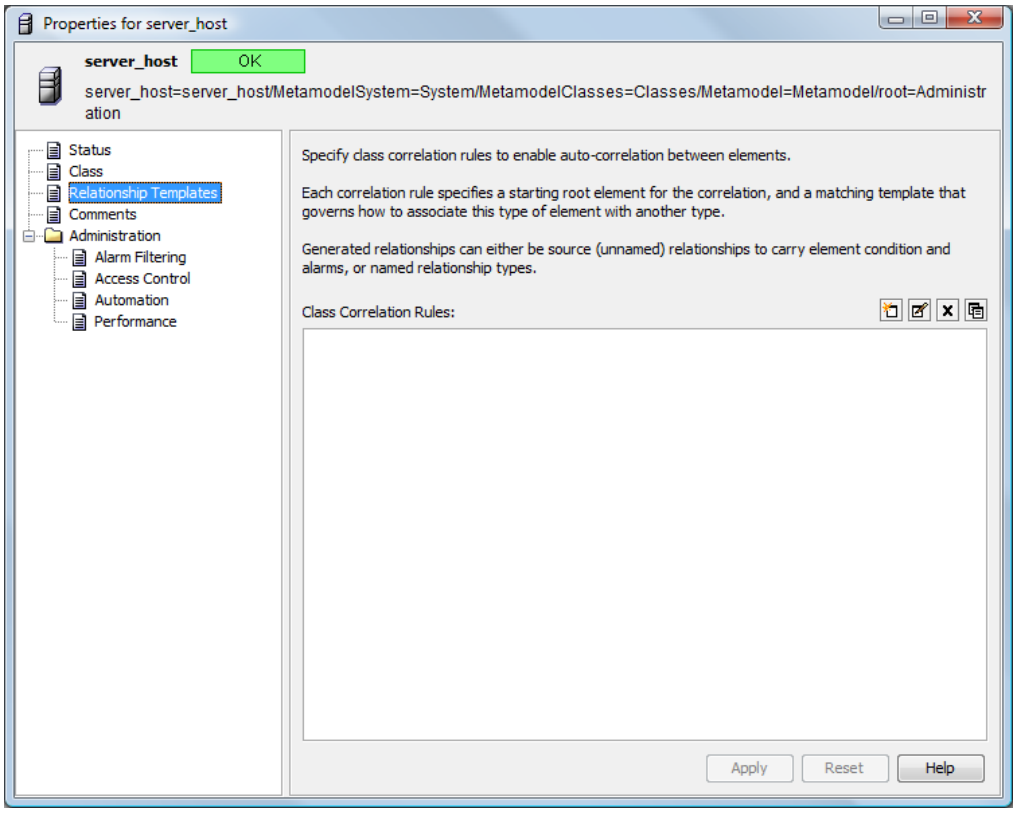
The element class can be used to automatically correlate different types of elements. Define Class Relationship templates to generate relationships between elements of different classes, and associate element conditions and alarms.

A Class Relationship template is one of several methods that can be used to correlate different types of elements. For descriptions of all available correlation methods, see Section 8.2, "Understanding Element Correlation Methods," on page 126.

To define a Class Relationship template:

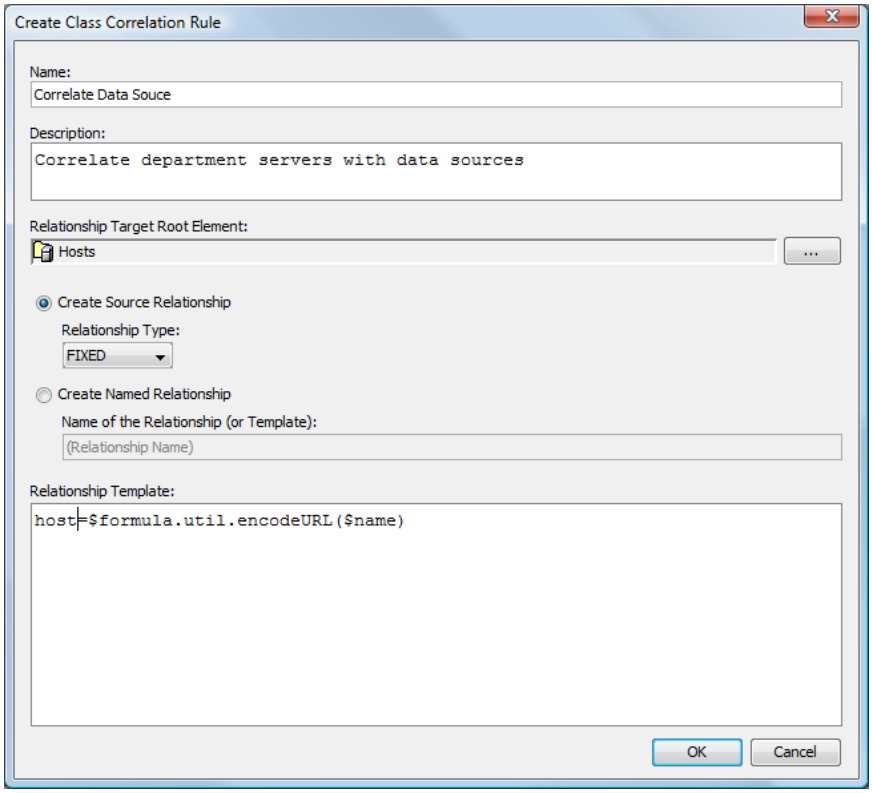
1. Right-click a class element, then select *Properties*.
2. In the left pane, click Relationship Templates.

   The right pane updates to display a list of class correlation rules used in the template. The list is empty if no rules have been created.



3 Click (*New*) to define a rule for correlating the selected element class with another class.

The Create Class Correlation Rule dialog box opens:



4 Define the following correlation rule settings:

Name: A name for the rule.

Description: A description of the rule's purpose or function.

Relationship Target Root Element: Identify a branch in the *Elements* hierarchy that should be automatically correlated.

Click *Select*, then select an element that serves a starting point (or root) for correlation with the selected class.

Relationship Type: Source or Named: If the relationship is between structural data and monitoring sources, select the *Create Source Relationship* radio button, then select a relationship type from the drop-down list (*FIXED*, *LDAP*, or *REGEX*).

Otherwise, select the *Create Named Relationship* radio button and specify a name for the relationship.

Relationship Template: Specify template parameters, usually the class name to correlate. Use the `${formula.util()}` macro expansion to encode a DName component with URL encoding syntax:

${formula.util.encodeURL($name)}

This is required for formatting a DName in the proper match output to escape spaces into '+' characters.

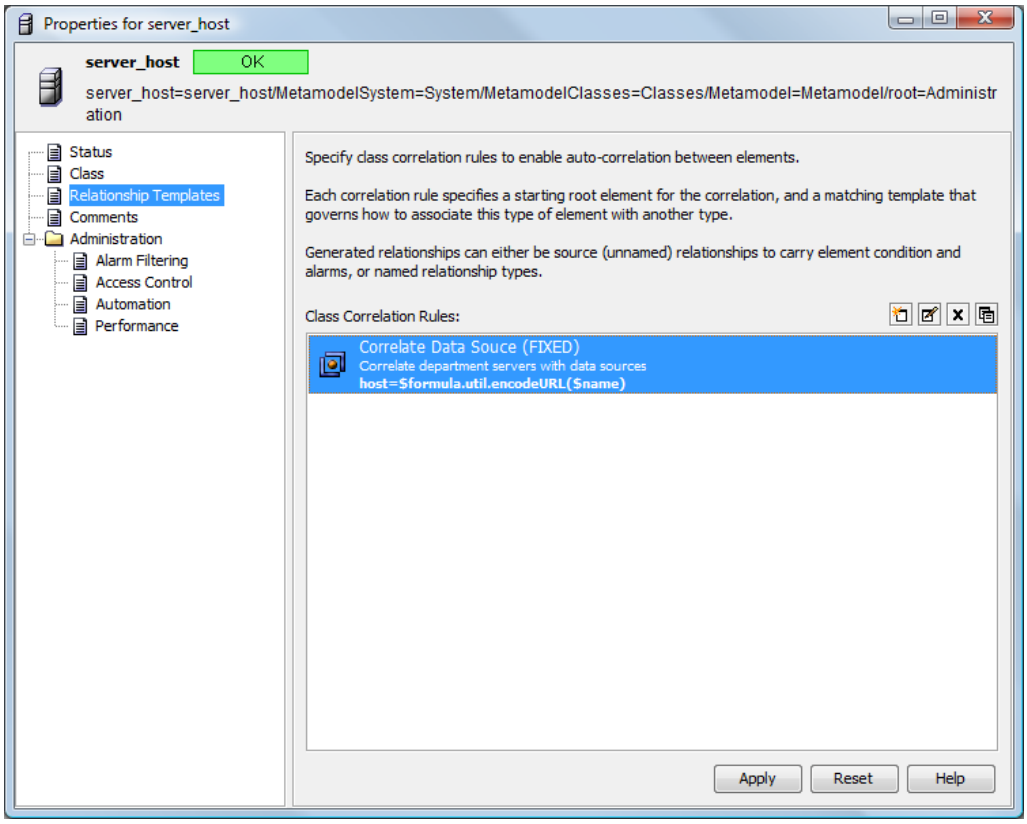
For use case examples, see "Use Case Examples" on page 75.

For additional syntax tips, see "Understanding Expression Template Parameters" on page 146.

5 Click *OK* to save changes on the Create Class Correlation Rule dialog box.

6 Click *Apply* to save changes on the Relationship Templates properties page.

This uses the correlation rule and automatically associates elements with different classes. For example:



4.4.4 Use Case Examples

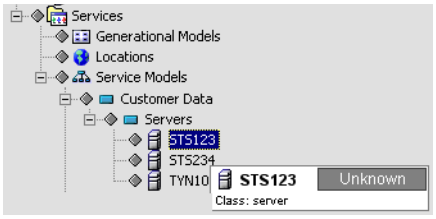
The following sections walk you through various use case examples:

- "Use Case 1: Correlation between Elements in Different Branches" on page 76
- "Use Case 2: BSM Correlation using Fixed and Variable Names" on page 78
- "Use Case 3: Named, Property-Based Relationships for CMDB" on page 80

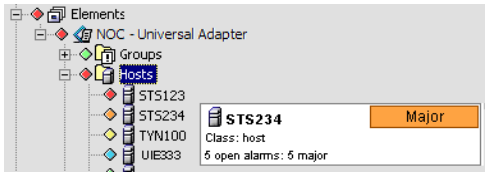
Use Case 1: Correlation between Elements in Different Branches

A simple use case automatically correlates elements created with a class named server, in the *Service Models* hierarchy, with like-named elements with a class named host in a branch of the *Elements* hierarchy:

Under the Servers element:

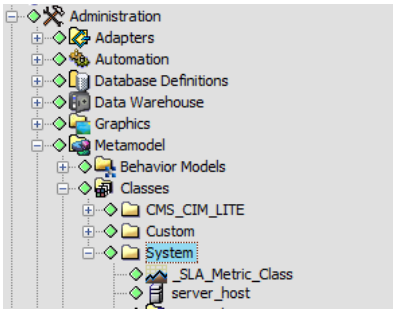


Under the Hosts element:



Example of creating a Class Relationship template for the server class:

Figure 4-2 *Hierarchy of Class Relationship*



The template correlation rule, shown in Figure 4-3, identifies the *Hosts* branch of the *Elements* hierarchy as the starting point for automatic element correlation. A FIXED Source relationship is specified.

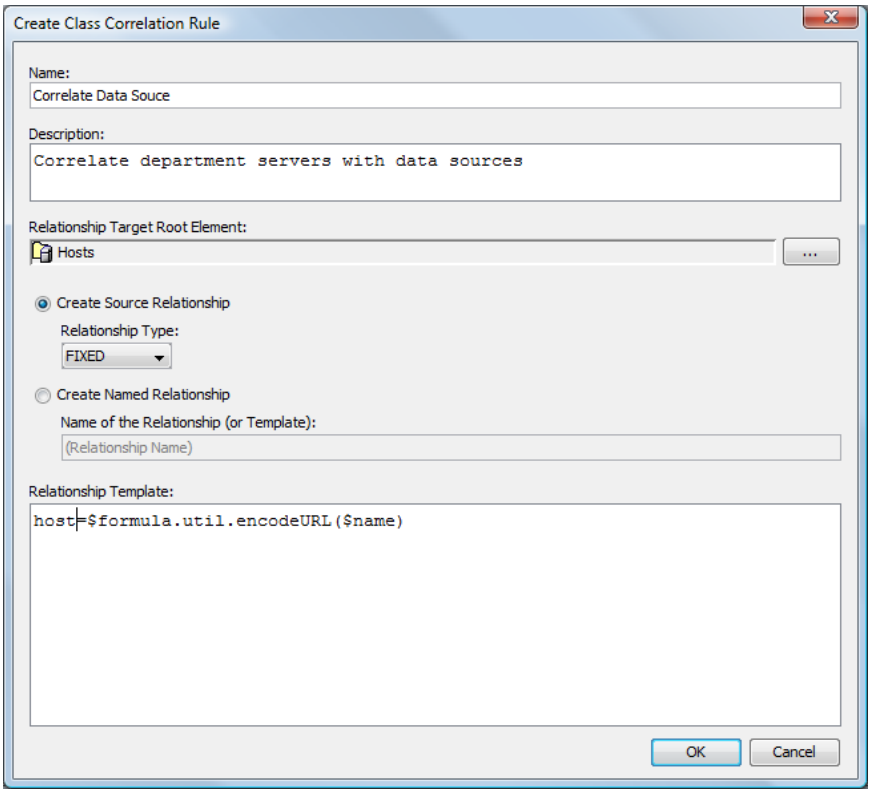
In the *Relationship Template* section, the specified expression generates the DName portion to correlate, using the `$formula.util.encodeURL()` function to ensure DName rules are followed. `host` is specified as this is the URL encoding for the host string. An alternative expression is:

```
$formula.util.encodeURL('host')=$formula.util.encodeURL($name)
```

This DName match template is similar to the Dynamic correlation, FIXED policy in SCM. For more information, see Section 8.3.6, “STEP 6: Select Element Correlation Options,” on page 145.

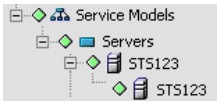
The following illustrates a correlation rule for a relationship template associated with the Server Class:

Figure 4-3 *Create Class Correlation Rule Dialog Box*



View the results in the element hierarchy. In this example, creating a service model element with the server class automatically generates a static relationship with an element that has the identical name but with the host class, in the *Hosts* branch of the *Elements* hierarchy. Note the element condition and all alarms are also correlated:

Figure 4-4 *Service Models Hierarchy*

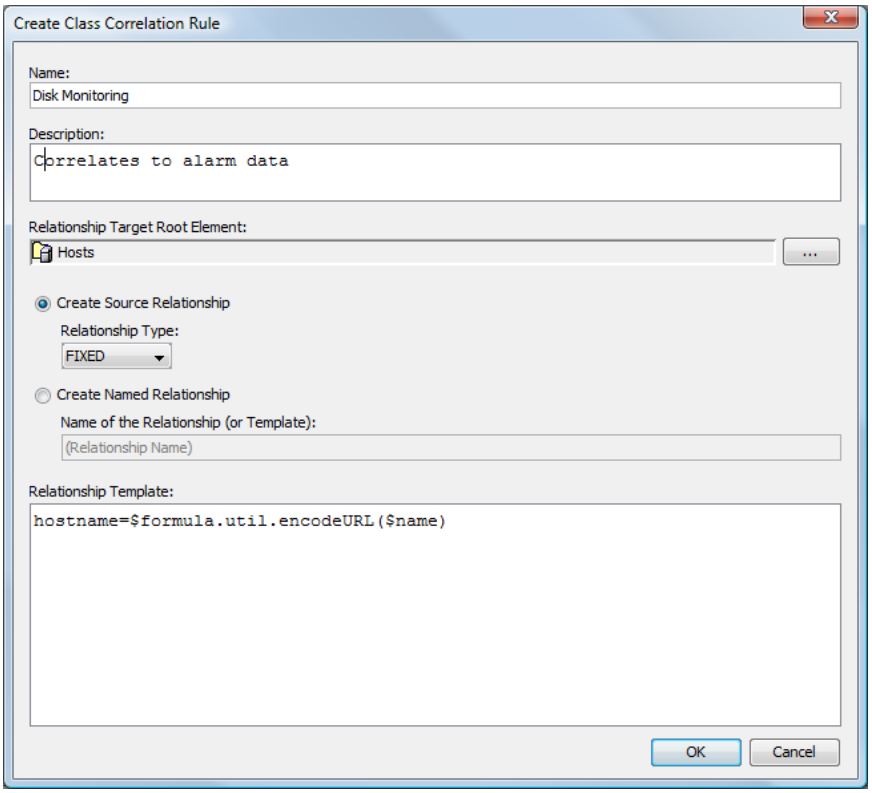


Use Case 2: BSM Correlation using Fixed and Variable Names

A common application of Class Relationship templates is BSM correlation. Assume that conditions and alarms from a Tivoli T/EC data source should be correlated with any element that has the class Model Disk Array. The Class Relationship template is created for the Model Disk Array class.

The correlation rule is shown in Figure 4-5:

Figure 4-5 *Create Class Correlation Rule Dialog Box*



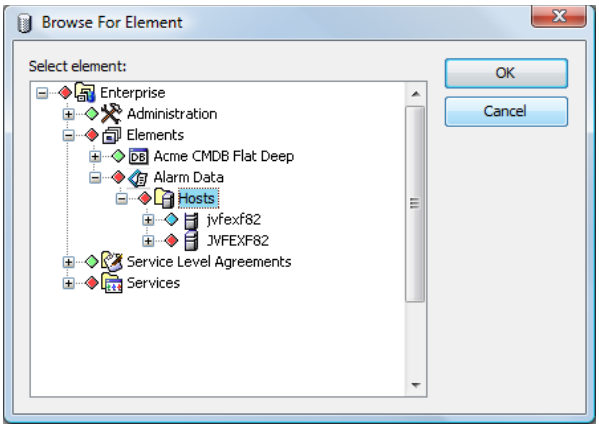
The correlation begins with the *Hosts* element and include all elements in this container whose names match those with a Model Disk Array class. A FIXED Source relationship is specified.

In the *Relationship Template* section, the specified expression generates the DName portion to correlate, using the `$formula.util.encodeURL()` function to ensure DName rules are followed. `hostname` is specified as this is the URL encoding for the hostname string. An alternative expression is:

```
$formula.util.encodeURL('hostname')=$formula.util.encodeURL($name)
```

Use an LDAP expression to allow for variability under the *Hosts* container. Figure 4-6 shows two elements with the same name but different capitalization:

Figure 4-6 *Browse for Element Dialog Box*

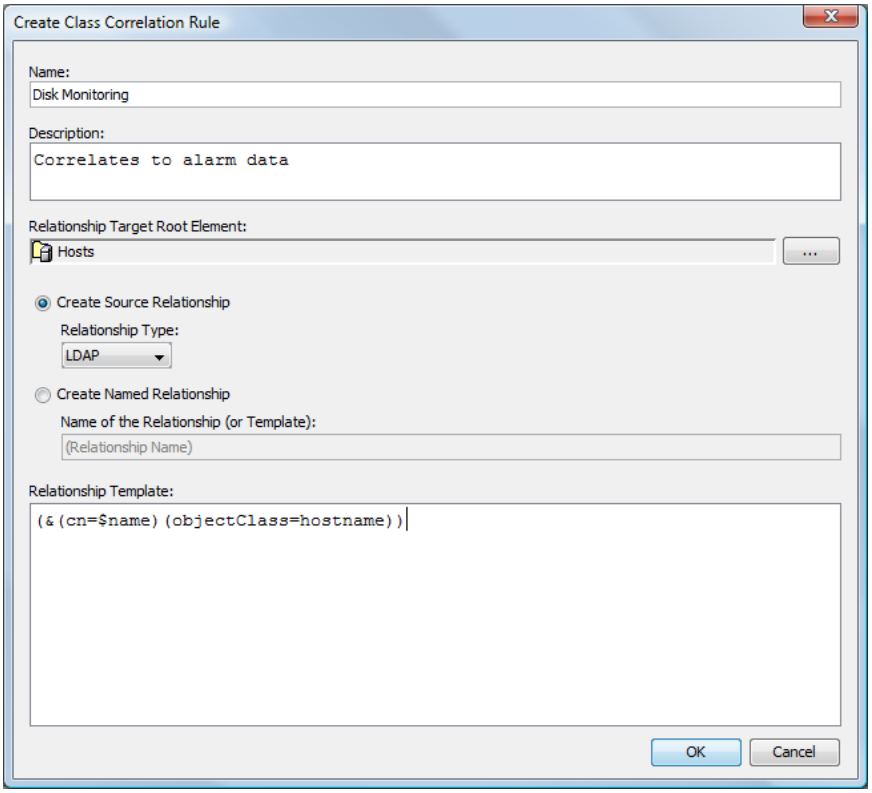


Create an LDAP expression that requires the name (cn) of the *Model Disk Array* element to match elements of the class `hostname` anywhere in the *Hosts* branch of the *Element* hierarchy:

(&(cn=$name) (objectClass=hostname))

Figure 4-7 shows the modified correlation rule. Note the relationship type is changed to LDAP and the expression above is entered in the *Relationship Template* section.

Figure 4-7 *Create Class Correlation Rule Dialog Box*



The modified correlation rule requires the name (cn) of the *Model Disk Array* element to match elements of the class `hostname` anywhere in the *Hosts* branch of the *Element* hierarchy

Use Case 3: Named, Property-Based Relationships for CMDB

Another common application of Class Relationship templates uses metamodel property values to generate relationships between two element classes. For example, a metamodel property named StoredOnArrays contains a comma-delimited list of elements that should be associated between Model Database classes and Model Disk Array classes. This property could be automatically pulled from an external data feed into a comma-separated list of array names.

The result of the correlation rule shown in Figure 4-8 is that a new relationship named `Uses` is generated between any Model Database, which contains such a property and which matches applicable *Model Disk Array* elements:

Figure 4-8 *Create Class Correlation Rule Dialog Box*

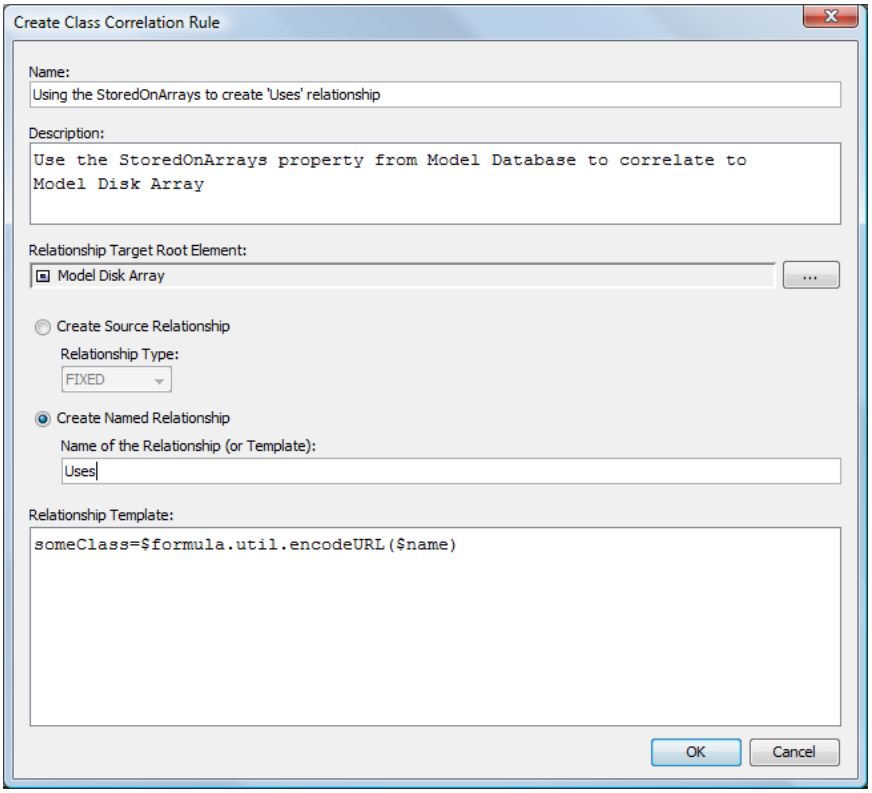
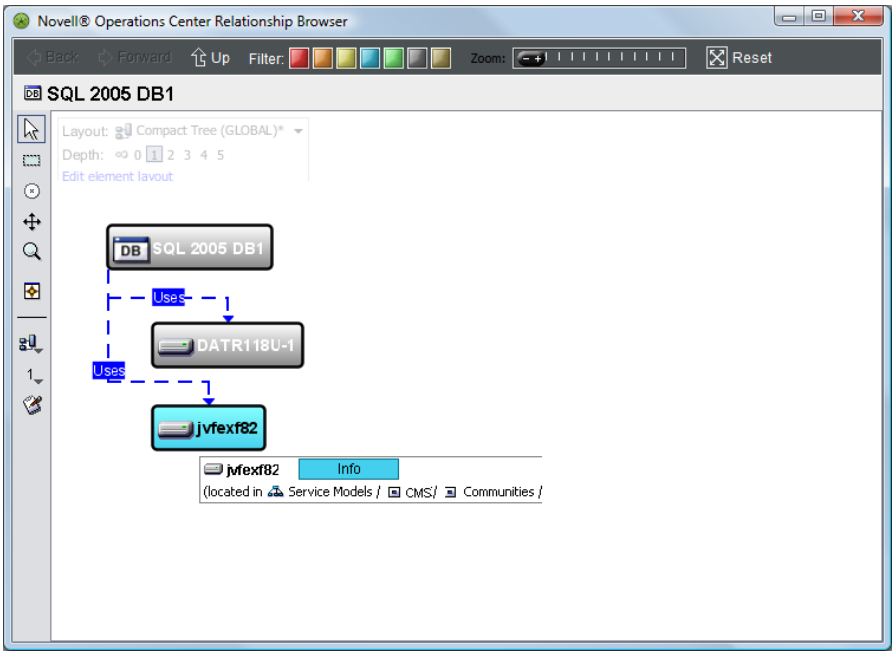


Figure 4-9 shows the generated new type of relationship named “Uses” displayed in the Operations Center Relationship Browser:

Figure 4-9 *Relationship Browser*



4.5 Editing Classes, Behavior Models, Property Pages, and Properties

You can update classes, behavior models, and property pages and their properties after they are saved:

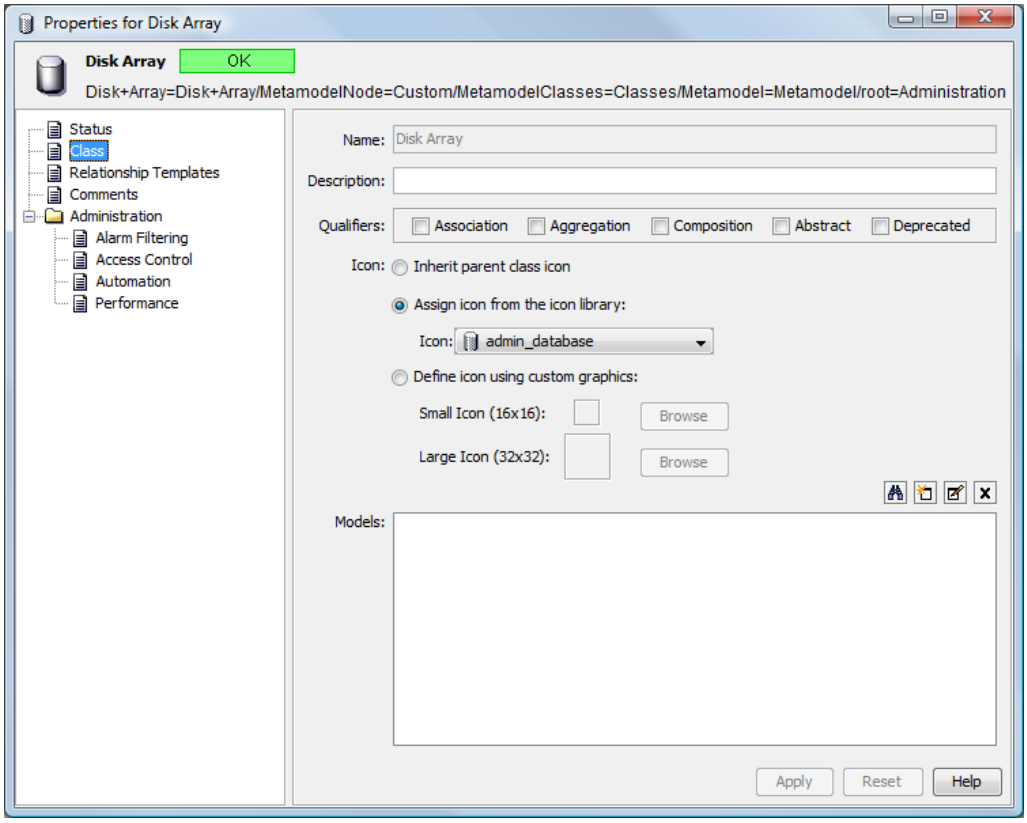
- Section 4.5.1, “Editing a Class,” on page 82
- Section 4.5.2, “Editing a Behavior Model,” on page 83
- Section 4.5.3, “Editing a Property Page,” on page 84
- Section 4.5.4, “Editing a Property,” on page 84

4.5.1 Editing a Class

1 In the Explorer pane, expand *Metamodel* > *Classes*.

2 Right-click the class element, then select *Properties* to open the Status property page.

3 In the left pane, click Class to update the property page.



4 Edit the class as appropriate.

For more information, see "Creating and Deleting Classes" on page 70.

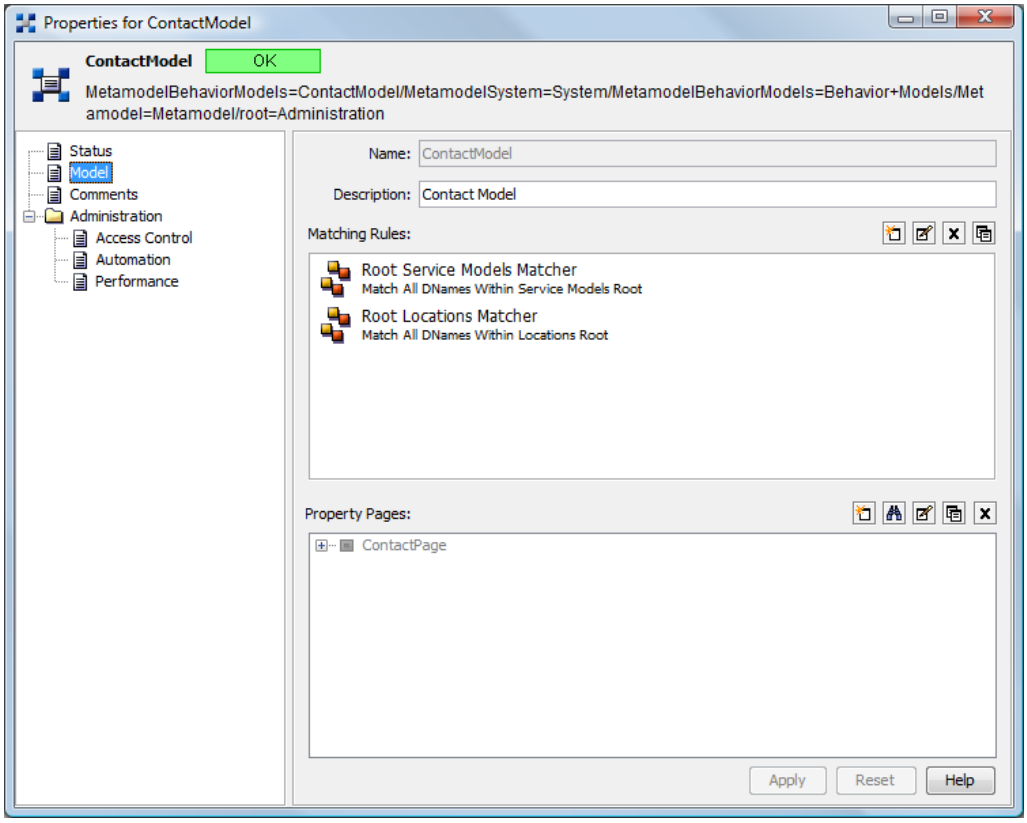
5 Click *Apply* to update the class.

4.5.2 Editing a Behavior Model

1 In the Explorer pane, expand *Metamodel* > *Behavior Models*.

2 Right-click the behavior model, then select *Properties* to open the Status property page.

3 In the left pane, click Model to update the property page.



4 Edit the model as appropriate.

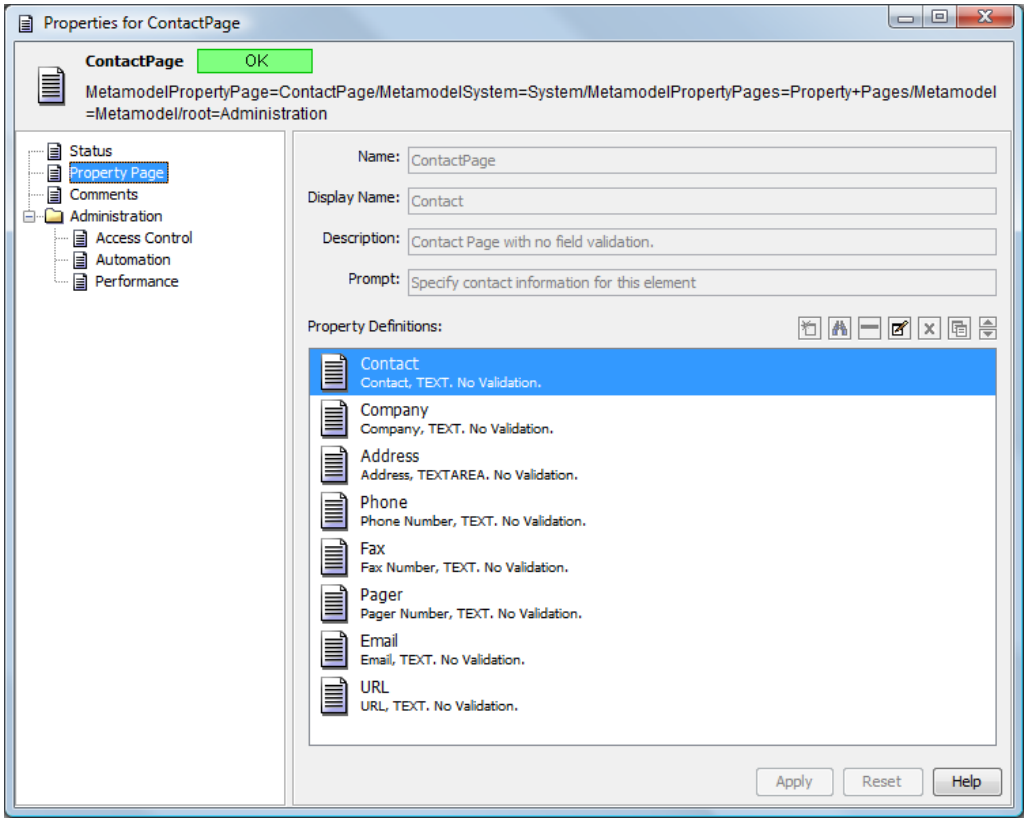
For more information, see "Creating and Deleting Behavior Models" on page 65.

5 Click *Apply* to update the behavior model.

4.5.3 Editing a Property Page

1 In the Explorer pane, expand *Metamodel* > *Property Pages*.

2 Right-click the property page, then select *Properties* to open the Status property page:



3 In the left pane, click Property Page to update the property page.

4 Edit the property page as appropriate.

For more information, see "Creating and Deleting Property Pages and Properties" on page 41.

5 Click *Apply* to update the property page.

4.5.4 Editing a Property

1 In the Explorer pane, expand *Administration* > *Metamodel* > *Property Pages*.

2 Right-click the property page that contains the property you want to edit, then select *Properties* to open the Status property page.

3 In the left pane, click Property Page to update the property page.

4 In the *Property Definitions* section, select the property you want to edit, then click Edit to open the Edit Property page.

5 Edit the property.

For more information, see "Creating a Property Page" on page 42.

6 Click *Apply* on the property page to save changes.

4.6 Security on Metamodel Elements

The access rights assigned to behavior models and property pages determine the access rights to property pages on elements. For more information, see the *Operations Center 5.5 Security Management Guide*.

4.7 Creating Custom Tooltips

Custom tooltips allow you to view important information about a service model or adapter element without opening property pages. The default tooltip shows basic information: element condition, class, and custom algorithm (if it exists). Use a custom tooltip property to display additional information in the element tooltip, such as user contact information.

In the following illustration, a custom tooltip property definition displays an element's Contact information:

Figure 4-10 *Tooltip with Contact Information*



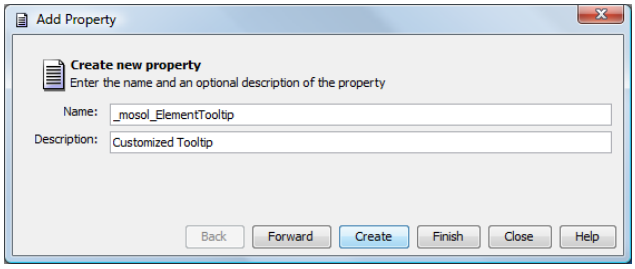
- Section 4.7.1, "Creating a Custom Tooltip," on page 85
- Section 4.7.2, "Removing Element Class from Tooltips," on page 87

4.7.1 Creating a Custom Tooltip

1. In the Explorer pane, expand *Administration* > *Metamodel* > *Property Pages*.
2. From an existing property page or from a newly created property page, click *Create a Property* to create a new property.

   The Add Property dialog box opens.

   For more information about property pages and custom properties, see "Creating and Deleting Property Pages and Properties" on page 41.
3. Enter `_mosol_ElementTooltip` in the *Name* field as shown:

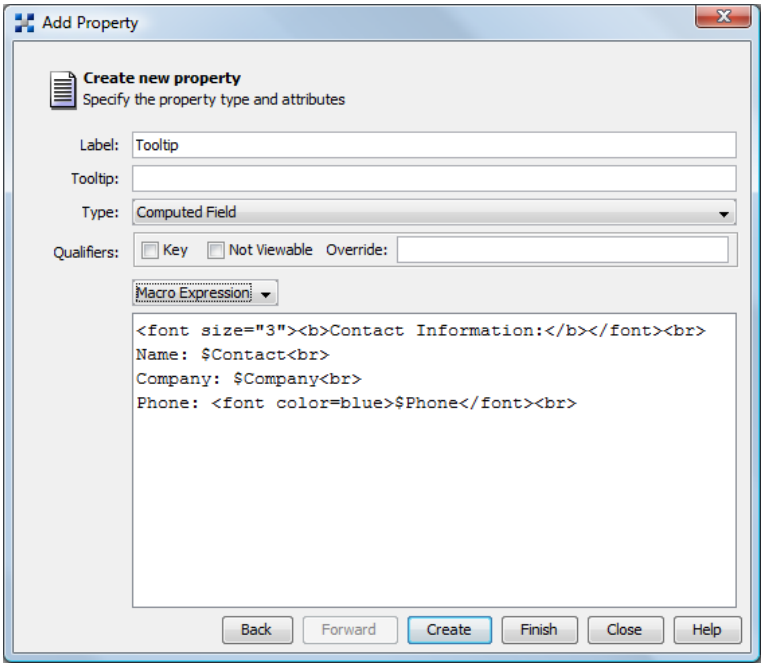


   This is an important step as the `_mosol_ElementTooltip` property name tells Operations Center to expose the property as a tooltip.
4. Click *Forward*.

   The dialog box updates to define property attributes.

5 Define the property attributes using the following information:

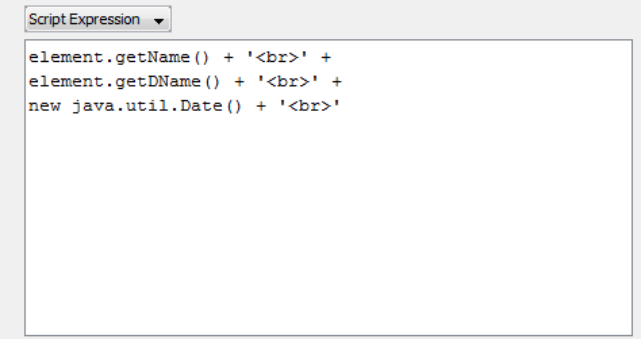
- A custom tooltip property can be any property type that allows you to store text. Custom tooltips support the use of HTML, but it is not necessary to enter the `<html>` and `<body>` tags. Just enter the actual markup tags.
- Use the *Computed Field* type to embed logic when the custom tooltip is generated. For example, in the following illustration, a Macro Expression is used to show contact information from the element's *Contact* property page, with a code check to not display anything if the property hasn't been defined.



The code in the above illustration results in the following tooltip:



- The computed field can contain NOC Script language or a velocity macro. For more information about velocity macros, go to http://velocity.apache.org/engine/devel/user-guide.html (http://velocity.apache.org/engine/devel/user-guide.html). For example, the following Script Expression uses NOC Script to get and display element name, DName, and current time in a table format using HTML:



6 Click Finish to apply the custom property page to adapter or service model elements.

When a user hovers the mouse over an element with the `_mosol_ElementTooltip` property, the tooltip displays information based on the custom tooltip definition.

4.7.2 Removing Element Class from Tooltips

By default, element class displays in element tooltips. It can be disabled if necessary in the `applet_params.xml` file.

To disable class name from tooltips:

1 Open the */OperationsCenter_install_path*`/html/applet_params.xml` file in a text editor.

2 Set the following line to `f"alse"` to omit the class name from tooltips:

```
<param name="ElementToolTip.ShowClass" value="false" />
```

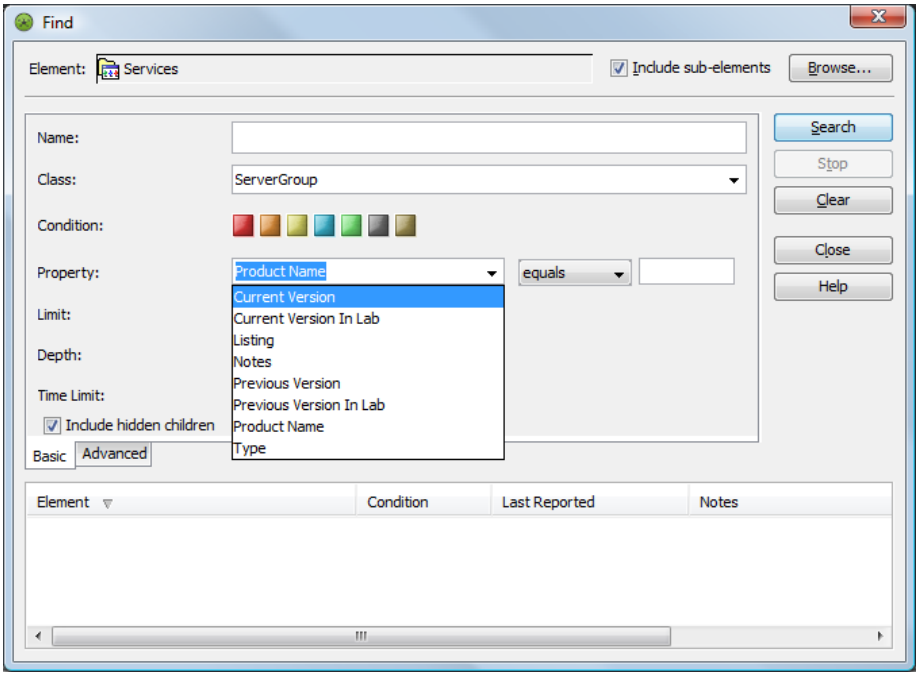
4.8 Using Classes and Properties to Enhance Finding Elements

When using the Find feature in the Operations Center console, it is possible to either enter a class or property name or select a custom class or property value from a drop-down list as search criteria.

Define the list of classes and properties that display by selecting the *Include in Find Dialog* check box when creating a custom class or property. The classes and properties are added to the *Class* or *Property* drop-down lists in the Find dialog box.

The following illustrates using the drop-down lists to view classes or properties that have been selected for display in the Find dialog box:

Figure 4-11 *Class and Property Look Up Enabled in the Find Dialog*



To enable the Find dialog with custom Class and Property look up:

1. Edit the */OperationsCenter_install_path*/html/client/template/launch.jnlp file, then add the following property assignment in the `resources` section:

   ```
   <property name="PretaggedSearchMode" value="true"/>
   ```

2. Run the Operations Center Configuration Manager and click *Apply*.

   For more information on the Operations Center Configuration Manager, see the *Operations Center 5.5 Server Configuration Guide*.

For more information about creating a class, see "Creating a Class" on page 71.

For more information about custom properties, see "Defining Properties for a Property Page" on page 43.

For more information on using the *Find* option, see *Using Find to Search for Elements* in the *Operations Center 5.5 User Guide*.

5 Adding Custom Properties to Service Elements

The previous section discussed how administrators can define custom property pages for service model elements. When defining custom properties for these pages, administrators can set up validation rules for entering data. For example, a specific property value can require entering an exact number of characters. In other cases, it is not possible to save a property page unless you enter valid values for specific required properties.

Only elements in the *Services* hierarchy can have custom property pages. Also, you cannot view specific custom property pages if the administrator has not granted you View access privileges for an element's custom property pages.

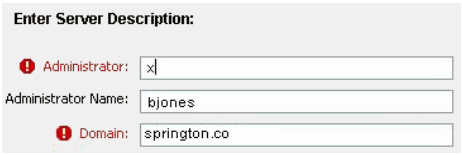
Read this section to learn how to add custom property values for individual service model elements.

- Section 5.1, "Editing Custom Properties," on page 89
- Section 5.2, "Creating and Viewing a Custom Property Page," on page 90

5.1 Editing Custom Properties

Table 5-1 describes the custom properties features.

Table 5-1 *Custom Properties Features*

Feature	Description
Required Properties	Red property names indicate a required property is blank or contains invalid values. Enter a valid value in a required property in order to apply changes to the property page.  In the left pane of the property page, bold page names contain required properties that contain no values or invalid values. When an invalid value is entered, a message displays at the bottom of the page. It is necessary to enter a valid value in order to apply changes to the property page.
Read Only/ Calculated Properties	Read only properties are dimmed and cannot be edited. If only some properties are dimmed, they are computed fields that are calculated using values entered in other properties or a regular or macro expression. If all properties are read only, it means the administrator has given you Read Only privileges to the element properties.

To update custom property values for a service model element:

1. Under the *Services* root element, right-click an element, then select *Properties* to open the Status property page.

   Property pages are listed in alphabetical order, with Administration pages listed last.

   Custom property pages that contain required properties are listed in bold typeface.
2. Review the custom property pages first by clicking the property page name.
3. Edit property values by clicking a property value field and entering a value.

   Consider the following when editing the property values:

   - Custom properties could have default values. Place the cursor in a property value field, then click *Set to Default* to update the property value. If a default value exists, it displays in the field.
   - Properties can display a mask to identify the format and/or number of characters required. For example, a telephone number mask:

     (###)###–####
   - Some property values might be read-only.
   - Property values can be calculated based on values entered for other properties.
4. Click *Apply* to save your edits and apply changes to the current property page.

   or

   Click *Apply All* to save all property pages.

   It is not possible to save property pages that contain required properties with no property values entered, or pages that contain invalid property values. Therefore, if you cannot save property pages, review the property pages, make corrections, then try to save again.
5. Close the property pages by clicking *Close*.

5.2 Creating and Viewing a Custom Property Page

It is possible to create custom property pages for individual elements in the *Services* hierarchy. Custom property pages display alongside the other property pages associated with an element. Access these pages by right-clicking an element in the Explorer pane, then selecting *Properties*.

NOTE: Custom property pages apply only to the element for which they are created. If an existing property page is "added" to an element using *Manage Properties* > *Browse*, a copy of the property page is created rather than linking the element to the metamodel property page definition. Any changes to the original property page definition in *Administration* > *Medamodel* > *Property Pages* are not updated in any copied property pages assigned to the element using *Manage Properties* > *Browse*. To assign an existing metamodel property page to an element or elements without making a custom copy, add a matching rule to a *Behavior Model* that then adds the *Property Page*. For more information about Behavior Models, see Section 4.3, "Creating and Deleting Behavior Models," on page 65.

Table 5-2 describes the operations that can be performed on custom property pages and fields.

Table 5-2 *Custom Property Operations*

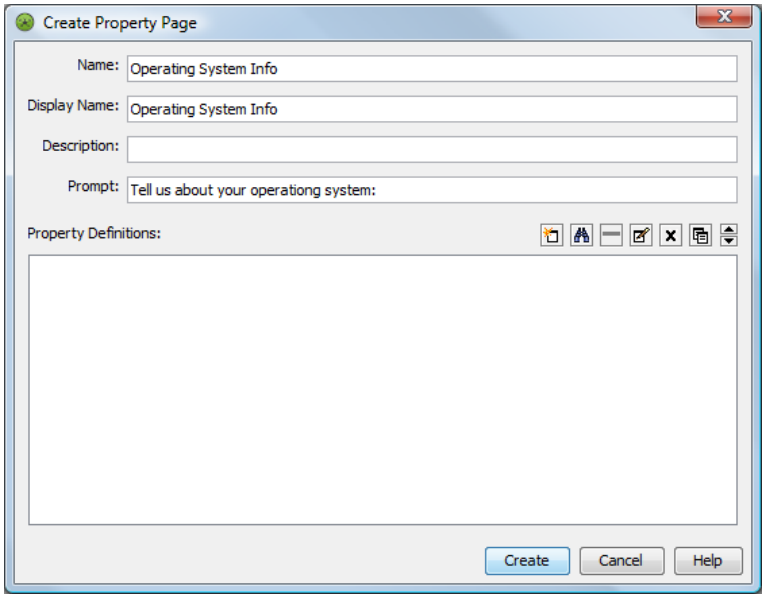
Operation	Button	Description
New Property Page		Creates a property page.
Browse		Select a property page created under *Administration* > *Metamodel* > *Property Pages*. Selecting a property page, makes a copy of the property page definition rather than linking the existing metamodel property page definition to the element. This new definition and any modifications to it, applies only to the selected element. For information on custom property pages, see Chapter 4, "Defining Classes, Behavior Models, and Property Pages," on page 37.
Edit		Enables editing either the highlighted property page or field.
Copy		Creates a copy of the highlighted property page or field. The copy has the same name as the original with a number in parenthese. For example: copy(1)
Delete		Deletes the highlighted property page or field.

- Section 5.2.1, "Creating a Custom Property Page," on page 92
- Section 5.2.2, "Viewing Custom Property Pages," on page 94

5.2.1 Creating a Custom Property Page

To create a custom property page:

1. In the Explorer pane, right-click an element in the *Services* hierarchy, then select *Manage Properties* to open its dialog box.
2. Click (*New*) topen the Create Property Page dialog box:



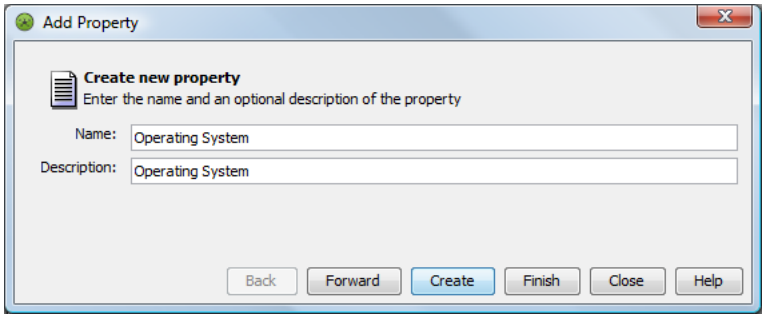
3. In the *Name* field, specify the new property page name.
4. To add custom property fields to the page, click (*New*) in the *Property Definitions* section to open the Add Property dialog box.
5. Specify the first property name and description, then click *Forward*.

   The remaining dialog boxes, which set up the property type, the default values, masks, required properties, and validation rules, are explained in "Defining Properties for a Property Page" on page 43.

   Property names cannot include the space character or the following special characters:

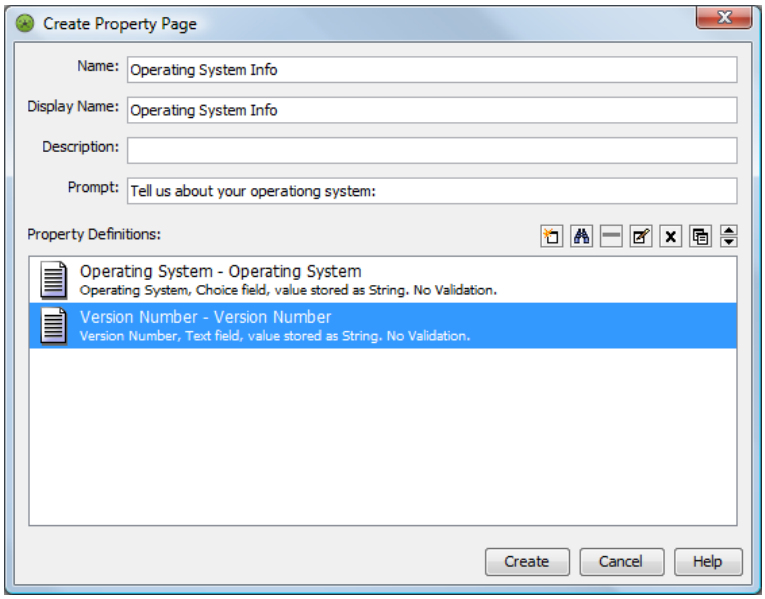
   <>#%"'{}()[]|\^`)
6. Click *Create* to add the property.

   Add more properties as necessary.



7 Click *Close* when finished defining properties.

The Create Property Page dialog box displays the properties:



8 Click *Create* to create the custom property page.

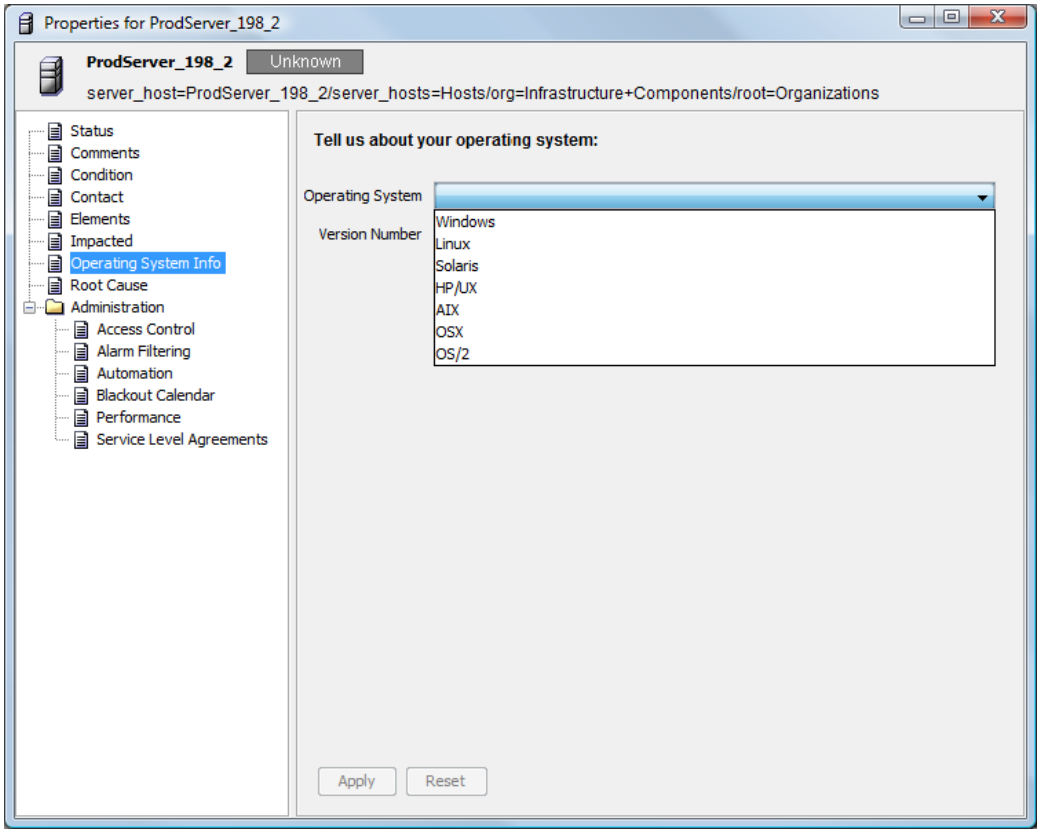
5.2.2 Viewing Custom Property Pages

1 In the Explorer pane, right-click the element, then select *Properties* to open the Status property page.

All custom property pages display at the bottom of the list on the left.

2 In the left pane, click the custom property page name to displya its page.

The following illustration shows a custom property page named BPA. The custom property Operating System displays a drop-down list of options.



6 Understanding Element Relationships

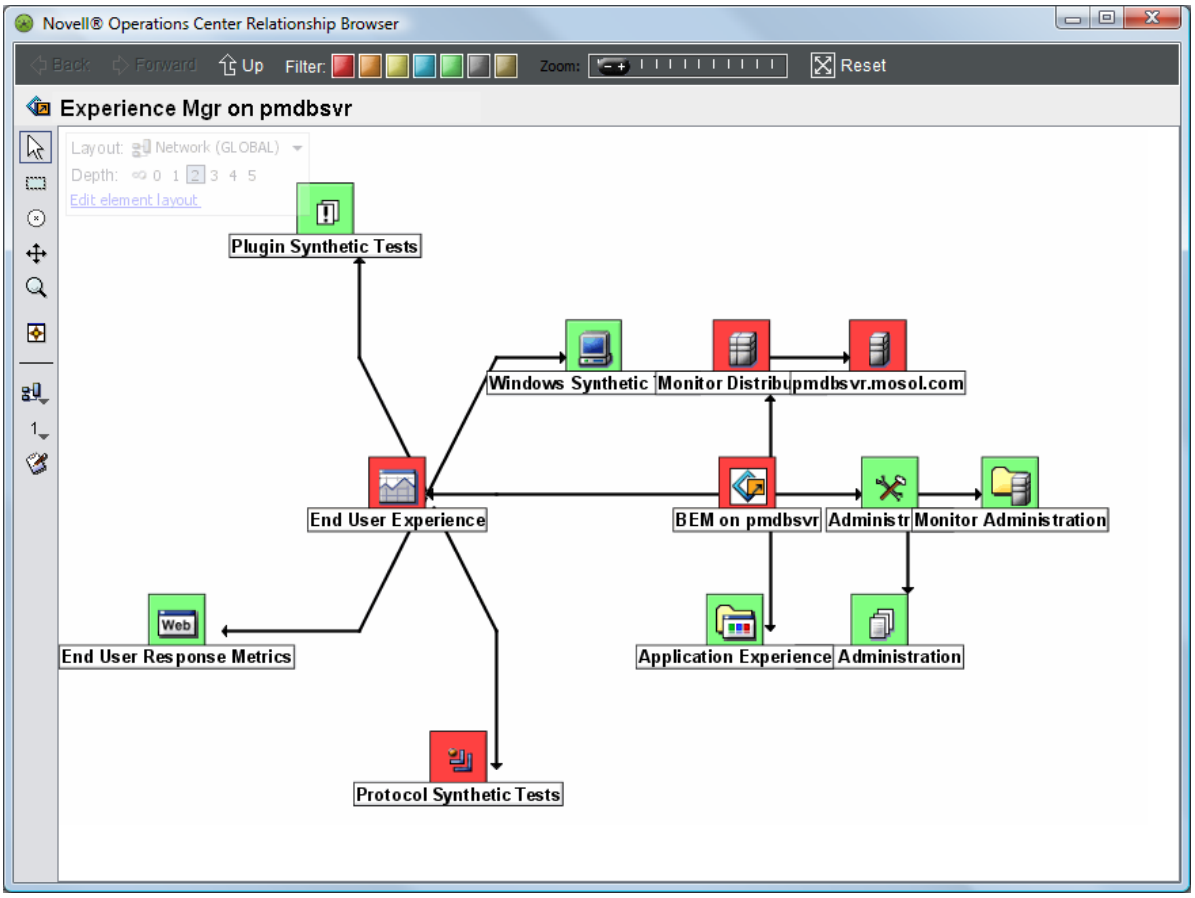
The Relationship Browser provides advanced features that enhance visualization and navigation of element relationships.

- Section 6.1, “Visualizing Relationships Using the Relationship Browser,” on page 96
- Section 6.2, “Navigating the Layout and Exploring Element Hierarchies,” on page 97
- Section 6.3, “Rendering and Exploring Relationships,” on page 98
- Section 6.4, “Relationship Diagram Display Options,” on page 101
- Section 6.5, “Relationship Browser Configuration Presets,” on page 103
- Section 6.6, “Using Layout Rendering Features,” on page 109
- Section 6.7, “Adding Elements,” on page 117

6.1 Visualizing Relationships Using the Relationship Browser

The Operations Center Relationship Browser provides advanced presentation features, increased visualization of element relationships, and enhanced navigation of hierarchical structures.

Figure 6-1 *Relationship Browser*



The Relationship Browser renders an enhanced layout of the selected hierarchy, including relationships between such objects as technology and service components.

The term "layout" used throughout this section refers to the placement of elements in the Relationship Browser. It is independent of the *Layout* view, which is described in the *Operations Center 5.5 Custom Drawing and Layout Guide*. However, the layout selections for specific elements are linked. If you apply a specific layout to an element in the Relationship Browser, the same layout displays when the element is selected in the *Layout* view, and vice-versa.

To open the Relationships Browser for an element hierarchy:

1 In the Explorer pane, right-click the element, then select *Show Relationships* to open the Relationship browser.

An element discovery process occurs and a default layout displays.

6.2 Navigating the Layout and Exploring Element Hierarchies

Use the navigational tools in the Relationship browser sidebar to locate and zoom in on objects:

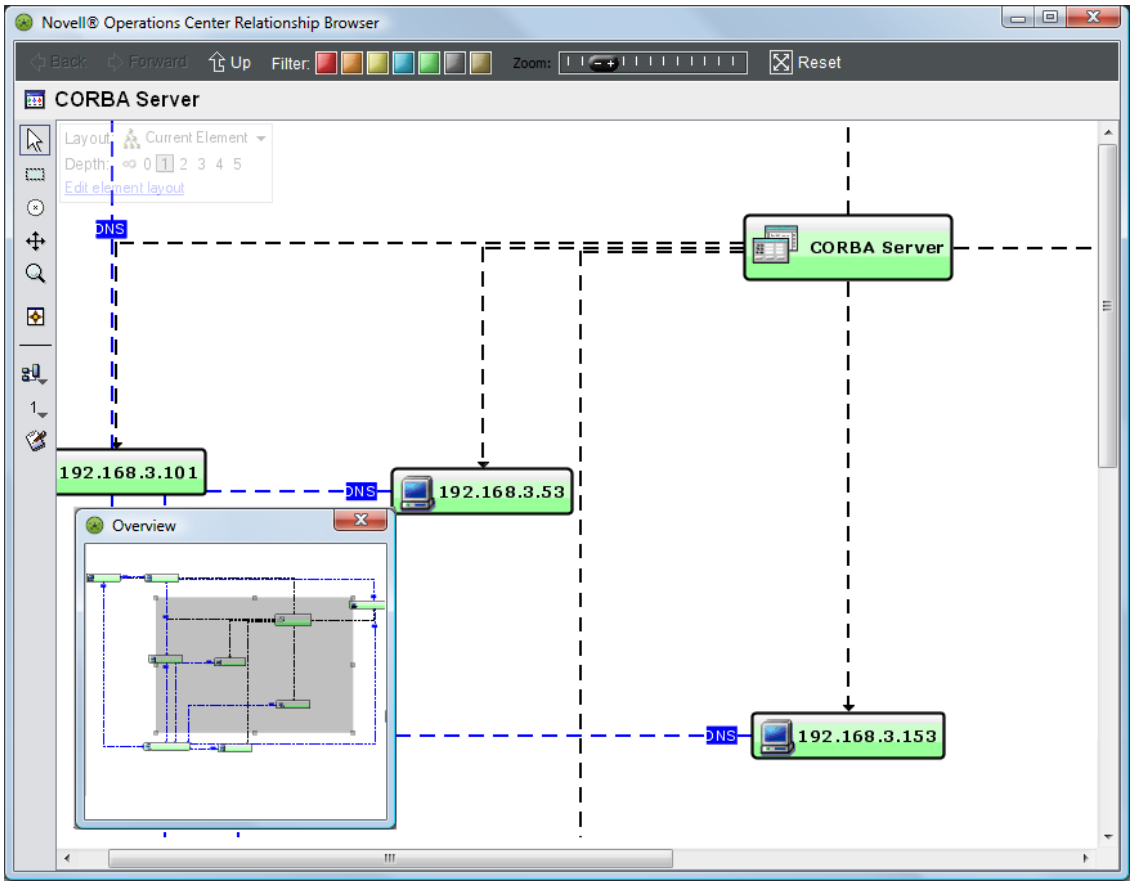
Table 6-1 *Navigational Tools*

Icon	Name	Description
	Standard View	Standard viewing mode. Click an element to zoom and center it in the browser. Double-click the layout background to center all the elements.
	Marqee Zoom	Zooms in to a specific area. Click and drag to select an area. To gradually zoom out, click the background.
	Focus Zoom	Zooms the entire diagram in relation to the location of the focus zoom's circle. Click and drag the gray circle to enlarge (zoom in) it or shrink it (zoom out).
	Pan Map Navigation	Click and drag to navigate across the entire layout.
	Fisheye Zoom	Hover the cursor over an element to view element details. This is most effective for viewing a large layout where details are not easily legible.
	Overview Navigator	Opens a small dialog box for viewing and navigation purposes. Click and drag the cursor to mark the area to focus on in the Relationship browser.

A minimum of View security privileges to both elements in a relationship is required, or else the relationship is not displayed in the Relationship browser. For more information on element security privileges, see the *Operations Center 5.5 Security Management Guide*.

To use the Overview Navigator to navigate the relationship diagram:

1 Click Overview Navigator on the left sidebar to open the small Overview dialog box, overlaying a portion of the Relationship browser:



You can navigate the diagram using the Overview dialog box.

2 Do any of the following:

- In the Overview dialog box, click and drag the cursor over the area to display.

  The Relationship browser updates to zoom into the selected portion.

  A gray square displays over the selected portion of the diagram.
- Click the gray navigational square and drag to a new position.

  The Relationship browser updates to show the new selection.

3 When finished, close the Overview dialog box.

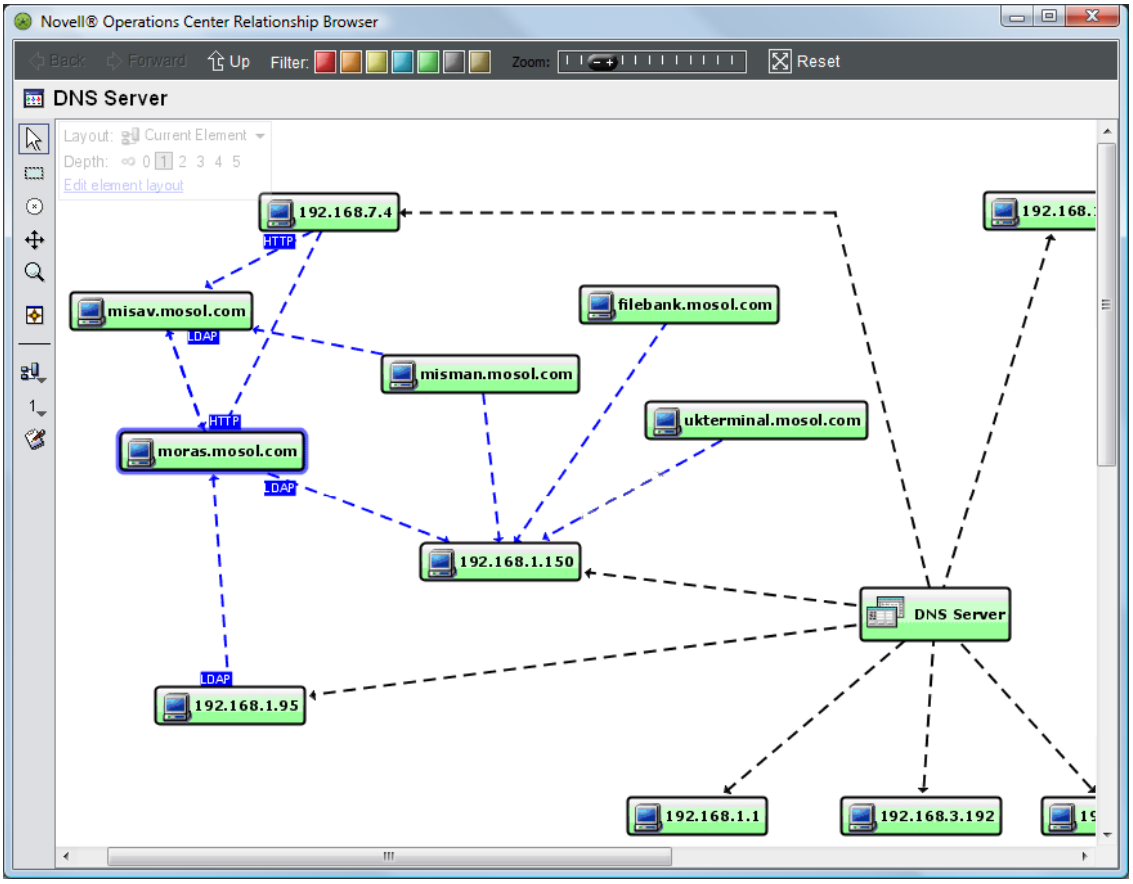
6.3 Rendering and Exploring Relationships

- Section 6.3.1, “Understanding Relationships,” on page 99
- Section 6.3.2, “Viewing Additional Relationships,” on page 100
- Section 6.3.3, “Exploring Dependency Relationships,” on page 101

6.3.1 Understanding Relationships

The Relationship browser distinguishes between standard parent/child relationships and linked relationships that can occur between elements. For more information on linked relationships. see "Copying, Moving and Linking Elements" on page 34.

Figure 6-2 *Relationship browser*



Relationships between elements (called nodes) display in the Relationship browser using different connector styles. Table 6-2 describes the connector styles and the corresponding relationship types.

Table 6-2 *Relationship Connector Styles*

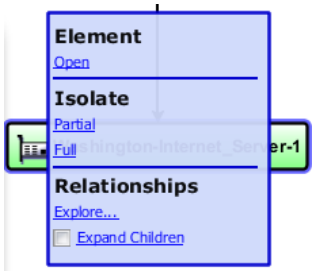
Connector Type	Definition
	**Basic parent-child connector.** One element is a natural child of the other element, meaning it was created directly under its parent. This occurs most often in relationships between elements under the *Elements* root element.
	**Linked parent-child connector.** One element is a linked child of the other element. This occurs in the *Services* hierarchy when a child element is linked using the *Link* option. It contributes to the condition of the linked parent.

Connector Type	Definition
	**Dependency connector.** One element is dependent on the other element. This occurs when dependency relationship information is available, mostly through the use of Discovery Tools. For information about the discovery tools that Operations Center supports, see the *Operations Center 5.5 Adapter and Integration Guide*.

6.3.2 Viewing Additional Relationships

To view additional relationships in the Relationship browser:

1 Double-click an element to open the following menu:



Use this menu to expand your view of an element's relationships.

2 Select the relationship type to display:

- **Open:** Select to display only the selected element and its children.
- **Isolate:** Select *Full* to display only the element and its parent. Select *Partial* to display the entire branch leading to the element.
- **Explore:** Opens the Explore Dependencies dialog box so you can view an element's dependencies on other elements.
- **Expand Children:** Select this check box to display the element's children in the Relationship browser.

The diagram expands to display additional elements. Based on the relationship types selected, connectors and nodes display to depict the additional relationships.

Dependency relationships connectors display in blue with labels to show the relationship type, as shown in the previous table and illustration.

TIP: Use the element right-click *Navigate Relationships* option to quickly view an element's dependency relationships and navigate to one of the end-point elements named in a relationship. For more information, see "Analyzing Dependencies" on page 33.

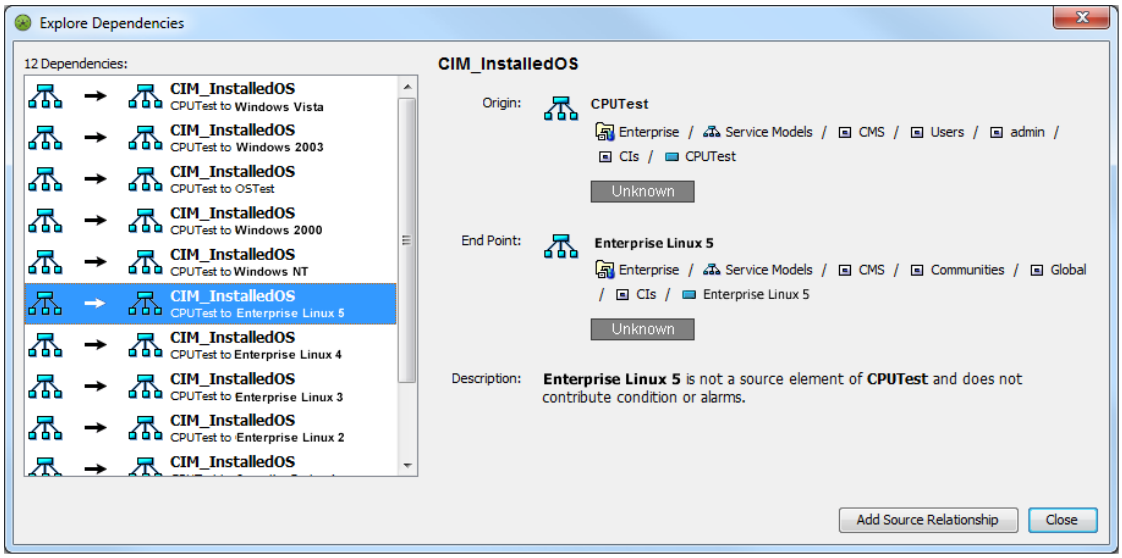
6.3.3 Exploring Dependency Relationships

A minimum of View security privileges to both the origin and end point elements in a relationship is required, or else the relationship is not displayed in the Explore Dependencies dialog box. For more information on element security privileges, see the *Operations Center 5.5 Security Management Guide*.

To explore dependency relationships:

1. In the Relationship browser, do one of the following to open the Explore Dependencies dialog box:
   - In the Relationship browser, right-click the element node, then select *Explore Dependencies*.
   - Double-click an element, then when the menu shown above displays, click the Explore link.
2. In the Explore Dependencies dialog box, select a dependency.

   Dependency information displays in the right pane:



3. Click *Close* when finished viewing dependencies.

6.4 Relationship Diagram Display Options

Customize the relationship diagram by:

- Selecting a style for displaying element nodes

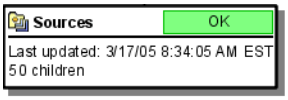
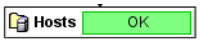
  For instructions, see "Selecting Element Node Styles" on page 102.
- Displaying or hiding the root element or specific types of relationship connectors

  For instructions, see "Relationship Display Options" on page 102.

6.4.1 Selecting Element Node Styles

Select from several styles for rendering element nodes in relationship diagrams. Table 6-3 describes the styles:

Table 6-3 *Element Node Display Styles*

Name	Display renders…	Sample Rendering
Classic	Element nodes display similar to the classic *Layout* view.	
Bubble	Element nodes display with rounded strong borders.	
Detailed	Element nodes display with larger boxes that show basic details about the element state.	
Tooltip	Element nodes display in narrow tooltip boxes.	

To change the node display type:

1 Right-click the browser background and select one:
   - *Display Style*
   - *Classic*
   - *Bubble*
   - *Detailed*
   - *Tooltip*

   The display updates to the selected display type.

6.4.2 Relationship Display Options

Relationship diagrams also provide options for including the root element, parent/child relationships, and dependency relationships in the diagram. Use the following steps to select these options:

- "Hiding or Displaying the Element Root" on page 102
- "Hiding or Displaying Dependency Relationship Connectors" on page 103
- "Hiding or Displaying Basic Parent-Child Relationship Connectors" on page 103

Hiding or Displaying the Element Root

To hide or display the element root (the topmost element) in the Relationship browser, right-click the background, then select *Show Root*.

When selected, the root element displays.

Hiding or Displaying Dependency Relationship Connectors

To hide or display dependency relationship connectors, right-click the background, then select *Include Relationships in Layout*.

When selected, the dependency relationship connectors display.

Hiding or Displaying Basic Parent-Child Relationship Connectors

To hide or display basic parent-child relationship connectors, right-click the background, then select *Show Parent-Child Relationships*.

When selected, the parent-child relationship connectors display.

6.5 Relationship Browser Configuration Presets

The Relationship browser includes several global layout configurations, called presets. Define configuration presets at the following levels:

- **Global:** A configuration available to all users.
- **User:** A configuration defined by a user and available only to that user.
- **Element:** A configuration setting saved and applied only to the current element in view.

To define configuration presets, review the following sections:

- Section 6.5.1, "Updating the Relationship Browser Configuration for the Current Element," on page 103
- Section 6.5.2, "Working with Global and User Presets," on page 105
- Section 6.5.3, "Creating and Deleting Presets," on page 108

6.5.1 Updating the Relationship Browser Configuration for the Current Element

It is possible to customize and save Relationship Browser settings for individual elements. This saved configuration is called an element preset.

A customized element preset takes precedence over global and user presets when displaying the element's relationship diagram. If no element preset exists, user presets take priority over global presets.

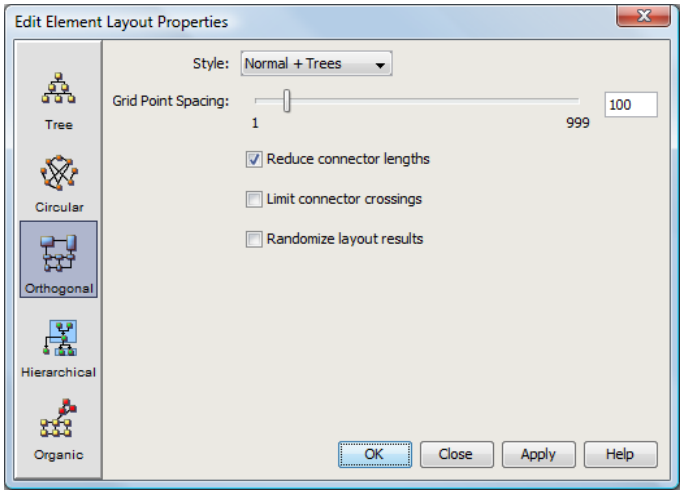
- "Modifying Configuration Settings" on page 104
- "Saving the Relationship Browser Layout" on page 105

Modifying Configuration Settings

To modify configuration settings for the currently selected element:

1 Do one of the following in the Relationship browser to open the Edit Element Layout Properties dialog box:

- In the sidebar, click Edit Element Layout Properties.
- In the Control Panel, click the Edit element layout link.



2 In the sidebar, select a layout type:

- *Tree*
- *Circular*
- *Orthogonal*
- *Hierarchical*
- *Organic*

3 The Style and other settings in the dialog box vary depending on the selected layout type.

If necessary, update these settings. For more information to select the appropriate link, see "Using Layout Rendering Features" on page 109.

4 Do one of the following:

- Click *Apply* to apply selections to the current view, but keep the dialog box open.
- Click *OK* to save the settings and close the dialog box.

Saving the Relationship Browser Layout

There are two ways to save the displayed view for the current element. Save the layout for the element only or save it for the class to which the element belongs. Saving a class layout automatically applies the saved layout to all elements in that class. If the *Save Element Layout* option is used, that layout displays when the Relationship browser is open for the element. In other words, the saved element layout takes priority over the class layout.

- To save the Relationship Browser layout for the current element only, right-click the browser background, then select *Save Element Layout*.

  This saved layout is displayed when the Relationship browser is open for the element.
- To save the Relationship Browser layout for the class to which the element belongs, right-click the browser background, then select *Save Element Class Layout*.

  This saved layout is displayed for any element in the class, unless a different layout was saved using the *Save Element Layout* option.

6.5.2 Working with Global and User Presets

To change the default global and user presets, use the Control Panel in the upper left corner of the Relationship browser or use the sidebar icons.

Figure 6-3 *Relationship Browser Control Panel*



Figure 6-4 *Sidebar Controls*



The following sections describe how to change both global and user presets as well as other Relationship Browser settings:

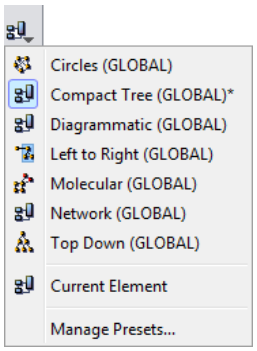
- "Applying a Global or User Preset" on page 106
- "Modifying the Settings for a Preset" on page 107
- "Editing the Preset Name, Display Type or Global/User Setting" on page 107

Applying a Global or User Preset

The first step is selecting an existing global or user preset to see what it looks like when applied to an element in the Relationship browser.

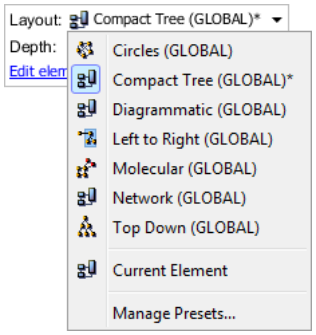
To apply a global or user preset, do one of the following in the Relationship browser:

- In the sidebar, click Layout Preset, then select a preset from the drop-down list.



TIP: The Layout Preset icon changes based on the current layout type selected.

- In the Control Panel, click the current Layout selection, then select a preset from the drop-down list.

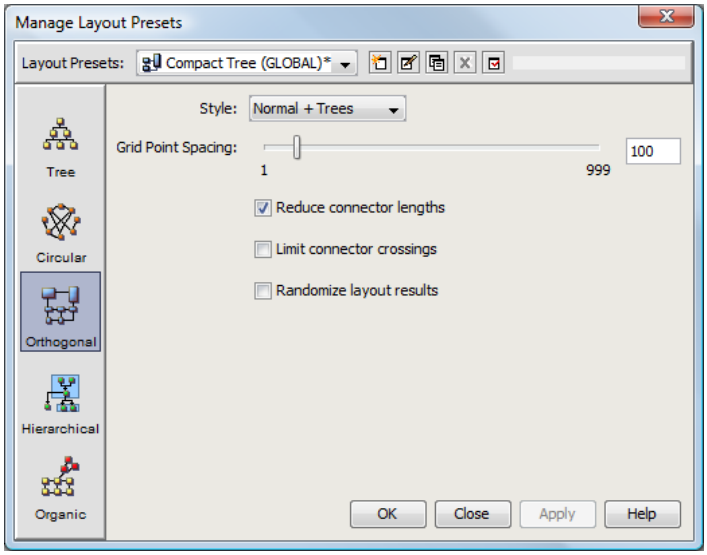


The diagram updates using the selected preset. The next step is modifying the global or user preset.

Modifying the Settings for a Preset

1 Do one of the following in the Relationship browser to open the Manage Layout Presets dialog box:

- In the sidebar, click Layout Preset, then select *Manage Presets* from the drop-down list.
- In the Control Panel, click the current Layout selection, then select *Manage Presets* from the drop-down list.



2 Click the *Layout Presets* drop-down, then select the preset to modify.

The settings update in the dialog box.

3 Edit the settings in the dialog box.

The settings vary by layout style (*Tree*, *Circular*, and so on).

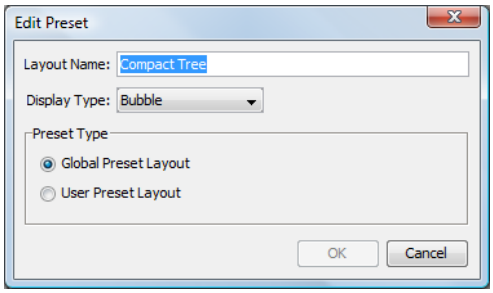
For more information, see "Using Layout Rendering Features" on page 109.

4 To save the edited preset, do one of the following:

- To apply the settings and keep the dialog box open, click *Apply*.
- To save the settings and close the dialog box, click *OK*.

Editing the Preset Name, Display Type or Global/User Setting

1 In the Manage Layout Presets dialog box, click Edit Preset to open its dialog box:



2 Specify the name for the preset in the *Layout Name* field.

3 Click the *Display Type* drop-down list, then select the layout type.

   For more information, see "Using Layout Rendering Features" on page 109 for details on the different types.

4 To select a preset type, do one of the following:

   - Select the *Global Preset Layout* radio button to make this layout available to all users.
   - Select the *User Preset Layout* radio button to make this layout available in the user's library of presets.

5 Click *OK* to update the preset.

6.5.3 Creating and Deleting Presets

In some situations, creating a preset is preferable to modifying an existing preset. Review the following sections to create or delete a preset:

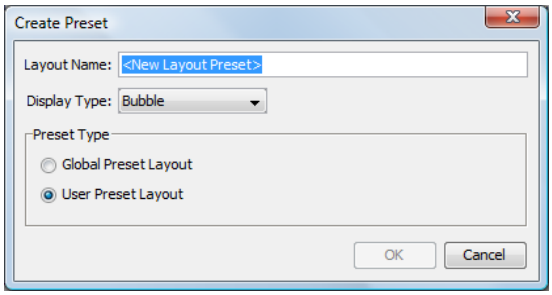
- "Creating a Preset" on page 108
- "Deleting a Preset" on page 109

Creating a Preset

1 Do one of the following in the Relationship browser to open the Manage Layout Presets dialog box:

   - In the sidebar, click Layout Preset, then select *Manage Presets* from the drop-down list.
   - In the Control Panel, click the current Layout selection, then select *Manage Presets* from the drop-down list.

2 In the Manage Layout Presets dialog box, click New Preset to open the Create Preset dialog box:



3 Specify the name for the new preset in the *Layout Name* field.

4 Click the *Display Type* drop-down list, then select the layout type.

5 To select a preset type, do one of the following:

   - Select the *Global Preset Layout* radio button to make this layout available to all users.
   - Select the *User Preset Layout* radio button to make this layout available in the user's library of presets.

6 Click *OK*.

   The new preset displays as the current preset in the Manage Layout Presets dialog box.

7 In the Manage Layout Presets dialog box, select the layout style (*Tree*, *Circular*, and so on).

The settings on the right side of the dialog box update.

For more information, see "Using Layout Rendering Features" on page 109.

8 Edit the settings in the dialog box, which vary by layout style.

For more information, see "Using Layout Rendering Features" on page 109.

9 Do one of the following:

- To apply the preset and keep the dialog box open, click *Apply*.
- To save the preset and close the dialog box, click *OK*.

Deleting a Preset

1 Do one of the following in the Relationship browser to open the Manage Layout Presets dialog box:

- In the sidebar, click Layout Preset, then select *Manage Presets* from the drop-down list.
- In the Control Panel, click the current Layout selection, then select *Manage Presets* from the drop-down list.

2 In the Manage Layout Presets dialog box, click the *Layout Presets* drop-down list, then select the preset.

The settings update in the dialog box.

3 Click Delete to open a confirmation dialog box.

4 Click *Yes* to confirm the deletion.

The preset is removed.

6.6 Using Layout Rendering Features

The Relationship browser can render a hierarchy using the following layout styles:

Table 6-4 *Relationship Browser Layout Rendering Options*

Layout Style		Renders…
Tree		A tree-like structure with variations. Select from tree layout styles including: *Directional*, *Balloon*, *Horizontal-Vertical*, and *Compact*. For more information, see "Tree Layout" on page 111.
Circular		An interconnected ring or star topology. Layout emphasizes group structures within a network. It creates node partitions by analyzing the connectivity structure of the network, and arranging the partitions as separate circles. The circles are arranged in a radial tree layout. Select from circular layout styles including: *BBC Compact*, *BBC Isolated*, and *Single Cycle*. For more information, see "Circular Layout" on page 112.

Layout Style	Renders…
Orthogonal 	Members of a group share a common rectangular area. This produces compact drawings that are easy to read with no overlaps, few crossings, and few bends. Select from orthogonal layout styles including: *Normal*, *Normal + Trees*, *Node Boxes*, *Uniform Node Sizes*, and *Mixed*. For more information, see "Orthogonal Layout" on page 114.
Hierarchical 	A visualization of hierarchical or pseudo-hierarchical scenarios. Portrays the precedence relation of directed graphs and highlights the main direction or flow within a directed graph. Automatically detects and resolves cyclic dependencies of nodes are. Places nodes in hierarchically arranged layers. Additionally, orders nodes within each layer to produce a minimal number of line (or edge) crossings. For more information, see "Hierarchical Layout" on page 115.
Organic 	A clear representation of complex diagrams similar to ER diagrams or UML diagrams. Resulting layouts often expose the inherent symmetric and clustered structure of a graph, a well balanced distribution of nodes and few edge crossings. This is well suited for the visualization of highly connected backbone regions with attached peripheral ring or star structures. These structurally different regions of a network are easily identified in a diagram produced by this layout type. For more information, see "Organic Layout" on page 116.

These styles are listed in the sidebar of the Manage Layout Presets dialog box.

To manage layout styles, review the following sections:

- Section 6.6.1, "Understanding the Different Layout Styles," on page 110
- Section 6.6.2, "Using a Different Layout Style in the Relationship browser," on page 116
- Section 6.6.3, "Saving the Current Layout as an Element Preset," on page 117

6.6.1 Understanding the Different Layout Styles

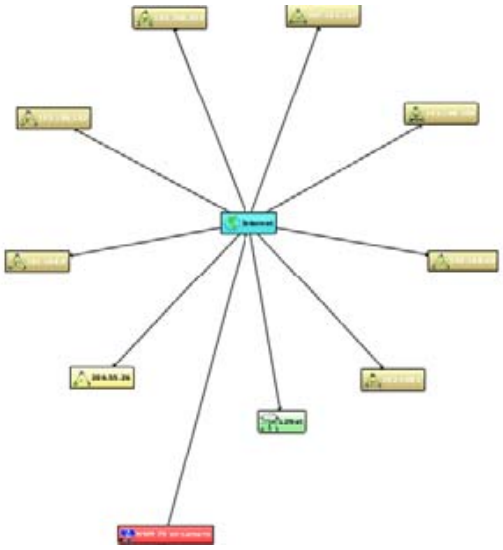
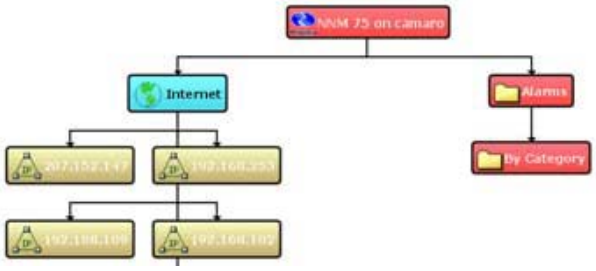
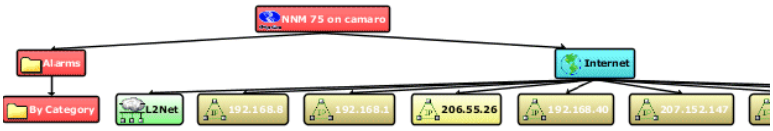
Review the following sections for an understanding of the different layout styles:

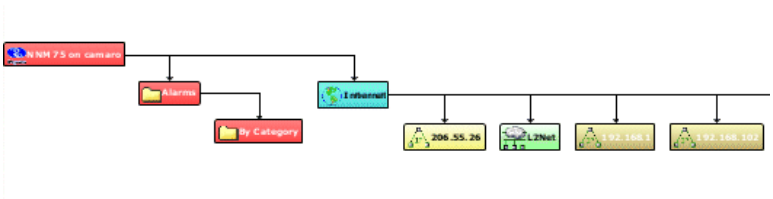
- "Tree Layout" on page 111
- "Circular Layout" on page 112
- "Orthogonal Layout" on page 114
- "Hierarchical Layout" on page 115
- "Organic Layout" on page 116

Tree Layout

Table 6-5 lists the Tree layout styles. The resulting layout can differ from the examples shown because of the different structure elements, relationships, and layout options.

Table 6-5 *Tree Layout Styles*

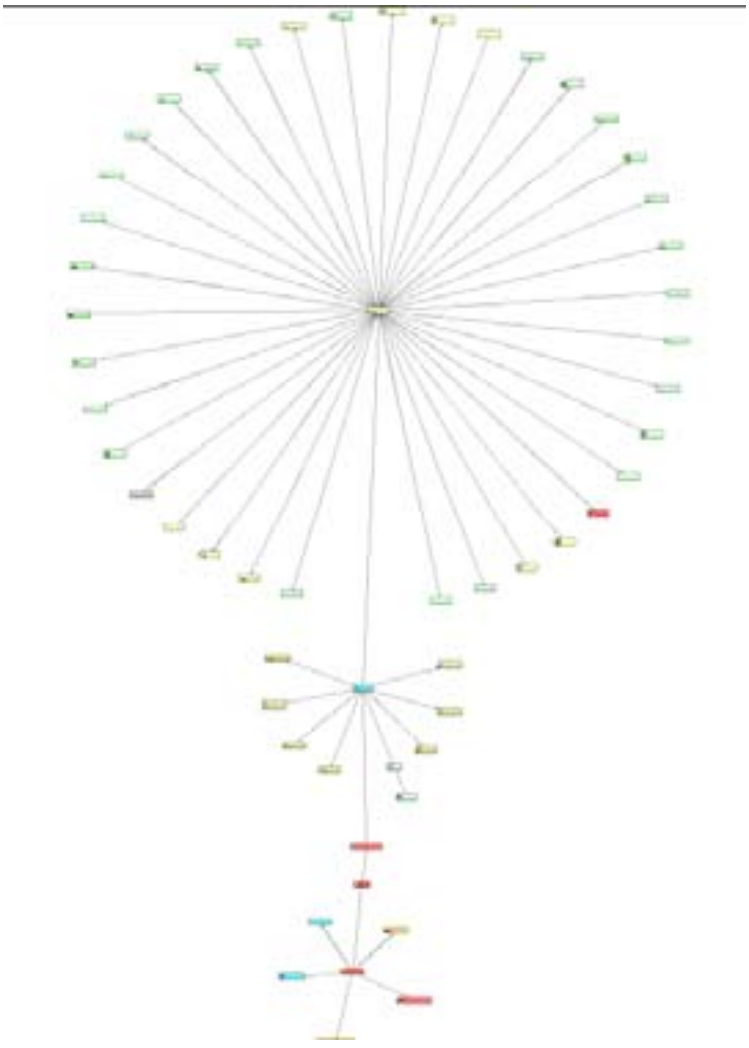
Style	Description	Sample Rendering
Balloon	Generates subtrees in a radial shape around the parent node. Ideally suited for networks with a large number of nodes (such as 10,000 nodes).	
Compact	Generates compact orthogonal tree drawings. Consider specifying a preferred aspect ratio (relation of width to height), which is useful when the layout needs to fit perfectly in the space available.	
Directional	Generates tree drawings with a root node. Nodes are arranged either from top to bottom, left to right, or bottom to top. Mainly used for directed trees that have a unique root element. Route edge connectors as either straight lines or multisegmented lines.	

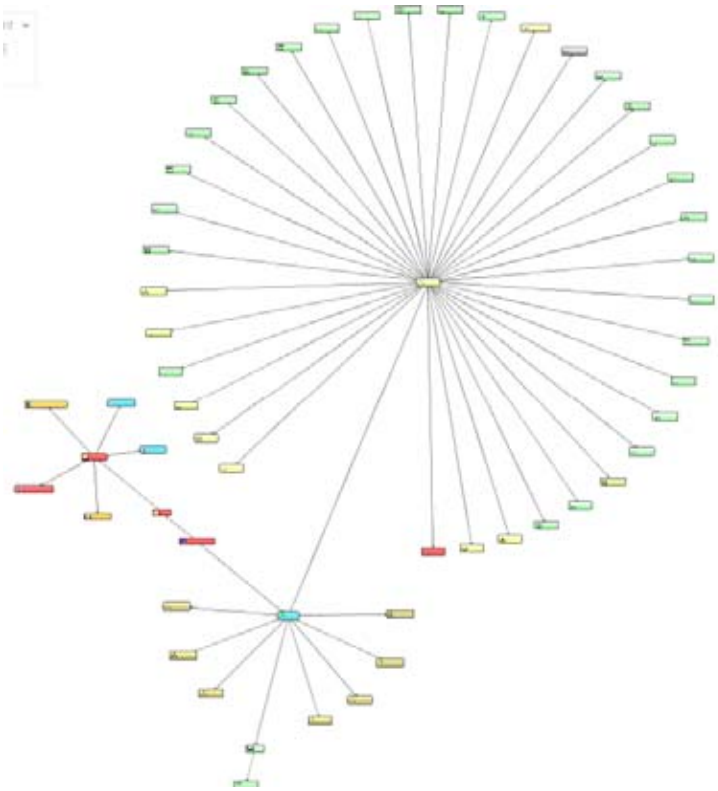
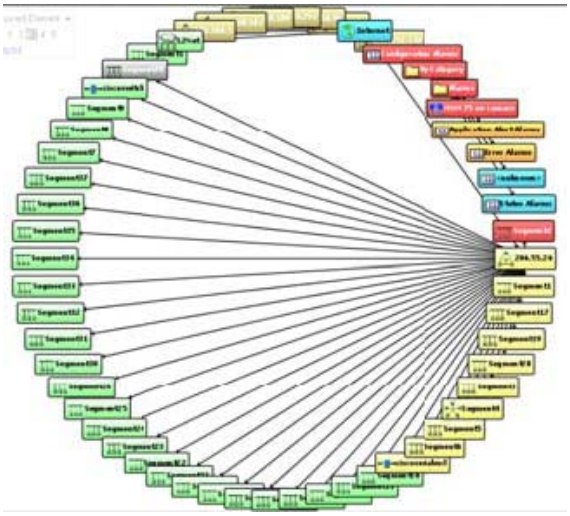
Style	Description	Sample Rendering
Horizontal/ Vertical	Generates subtrees with horizontal or vertical layout.	

Circular Layout

Table 6-6 lists the Circular layout styles. The resulting layout can differ from the examples shown because of the different structure elements, relationships, and layout options.

Table 6-6 *Circular Layout Styles*

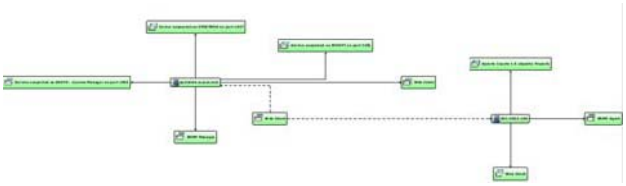
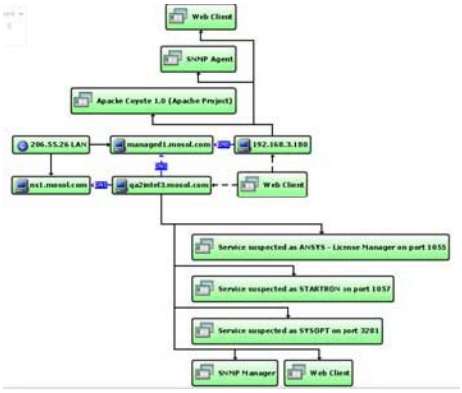
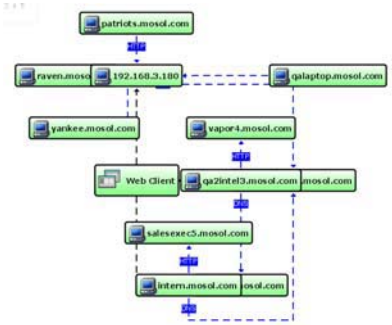
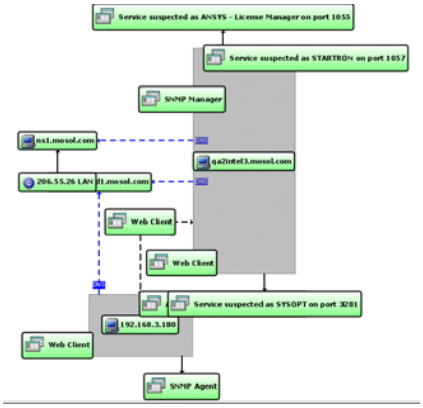
Style	Description	Sample Rendering
BCC Compact	Each partition represents a bi-connected component (BCC) of the graph. A bi-connected component consists of nodes that are reachable by two edge-disjointed paths. Nodes that belong to more than one bi-connected component are assigned exclusively to one partition.	

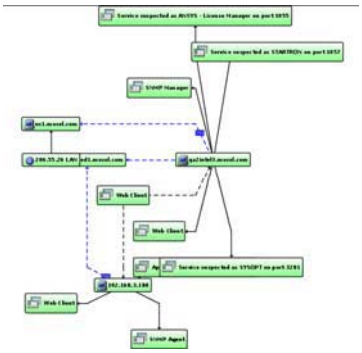
Style	Description	Sample Rendering
BCC Isolated	Node partitions are formed the same way as for the BCC Compact style, except that all nodes belonging to more than one bi-connected component are assigned an isolated partition.	
Single Cycle	Arranges notes in a single circle.	

Orthogonal Layout

The Orthogonal layout provides the style options as shown in Table 6-7. The resulting layout can differ from the examples shown because of the different structure elements, relationships, and layout options.

Table 6-7 *Orthogonal Layout Styles*

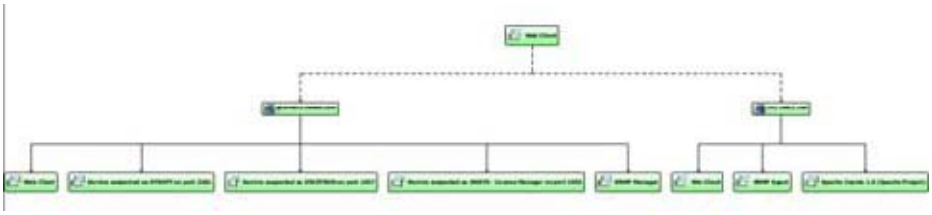
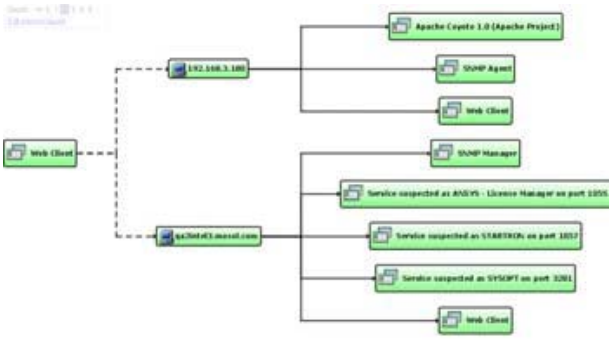
Style	Description	Sample Rendering
Normal	This layout does not change the node sizes. The drawing contains very few bends.	
Normal + Tree	Same as Normal, except it processes larger subtrees using a specialized tree layout algorithm, which is better suited for tree-like structures than the original orthogonal layout style.	
Uniform Node Sizes	Changes all node sizes to equal sizes before the processing the drawing.	
Node Boxes	Resizes nodes according to the number and position of their neighbors to reduce the overall number of bends.	

Style	Description	Sample Rendering
Mixed	Similar to Node Boxes, except it resizes all nodes to an equal size by introducing additional bends and routing the last line segment of these edges nonorthogonally to their adjacent nodes.	

Hierarchical Layout

The Hierarchical layout provides the style options shown in Table 6-8 for rendering the relationship layout. The resulting layout can differ from the examples shown because of the different structure elements, relationships, and layout options.

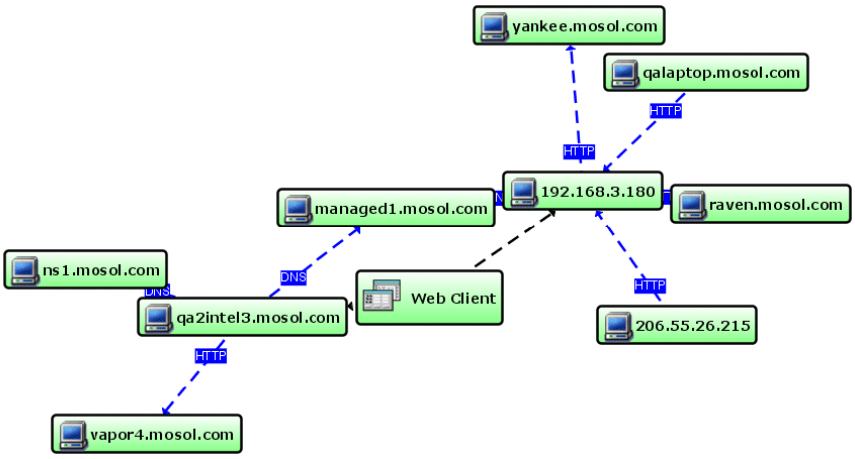
Table 6-8 *Hierarchical Layout Flows*

Layout Flow	Could render similar to …
Bottom to Top, Top to Bottom	
Right to Left, Left to Right	

Organic Layout

The Organic layout renders the layout based mainly on structures and relationships. Figure 6-5 is an example of a relationship layout rendered using the Organic layout type:

Figure 6-5 *Organic Layout*



6.6.2 Using a Different Layout Style in the Relationship browser

1. Do one of the following to open the Edit Element Layout Properties dialog box:
   - In the sidebar, click Edit Element Layout Properties.
   - In the Control Panel, click the Edit element layout link.
2. In the Edit Element Layout Properties dialog box, select a layout type in the sidebar (*Tree*, *Circular*, *Orthogonal*, and so on).
3. If necessary, modify the settings for the selected layout on the right side of the dialog box.

   These settings vary by layout type.

   For information on selecting the appropriate link, see "Using Layout Rendering Features" on page 109.
4. Click *OK*.

   The diagram updates to use the selected layout style.

6.6.3 Saving the Current Layout as an Element Preset

Layout style changes display for the current session only. Save them explicitly for future use.

To save the current layout as an element preset:

1. To save the current view as an element preset for future use, right-click the layout background, then select *Save Element Layout*.
2. To select and display the saved element preset, do one of the following in the Relationship browser:
   - In the sidebar, click Layout Preset, then select *Current Element* from the drop-down list.
   - In the Control Panel, click the current Layout selection, then select *Current Element* from the drop-down list.

   The diagram updates using the element preset.

6.7 Adding Elements

Use the Relationship Browse to add new elements to the *Services* hierarchy:

- Section 6.7.1, “Adding a Child Element to an Existing Element,” on page 117
- Section 6.7.2, “Adding Existing Elements to a New Element and Defining the Dependency Relationships,” on page 118
- Section 6.7.3, “Adding One or More Elements to an Existing Services Element,” on page 120

6.7.1 Adding a Child Element to an Existing Element

1. In the Relationship browser, right-click an element, then select *Add Element*.
2. Provide a name for the new element, then select an element class.
3. Click *Finish*.

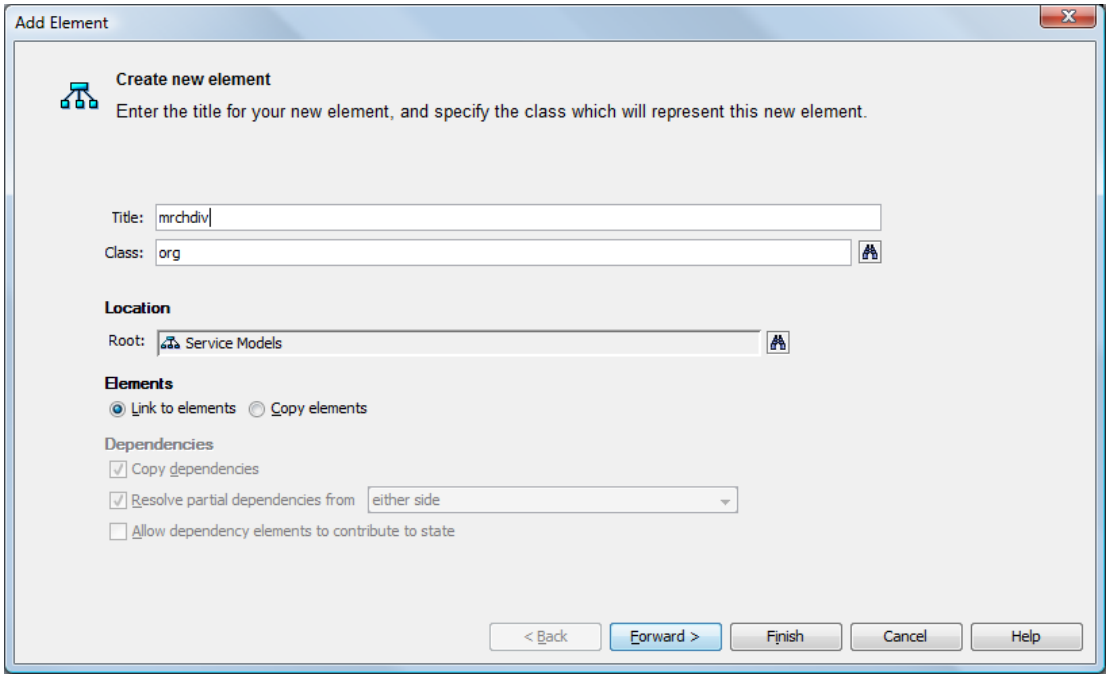
   The new element displays in the console Explorer pane. However, to view the new element in the Relationship browser, you might have to double-click the parent element, then select *Expand Children*.

6.7.2 Adding Existing Elements to a New Element and Defining the Dependency Relationships

The Add to feature enables defining dependency relationships between elements in different branches of the hierarchy. Based on customization, these relationships can contribute to state.

To add existing elements to new elements as define dependency relationships:

1. In the Relationship browser, select one or more (Shift+click) elements.
2. Right-click one of the selected elements, then select *Add To* > *New Element* to open the following dialog box:



3. In the *Title* field, specify a name for the new element.
4. Under *Location*, click (*Browse*), then select the root element where the new element should reside.
5. Under *Elements*, select one of the following radio buttons:

   Link to Elements: Creates a link to the source elements.

   Copy Elements: Copies the source elements to the new element.
6. If the *Copy Elements* radio button is selected, select the following check boxes under *Dependencies*:

   Copy Dependencies: Copies dependencies to the new service model. Selecting this check box activates the following two check boxes:

   - **Resolve Partial Dependencies From:** Partial dependencies are relationships in which the dependency does not exist for both elements. For example, the source could be dependent on the target element, but not vice-versa. Trace and resolve dependencies using one of the following methods:
     - **Either Side:** Resolves using either source (origin element) or target (termination element).
     - **Dependency Origin to End Point:** Resolves from source (origin element) to target (termination element) only.

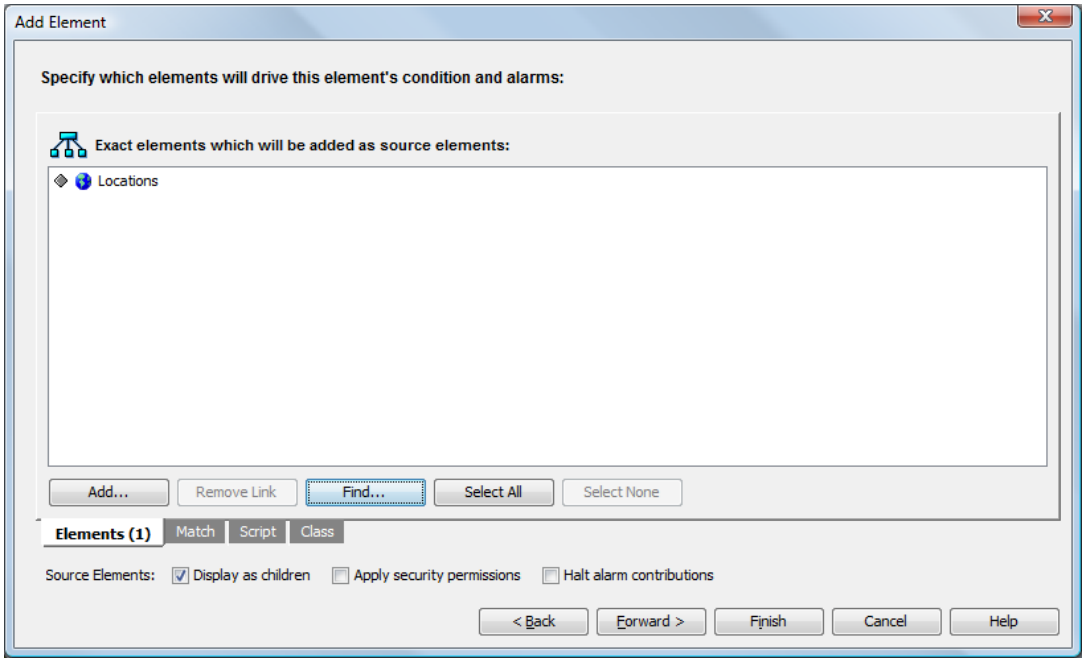
- **Dependency End Point to Origin:** Resolves from target (termination element) to source (origin element) only.

- **Allow Dependency Elements to Contribute to State:** The condition of source elements contributes to the state of the new element.

7 Click *Forward*.

The dialog box used to Specify which elements drive this element's condition and alarms opens. This dialog box and the subsequent dialogs are identical to those displayed when adding a new element in the *Layout* view.

Adding an element includes defining the *BSV* hierarchy, adding elements, and adding intelligence to the BSV by applying algorithms so that the state of the service can be defined. For details on the screens used to add a new element, see "Adding Child Elements to an Existing Service Model" on page 19.



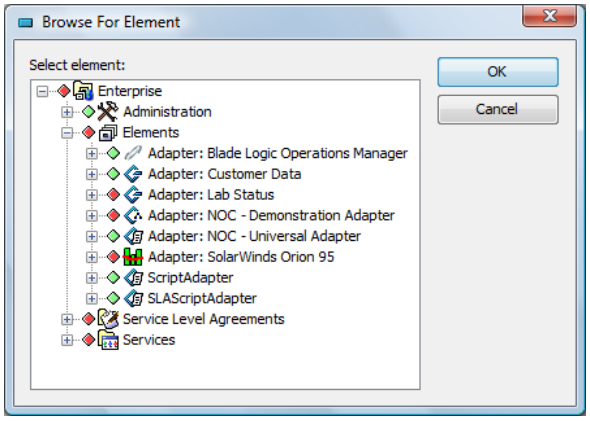
8 In the final screen, click *Finish*.

The new element displays in the *Services* hierarchy in the Explorer pane.

6.7.3 Adding One or More Elements to an Existing Services Element

1 In the Relationship browser, select one or more elements.

2 Right-click one of the selected elements, then select *Add To* > *Element* to open the Browse for Element dialog box.



3 Select the element, then click *OK*.

The selected elements display as links under this element in the Explorer pane:

A link symbol displays next to the added elements.



II Using the Service Configuration Manager

The Service Configuration Manager (SCM) allows you to dynamically generate new element hierarchies from multiple sources, to create custom end-to-end service views that enable users in different roles throughout the organization to assess the risk of change within their IT or business environment. It automates synchronization to ensure optimal data quality and real time visibility to changes in the IT infrastructure.

For example, a service configuration might:

- Combine two hierarchies from different sources that contain information about the same objects
- Apply trouble tickets to services
- Apply business metrics to services
- Show a hierarchy of correlated objects without state propagation that other service configuration hierarchies can leverage
- Find objects or services that are not managed
- Identify affected applications when a network object is down

The following sections provide details about using the Service Configuration Manager.

- Chapter 7, “Introduction to the Service Configuration Manager (SCM),” on page 123
- Chapter 8, “Creating a Service Configuration,” on page 125
- Chapter 9, “Use Case: Merging Configuration Management and Asset Databases,” on page 155
- Chapter 10, “Use Case: Mapping Application Dependencies,” on page 163
- Chapter 11, “Use Case: Correlating Network Objects,” on page 171
- Chapter 12, “Use Case: Automating Change Management,” on page 175
- Chapter 13, “Use Case: Reporting Service Configuration Problems,” on page 203

7 Introduction to the Service Configuration Manager (SCM)

Operations Center provides the Service Configuration Manager (SCM), which dynamically generates new element hierarchies from multiple sources. It automates Configuration Management Database (CMDB) synchronization to ensure optimal data quality and real time visibility to changes in the IT infrastructure. Customizable end-to-end service views enable users in different roles throughout the organization to assess the risk of change within their IT environment.

Use SCM to generate a CMDB that meets the following critical requirements:

- **Federation:** Brings in multiple data sources directly by linking to sources.
- **Reconciliation:** Avoids duplicates and enables matching of configuration items from different sources.
- **Synchronization:** Identifies changes to the CMDB as they occur, thereby ensuring consistent information across integrated systems.
- **Mapping and visualization:** Enables peer-to-peer and hierarchical views of the Configuration Items (CIs).

Review the following sections for an understanding of BCSM:

- Section 7.1, “What is Service Configuration Management?,” on page 123
- Section 7.2, “Key SCM Features,” on page 124
- Section 7.3, “SCM Methodology,” on page 124

7.1 What is Service Configuration Management?

Traditionally, configuration management dealt strictly with management and measurement at the component level. Today, service configuration management looks at all stages of the service lifecycle in order to provide information about an organization’s performance.

Managing service configuration is critical in understanding the lifecycle of a service. The Service Configuration Manager correlates views while tapping into discovery and mapping capabilities that provide new insight and visibility into the network, thereby providing the ability to manage the service and its lifecycle.

Service configuration management is composed of:

- **Configuration Management Database (CMDB):** Created by mining various configuration sources. A Configuration Management Database brings related information from multiple sources into one view, to create a CMDB without having to build a physical database.
- **Service Catalog:** Created by modeling business/technology services. A Service Catalog is a systematic list of service definitions and documentation complete with relationship mapping between services delivered as well as their underlying components.

- **Service Definition:** Augments the Service Catalog with dynamic links to underlying technology components.
- **Reports:** Assist with change/impact analysis, capacity planning, reconciliation, auditing, and data center consolidation or license usage.
- **Business Service:** Configuration alerts that foster proactive management.

7.2 Key SCM Features

The key features of SCM are:

- Auto-discovery that detects dependencies, baselines "normal," and identifies scheduled and unscheduled changes
- Integrated mapping of element relationships across an enterprise
- Integration of asset, configuration, and change sources into a CMDB
- Building and maintenance of Service Views automatically and dynamically, alleviating the need to create, update, and maintain views (models) manually

7.3 SCM Methodology

Implementing a service configuration is a multistep process that allows leveraging and integrating important data and technology into a unified view. This process leverages, reuses, and combines configurations to parse data and views in a customized way.

The process of implementing SCM can vary depending on requirements, but the basic process includes these key areas and their implementation steps:

1. **Technology Integration:** Define and create adapters to integrate technology, data, and discovery. This can include integrations with network and application management systems, trouble ticketing systems, discovery tools, and databases.
2. **Service Catalog:** Define Service Views using the *Services* hierarchy in the console.
3. **1st Generation Service Configuration:** Build a correlated CMDB to unify all objects by object name.
4. **2nd Generation Service Configuration:** Create additional SCM definitions that will leverage previous definitions to do any of the following:
    - Slice and dice the views in a functional way using class, purpose, or use case.
    - Link back to the correlated CMDB or combine hierarchical views as required. Use a discovery tool to reveal dependencies.

8 Creating a Service Configuration

Use the Service Configuration Manager to create a service configuration definition that generates a new hierarchical view for any service, and applies additional information available from other sources. Service configurations can also be used combine various views.

For example, a service configuration might:

- Combine two hierarchies from different sources that contain information about the same objects
- Apply trouble tickets to services
- Apply business metrics to services
- Show a hierarchy of correlated objects without state propagation that other service configuration hierarchies can leverage
- Find objects or services that are not managed
- Identify affected applications when a network object is down

Generally, the service configuration generates and populates a configuration hierarchy under *Services* by:

- Copying objects, data, and the hierarical structure from an adapter.
- Using Data Integrator definitions to leverage and apply information from management or configuration databases.

The following sections provides step-by-step instructions for creating and maintaining service configuration definitions:

- Section 8.1, “Understanding Multi-Definition and Multi-Generational Configurations,” on page 125
- Section 8.2, “Understanding Element Correlation Methods,” on page 126
- Section 8.3, “Creating Service Configuration Definitions,” on page 128
- Section 8.4, “Enabling Auditing, Debug, and Element Locking During Generation,” on page 151
- Section 8.5, “Managing Multiple Definitions,” on page 152
- Section 8.6, “Backing Up, Restoring and Deleting Definitions,” on page 152

8.1 Understanding Multi-Definition and Multi-Generational Configurations

It is possible to define more than one service configuration for a single element. The service configurations are built and applied in the order in which they are defined.

- Section 8.1.1, “Understanding Multiple Configurations,” on page 126
- Section 8.1.2, “Understanding the Generational Models Workspace,” on page 126

8.1.1 Understanding Multiple Configurations

When multiple configurations are defined for the same element, subsequent configurations can be generated using the results of the configurations defined before them. For an example that shows multi-definition configurations, see Section 12.3, “Use Case: Creating a Configuration Management Database,” on page 186.

It is also possible to use hierarchies generated from other configurations to help define or build a new configuration. These are called multi-generation configurations. A first generation configuration can contribute to (as a structure or source) a second generation configuration.

For an example of using one configuration to define another configuration, see Chapter 10, “Use Case: Mapping Application Dependencies,” on page 163, which leverages a *Service Catalog* hierarchy that a different service configuration previously generated.

8.1.2 Understanding the Generational Models Workspace

Models created under the *Services* node represent logical relationships among elements in a business. Models constructed under the *Service Models* node typically group elements in an enterprise that represent the various management activities or services related to a business. Models created under the *Locations* node group elements based on geographical location. Different users and groups of users are given permissions to view models based on operational need.

Whereas, the *Generational Models* node under the *Services* hierarchy, functions much like a “working space” for creating and maintaining models that can be viewed by a limited number of users and contain elements that do not have an impact on other service models.

Use *Generational Models* for configurations that are produced by SCM as part of the process of building service models. These would be the “building blocks” for a service model, and are not considered part of a service model.

Restricting access to some models is desirable for many reasons. For example, only the person creating a model should be able to view it during the creation process.

To restrict access to a model, construct it under the *Generational Models* node in the *Services* hierarchy, then set permissions for the *Generational Models* node to restrict access to only those users who have access to all the models under this node.

8.2 Understanding Element Correlation Methods

Correlating different types of elements from various sources can be accomplished using different methods. Consider the advantages and expected results when selecting a method of element correlation.

- Section 8.2.1, “Understanding Rule-Based Correlation,” on page 127
- Section 8.2.2, “Understanding Class-Based Correlation,” on page 127

8.2.1 Understanding Rule-Based Correlation

The rule-based correlation method uses SCM to copy element structures provided by CMDB sources with the Data Integrator, providing rule-based correlation in the SCM *Generation* policies to correlate structural data and monitoring sources (BSM).

This method involves creating service configuration definitions, as described in Section 8.3, "Creating Service Configuration Definitions," on page 128. The SCM modeling policies are used to generate rules that match and associate elements originating from different sources. As an example, a CMDB source can contain a list of hosts, and a list of applications that depend on hosts. The modeling policies correlate each host to a monitoring source, such as IBM Netcool Omnibus or CA Spectrum, associating element conditions and alarms with the resulting model. A potential drawback is that the SCM definitions used to generate these models can become large and unwieldy because each type of element requires a different set of rules to perform the correlation.

8.2.2 Understanding Class-Based Correlation

An alternative correlation method uses element class. Using this method, SCM generates an element structure by copying a BDI definition data source to the service model, and based on class, the Operations Center correlation engine automatically associates elements of specific classes. This approach uses element class relationship templates to perform correlation. The advantages of using class-based correlation:

- Simplifies the SCM definition by eliminating multistep definitions to generate correlation rules for different element types.
- Does not require relationship data to be stored as persistent information for each element. This can dramatically reduce the element and relationship storage requirements for a Operations Center implementation.

Many use cases exist, but a simple one involves correlating elements with same name but different classes. For example, assume that service model elements with the server class should be correlated with same-named elements with the hosts class, in a branch of the *Elements* hierarchy.

To accomplish this correlation, create a Class Relationship template for the server class, then select a branch in the *Elements* hierarchy that should be automatically correlated. This template includes a DName match template, similar to the Dynamic correlation FIXED policy in SCM (for more information, see "STEP 6: Select Element Correlation Options" on page 145).

A sample result is creating a server class element named deptserver11 in the *Service Models* hierarchy that automatically generates a static relationship with an element named deptserver11 in the *Elements* hierarchy. Element conditions and alarms are also correlated.

Class Relationship templates are defined as part of the Metamodel class element properties. For details on creating the templates and common use cases, see "Defining Class Relationship Templates" on page 72.

8.3 Creating Service Configuration Definitions

Create service configurations for any element under the *Services* hierarchy. Each generated service configuration is based on rules defined when creating the definition.

Each service configuration definition is comprised of five types of subdefinitions that determine how the hierarchy is shaped and built:

- **Structures:** When defining structures, you are defining what the service configuration will look like. An existing element hierarchy is modelled to build out the new service configuration's structure. For example, if using a BDI adapter or OpenView maps as the base structure, the hierarchy of the newly created service configuration looks just like the selected BDI or OpenView maps hierarchy. Based on matching rules, the object hierarchy is copied along with algorithms, icons, or menu options.
- **Sources:** When defining sources, you are defining how state and property information is applied to the service configuration's objects. Join rules are used to apply state and property information from an existing element hierarchy to the new service configuration objects. This can be a T/EC, BEM, or OpenView adapter or any other hierarchy that can provide state and property information to the configuration objects.
- **Dependencies:** When defining dependencies, you are defining various relationships that can be viewed in the Relationship browser from the service configuration's objects. Select points in the service configuration definition that rely on elements from other branches, or indicate implied relationships. These relationships may contribute to state of the service configuration. Dependencies can be defined using existing dependencies (consumers) or those generated from alarms (generators).
- **Modeling Policies:** When defining modeling policies, you'll set up options used when generating the service configuration. Generation options set up how state is propagated, property data is integrated, and how definitions contribute to the configuration. Correlation options are additional rules that determine which element are included in the new hierarchy. Scripting options allow a script to be run on the configuration elements during generation. Algorithm options allow inherited algorithms to be overridden by new algorithm policy definitions.
- **Schedule:** (Optional) Create a schedule to update and refresh the service configuration.

TIP: When building a complex configuration, subdivide the task into smaller steps. Build smaller configurations based on these smaller steps, which act as test modules, before building the complete configuration.

The high-level steps for creating a service configuration are:

1. Create a Service Configuration definition by naming the definition and selecting a general reconciliation policy.

   For instructions, see Section 8.3.1, "STEP 1: Create the Service Configuration Definition," on page 129.

2. Select structures for the configuration by selecting an existing hierarchy.

   For instructions, see Section 8.3.2, "STEP 2: Define Structures," on page 131.

3. Select sources that should supply state and property information to elements in the new structure.

   For instructions, see Section 8.3.3, "STEP 3: Define Sources," on page 132.

4. Define dependencies where the new configuration relies on other branches or relationships.

   For instructions, see Section 8.3.4, "STEP 4: Define Relationship Dependencies," on page 138.

5 Define modeling policies to determine how the configuration is generated and correlated. Also set up scripts to run or algorithms to apply to the configuration.

   For instructions on setting generation options, see Section 8.3.5, "STEP 5: Select Generation Options," on page 143.

   For instructions on selecting the type of correlation used with objects, see Section 8.3.6, "STEP 6: Select Element Correlation Options," on page 145.

   For instructions on defining a script to execute at generation-time, see Section 8.3.7, "STEP 7: Apply Scripts," on page 147.

   For instructions on applying custom algorithm rules, see Section 8.3.8, "STEP 8: Configure Custom Algorithms," on page 149.

6 Generate the configuration to determine if the generated tree meets expectations.

   For instructions, see Section 8.3.9, "STEP 9: Test and Generate the Configuration," on page 150.

   Verify that the configuration builds as expected in the Operations Center console. Make changes, then regenerate to make available to users.

7 Apply security permissions to specific element hierarchies to prevent unauthorized users from viewing them. Also prevent display of these elements in the Show Impacted dialog.

8 Create a schedule for the automatic refreshing or generation of the configuration.

   For instructions, see Section 8.3.10, "STEP 10: Schedule Updates (Regenerations) of the Configuration," on page 151.

8.3.1 STEP 1: Create the Service Configuration Definition

The Service Configuration Editor is used to create service configuration definitions. Service configurations can be defined from any existing service model elements or as part of the process when adding a new element to the *Service Models* hierarchy.

To create a service configuration:

1 In the Explorer pane, expand *Services* > *Service Models*.

2 Do one of the following:

- To create a service configuration from an existing service model, right-click a service model element, then select *Service Configuration* > *Create*.
- To create a service configuration on creation of a new service model, right-click the parent element, then select *Add Element*. When defining the element definition, select the *Launch Service Configuration Editor After Element Creation* check box. After the service model is created and property pages are closed, the Configuration Editor automatically launches.

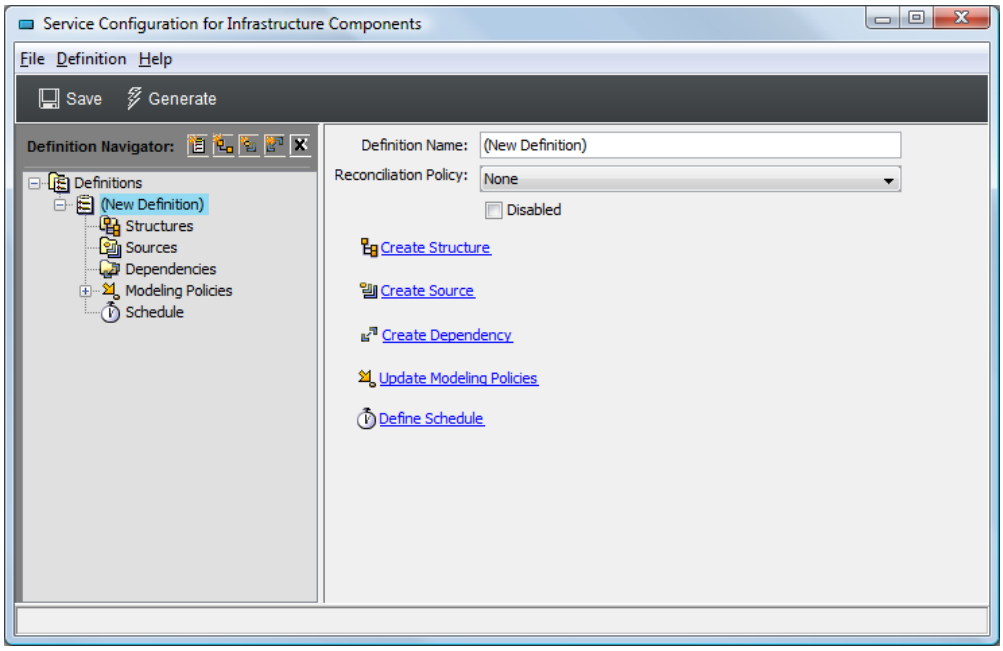
3 In the confirmation dialog box, click one of the following buttons:

   Yes: Creates a service configuration. If developing a configuration for the first time, consider enabling the debug features.

   No: Opens the editor settings for debug, auditing, and element locking features.

   For more information, see Section 8.4, "Enabling Auditing, Debug, and Element Locking During Generation," on page 151.

The Service Configuration Editor opens:



4 Specify a service configuration name in the *Definition Name* field.

5 From the *Reconciliation Policy* drop-down list, select a method for reconciling elements when generating the configuration:

None: Overwrites elements without deleting old elements first.

Merge: Combines the new configuration with any existing structure.

Delete-Before-Execute: Removes any existing structure before generating the new structure.

WARNING: Carefully consider the Reconciliation Policy in use when generating configurations, while users are actively using views. The Delete-Before-Execute policy is the most disruptive to active users. However, in development mode, this is the most desirable option because the new hierarchy is created from scratch instead of from merging existing elements.

---

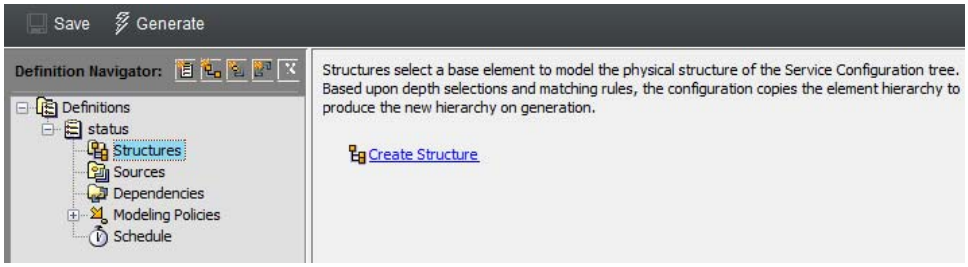
6 Click Save.

8.3.2 STEP 2: Define Structures

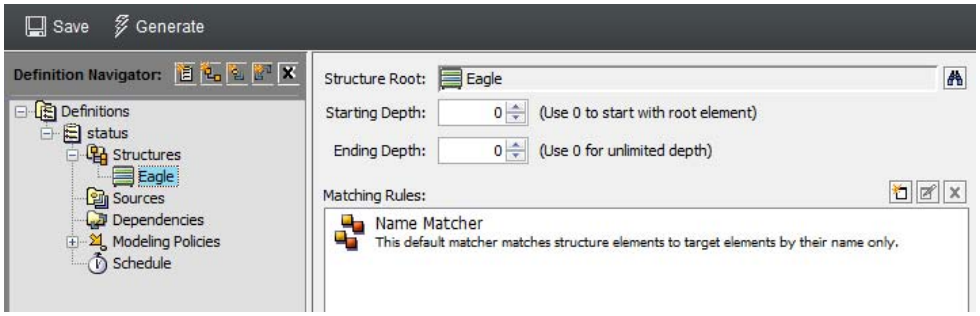
Model the new hierarchy on an existing element hierarchy. Structures copy existing element branches to form a new hierarchy. Use hierarchy depth selections and matching rules to select elements of interest.

To create a structure definition:

1. In the Explorer pane, right-click a service model element for which you have created a service configuration, then select *Service Configuration* > *Edit* to open the Service Configuration Editor.
2. In the Service Configuration Editor Definition Navigator pane, click (*New Structure*).



A new Structure definition displays in the hierarchy. The definition remains unnamed until the *Structure Root* is selected.



3. Click (*Browse for Root Element*) to select an element for the *Structure Root*.

   Select an element whose child structure should be used as a model for the new hierarchy.

   After selecting the *Structure Root* element, the definition element in the Definition Navigator updates to display the selected root element.
4. Use the *Starting Depth* spinner or specify the first level of children for the structure to include under the selected *Structure Root*.

   Leave *Starting Depth* at 0 to include the selected root element.
5. Use the *Ending Depth* spinner or specify the deepest level of children to include under the selected *Structure Root*.

   Leave *Ending Depth* at 0 for to include all levels of children.

6 To create a matching rule, click (*New Rule*), then complete the Add Match wizard.

Matching rules are used in multiple areas of Operations Center. For a complete explanation of creating and applying matching rules, see "Assigning Elements by Matching Properties" on page 26.

The Matching Rules specify criteria for selecting elements in the source hierarchy as candidates for the join rules (explained in the next step). The default Name Matcher uses the element name. However, it is also possible to match element DName or properties.

7 Click *Save*.

8.3.3 STEP 3: Define Sources

After defining the element structure, define sources for obtaining state and property information for the elements. Use join rules to apply state and property information from a source hierarchy to the new service configuration.

There might be situations where no source definitions are defined. These include configurations that:

- Derive source information from the structure elements. For more information, see "STEP 5: Select Generation Options" on page 143.
- Show structure only, by design. In this case, there is no need to expose any state information.
- Are defined only for their structure, which is leveraged by other (second generation) configurations. In this case, element state is configured for the second generation configuration hierarchies.

The source definition consists of existing elements that provide state and property information to the new service configuration. The source can be a T/EC, BEM, or OpenView adapter or any other element hierarchy that can provide source information.

IMPORTANT: When a configuration copies properties from structure elements while requiring a match in the join rule, Structures and Sources must not be defined in the same definition. The first definition must define Structures, while Sources are defined in a second and separate definition. The join rule is defined inside the Sources definition.

For more information on join rule settings, see "Using Join Rules" on page 134. For more information on copying properties from structure elements, see Section 8.3.5, "STEP 5: Select Generation Options," on page 143.

To create a source definition:

1 If you'll be using the *Require Match Between Structure and Source Elements* option in the Source join rule AND want to copy properties from the Structure elements, create a new definition.

For information on join rule settings, see "Using Join Rules" on page 134. For information on copying properties from structure elements, see Section 8.3.5, "STEP 5: Select Generation Options," on page 143.

1a In the Service Configuration Editor, click (New Definition) to create another configuration definition.

The new definition element is added beneath the existing definition in the *Definition Navigator* tree.

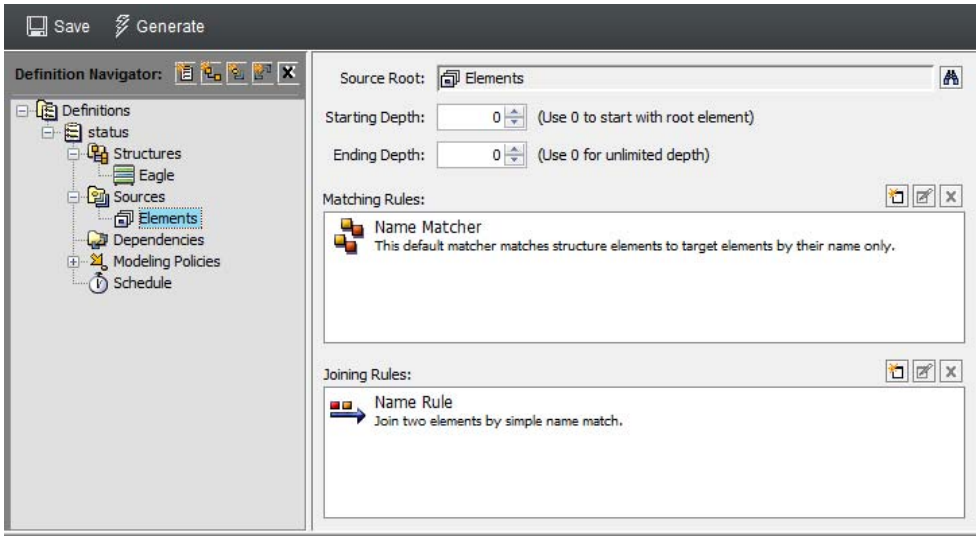
1b Enter the new definition name.

Note that illustrations in the following steps do not reflect the requirement of needing two definitions.

2 In the Definition Navigator pane, click (*New Source*).



A new Source definition displays in the *Definition* hierarchy. The Source definition remains unnamed until the *Source Root* is selected.



3 Click (*Browse for Root Element*) to select the *Source Root*.

Select any element that can supply state or property information.

4 (Optional) Restrict the source by specifying a start and end hierarchy level in the *Starting Depth* and *Ending Depth* fields.

Leave *Starting Depth* at 0 to include the selected root element.

Leave *Ending Depth* at 0 for to include all levels of children.

5 To create a matching rule, click (*New Rule*), then complete the Add Match wizard.

Matching rules are used in multiple areas of Operations Center. For a complete explanation of creating and applying matching rules, see "Assigning Elements by Matching Properties" on page 26.

The Matching Rules specify criteria for selecting elements in the source hierarchy as candidates for the join rules (explained in the next step). The default Name Matcher uses the element name. However, it is also possible to match element DName or properties.

6 To create a joining rule, click (*New Rule*), then complete the Add Join Rule dialog.

Join rules specify how to match the source elements and the element in the new configuration. The default rule matches element names. You can also match elements by property, script, hostname, or substring name. Rules are evaluated separately. If a match is found, the rule applies and a join occurs.

For more information, see "Using Join Rules" on page 134.

7 Click *Save*.

Using Join Rules

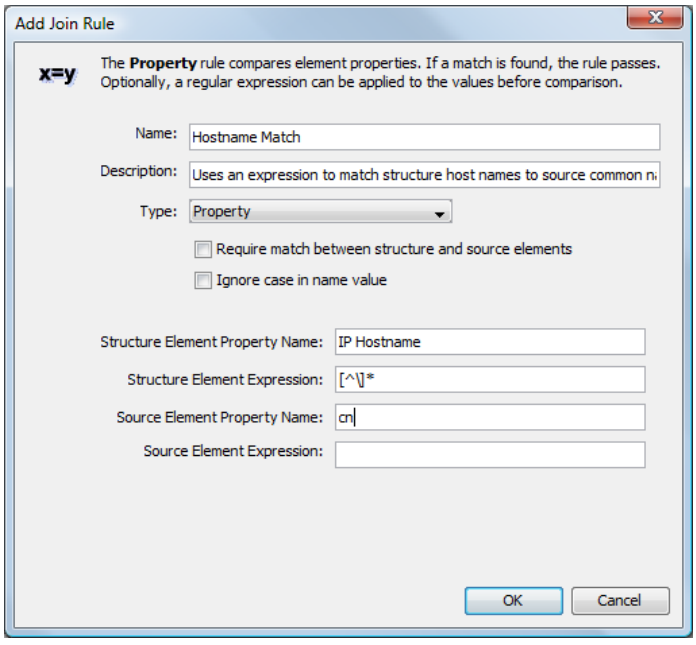
The join rules specify how to correlate source elements with elements in the new configuration. The default rule matches element names. You can also correlate elements by property, script, hostname, or substring name. Rules are evaluated separately. If a match is found, the rule applies and a join occurs.

- "Creating Join Rules" on page 134
- "Understanding Expressions Used in Join Rules" on page 135

Creating Join Rules

To create a new joining rule:

1 Click (*New Rule*) to open the Add Join Rule dialog box.



2 Select the type of rule from the *Type* drop-down list.

Name: Compares element names. Optionally, a regular expression can be applied to the values before comparison.

Property: Matches element properties based on the property names specified for both the source and structure. Optionally, a regular expression can be applied to the values before comparison.

Script: Passes the structure and source arguments to the script to evaluate if a join should occur. The last line of the script must indicate a True/False response.

Host Name: Compares element names, taking into consideration simple hostnames and fully qualified hostnames (FQDN).

Substring Name: Compares element names to determine if one name is contained within or is the same as the other.

The other fields in the dialog box change accordingly.

3 The fields in the Add Join Rule dialog box change depending on the type of join rule. Fill in the fields on the dialog box for the rule type.

For example, these fields are common to all join rule types:

- **Require match between structure and source elements:** Select this option to apply all specified Matching Rules when applying the join rules. When selected, structure elements are not copied unless a match occurs. Clear the option to ignore the Matching Rules and only apply the Join Rules. When cleared, all structure elements are copied based on Join Rules only.

  IMPORTANT: If you select this option to require a match, and intend to copy properties from structure elements, you must define two separate definitions the Service Configuration to handle structure and source. Use the first definition to define the structure, then create a new definition to define sources. The join rule with this restriction is then set for the source inside the second definition.

  For more information on copying properties from structure elements, see Section 8.3.5, "STEP 5: Select Generation Options," on page 143.

- **Ignore case in name value:** Select this option to ignore the letter case when comparing source and structure criteria. Clear the option to consider the letter case.

Structure Element Expression and *Source Element Expression* fields apply to Name and Property type join rules. Use regular expressions to perform a query before applying the join rules. For more information about using expressions, see "Understanding Expressions Used in Join Rules" on page 135.

4 Click OK.

Understanding Expressions Used in Join Rules

There is an option to query the structure and source elements using a regular expression or a macro expression before applying a name or property type of join rule.

Macro expressions provide increased flexibility in defining query text. They provide embedding control statements and variable substitution in query statements. Macro expressions use Velocity syntax. Velocity is a third party package provided by the Apache Organization and bundled into Operations Center. Documentation for Velocity is available at http://velocity.apache.org/engine/devel/user-guide.html (http://velocity.apache.org/engine/devel/user-guide.html).

Some of the Velocity syntax rules are:

- No spaces are used in macro expressions.
- The notation for a variable consists of a leading "$" character followed by a Velocity Template Language (VTL) identifier. A VTL identifier must start with an alphabetic character in curly brackets.

- Characters in a macro expression are limited to:
  - Alphabetic characters (a–z or A–Z)
  - Numeric characters (0–9)
  - Hyphen (–)
  - Underscore (_)

Use expressions to construct property lookups based on either parent or child element names. For example, use $parent.name to return the name of a parent element, then look up a specific property. Table 8-1 on page 136 shows parameter variables specific to Operations Center.

Table 8-1 *Operations Center Parameters*

Variable	Returns
`${parent.name}`	The name of the parent element. To traverse the tree to return values for non-immediate parents, add additional `parent.` declarations for each parent level. For example to return the name of the grandparent, use `$parent.parent.name`; and for two levels higher, use `$parent.parent.parent.name`.
`${parent.class}`	The class of the parent element.
`${sourceElements}`	The names of a service model's linked source elements.
`${sourceElementProperties}`	A comma-delimited list of all property values for service model's linked source elements. See the `${childrenProperties}` entry below for an example of a property lookup.
`${children}`	The names child elements.
`${childrenProperties}`	A comma-delimited list of all child property values. For example, the children of an element have a `Location` property, and there are three children having the following Location values: `Chicago`, `Washington`, and `Miami`. Use the expression `$childrenProperties.get("Location")` to return the value `Chicago,Washington,Miami`.

Expressions are useful when there is a need to correlate child element data containing multiple values that are a number of levels beneath the joining point of the source and structures element hierarchies.

A common use case involves the existence of multiple IP addresses on a host element. Assume there is a need to correlate a structure element, which has one or more IP address values (either in its name or a property), with a source element that intersects one of these IP address values (in either its name or property). An additional factor could be that the value of a property required to correlate an element resides in a fixed location under the element structure.

The following is an example source element hierarchy. Multiple IP addresses are elements modeled under the host element (Host A).

```
Host A
   Application
      Interfaces
         IP:206.55.26.1
         IP:206.55.26.2
         ...
```

To join this with a structure element named "206.55.26.1" requires joining at the `IP: 206.55.26.1` element. However, the actual join point is `Host A`, which is two levels up. The solution requires using a combination of the $childrenProperties and a regular expression to extract IP addresses from specific properties.

The property join rules allow using a regular expression to extract data from a string. Assume the structure contains the property "name" that contains multiple IP addresses as property values. The first part of the solution is to use the following regular expression:

```
\d{1,3}\.\d{1,3}.\d{1,3}.\d{1,3}
```

This expression obtains the first IP address from a string. For example, in the string `IP: 206.55.26.1`, it returns `206.55.26.1`.

To extract multiple values (multiple IP addresses in this example), use a group expression by placing parentheses around the original expression:

(\d{1,3}\.\d{1,3}.\d{1,3}.\d{1,3})

The group operator locates and joins all matching IP values.

This solution can extract repetitive groups, such as multiple IP addresses in a property that are comma-delimited or space-delimited. For example, if a string contains multiple IP addresses, in the form of `192.168.1.1,192.168.1.3` or `192.168.1.1|192.168.1.3` or `192.168.1.1 192.168.1.3` (or any other delimiter), apply the regular expression (\d{1,3}\.\d{1,3}.\d{1,3}.\d{1,3}) to these string values to return two group values that are used to intersect the join.

The second part of the solution looks up a property located in a fixed location in the source element hierarchy. Use ${children} and ${childrenProperties} to extract the IP address values. Using the example element structure listed above, use this expression:

```
${children.get('Application').children.get('Interfaces').childrenProperties.get('n
ame’)}
```

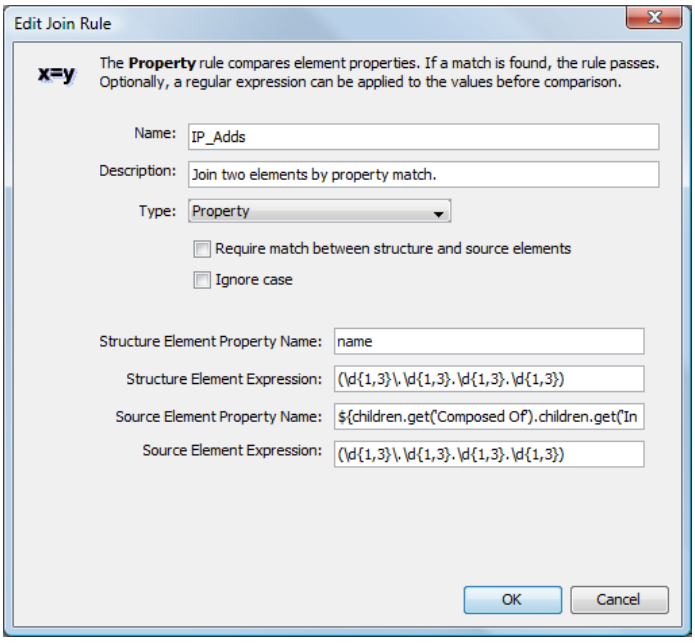
This returns a comma-delimited list of the names (such as IP: 206.55.26.1, IP: 206.55.26.2).

In summary, the following expressions are used to correlate child element data and multiple values:

- Use ${childrenProperties} to extract the IP address values embedded in the names within the element structure. This returns a comma-delimited list of names (such as IP: 206.55.26.1 or IP: 206.55.26.2).
- Uses the grouping regular expression (\d{1,3}\.\d{1,3}.\d{1,3}.\d{1,3}) to return the two values (or any number of values).

Figure 8-1 shows both expressions entered in the Edit Join Rule dialog box:

Figure 8-1 *Edit Join Rule Dialog Box*



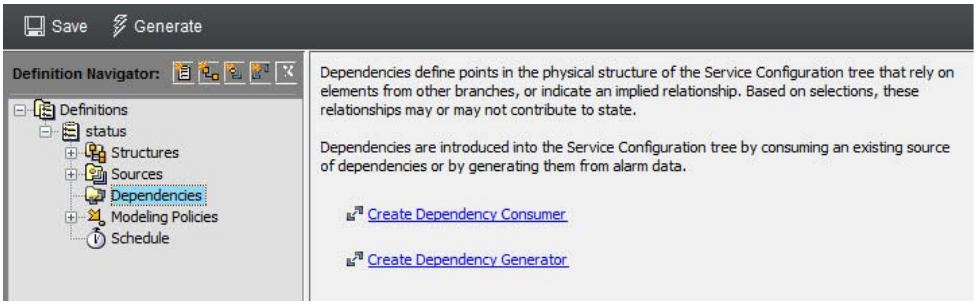
This join rule matches IP addresses between the source and structure using a combination of the ${childrenProperties} and a regular expression to extract IP addresses from specific properties.

8.3.4 STEP 4: Define Relationship Dependencies

Dependencies define points in the new hierarchy that rely on elements from other branches, or indicate an implied relationship. Based on selections, these relationships could contribute to state.

Dependencies defined relationships that can be viewed in the Operations Center Relationship browser. To open the Relationship browser, right-click an organization, then select *Show Relationships*. For more information about using the Relationship browser, see Chapter 6, "Understanding Element Relationships," on page 95.

Figure 8-2 *Service Configuration Editor: Defining Dependency Definitions*



To define the types of dependencies, review the following sections:

- "Using Dependency Consumers" on page 139

  Defines relationship based on an existing dependency source.
- "Using Dependency Generators" on page 141

  Defines relationships based on associations from alarm data.

Using Dependency Consumers

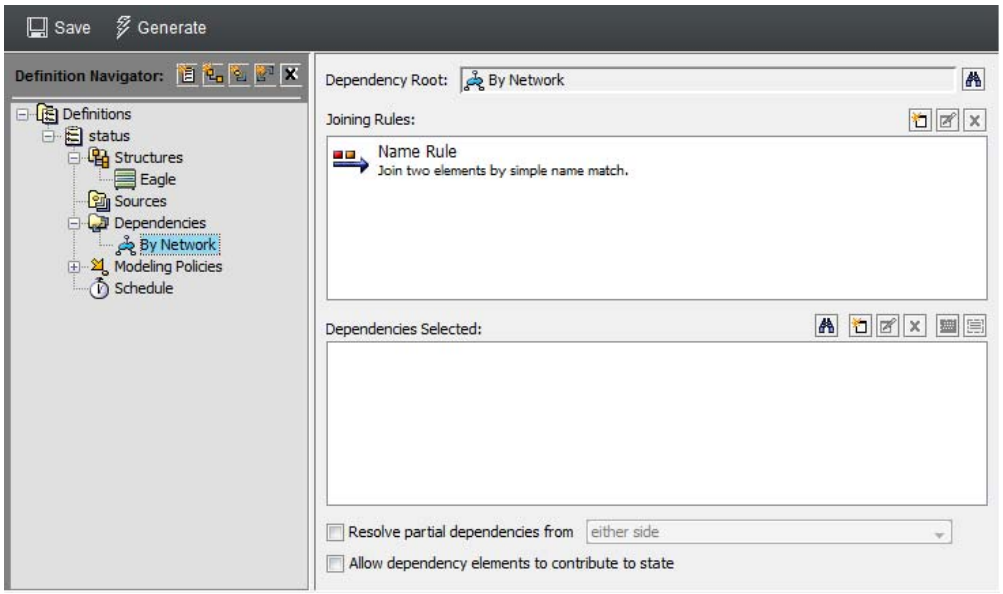
Use dependencies currently available in a hierarchy to further define the configuration. Many discovery tools look for and save information about dependencies in the network. Use a dependency consumer definition to select from and leverage any known dependency relationships.

- "Creating a Dependency Definition Using Existing Dependencies" on page 139
- "Selecting Dependency Relationships to Include in the New Hierarchy" on page 140
- "Adding a New Dependency Element" on page 141

Creating a Dependency Definition Using Existing Dependencies

1 In the Definition Navigator pane, do one of the following:

   - Click  (*New Dependency*).
   - Click the *Dependencies* element, then click the *Create Dependency Consumer* link.



   A new dependency definition displays in the *Definition* hierarchy. It remains unnamed until the *Dependency Root* element is selected.

2 Click the *Dependency Root* (*Browse*) to browse, then select the base element for the dependency rules.

3 Add, edit, or remove join rules as necessary.

Join rules specify where to merge elements from the *Dependency Root* tree into the configuration.

Specify join rules matching using Name, Property, Script, Host Name, or Substring Name.

For more information, see "Using Join Rules" on page 134.

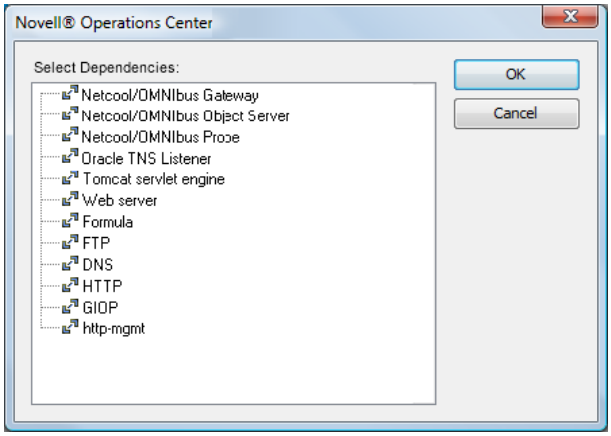
4 Click *Save*.

Selecting Dependency Relationships to Include in the New Hierarchy

The next step is to select dependencies from the selected *Dependency Root* element:

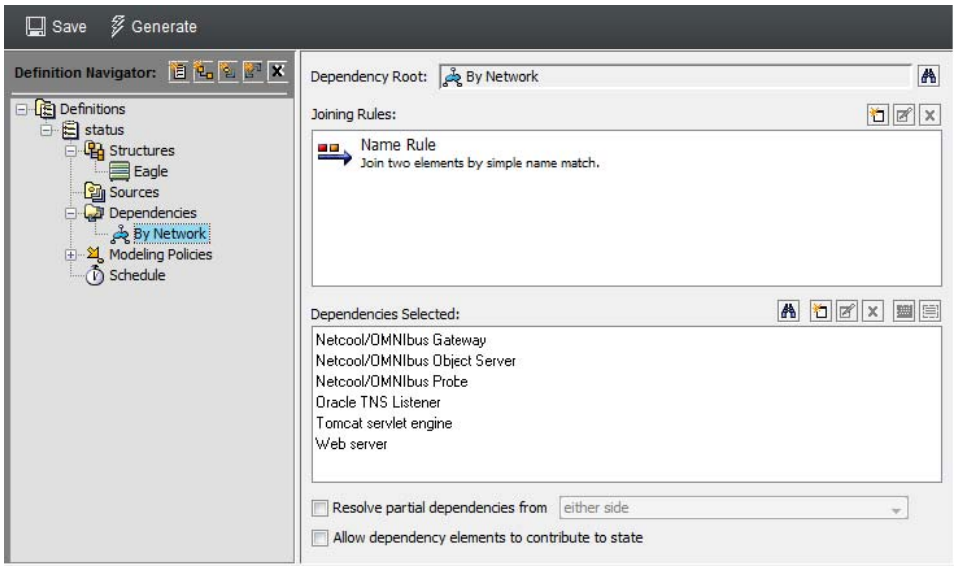
1 In the *Dependencies Selected* section, click (*Browse for Dependency*).

A dialog box displays existing dependencies for the selected *Dependency Root* element:



2 Select one or more dependencies, then click *OK*.

The selected dependencies display in the *Dependencies Selected* section:



3 Select the *Resolve Partial Dependencies From* check box, then specify the type of dependencies to capture:

Either side: Resolves using either source (origin element) or target (termination element).

Source to target: Resolves from source (origin element) to target (termination element) only.

Target to source: Resolves from target (termination element) to source (origin element) only.

Partial dependencies are relationships in which the dependency does not exist for both elements. For example, the source might be dependent on the target element, but not vice-versa.

4 To allow dependency elements to contribute to state, select the *Allow Dependency Elements to Contribute to State* check box.

5 Click *Save*.

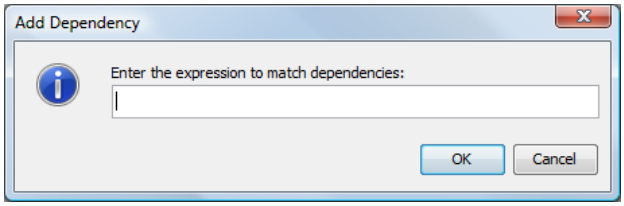
Adding a New Dependency Element

The alternative to selecting known dependencies is defining a new dependency.

To add a new dependency element:

1 In the *Dependencies Selected* section, click (*New Dependency*).

2 In the field, enter an expression that matches dependencies, then click OK.



3 (Conditional) To edit the expression used to select a dependency, select the dependency from the *Selected Dependencies* list, then click (*Edit Dependency*).

4 (Conditional) To delete a dependency, select a dependency from the *Selected Dependencies* list, then click (*Delete Dependency*).

Using Dependency Generators

If no dependencies are available, create dependencies using alarm information from any source. Matching an alarm property with an element property establishes element relationships that previously were unknown.

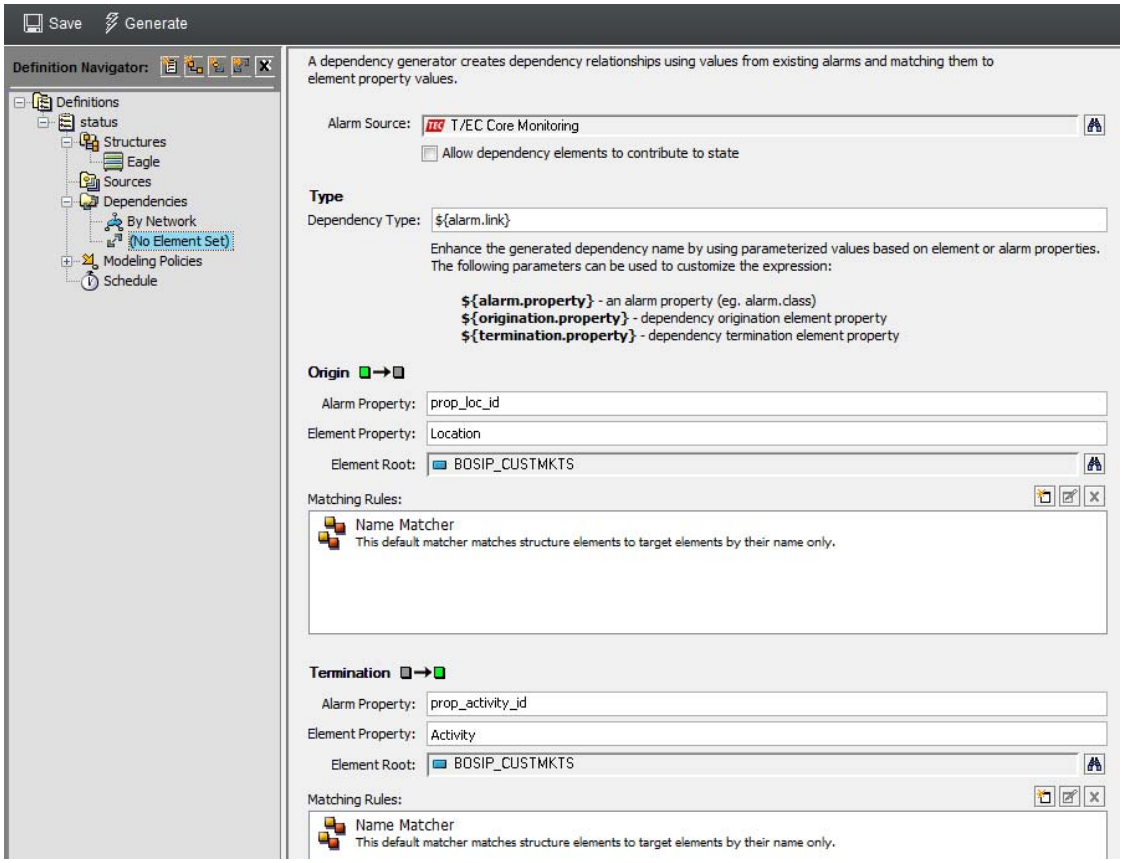
Each dependency has an origin and a termination. In other words, this establishes the element where the dependency starts (From) and the element where the dependency ends (To).

To say that a *Financial Application* is dependent on *Production Server* means that the dependency relationship exists from *Financial Application* (origin element) to *Production Server* (termination element).

To create a dependency definition from alarm information:

1 In the Definition Navigator pane, do one of the following:

- Click (*New Dependency*).
- Select the *Dependencies* element, then click the Create Dependency Generator link.



A new dependency definition displays in the definition hierarchy. It remains unnamed until the *Alarm Source* is selected.

2 Click (*Browse*) to select the element whose alarms are the *Alarm Source*.

3 Select the *Allow Dependency Elements to Contribute to State* check box to allow matched dependency elements to contribute to the state of configuration elements.

4 Add detail to the dependency names by including a property in the dependency naming convention.

5 In the *Dependency Type* field, specify an alarm or element property using the syntax as described in the dialog box.

6 Use the *Origin* settings to define where and how the dependency originates:

Alarm Property: In the field, specify the alarm column name that matches the Element Property for matched elements.

Element Property: In the field, specify the element property name that matches the Alarm Property value.

Element Root: Click (*Browse*) to select the *Element Root* where elements are selected for matching *Element Property* values with *Alarm Property* values.

Matching Rules: Define matching rules in the *Matching Rules* list. Matching rules select elements under the *Element Root* to use for dependencies. For more information, see "Creating a Matching Rule" on page 65.

7 Define dependency Termination settings, which determine how the dependency ends.

For information on setting definitions, see the previous step.

8 Click *Save*.

8.3.5 STEP 5: Select Generation Options

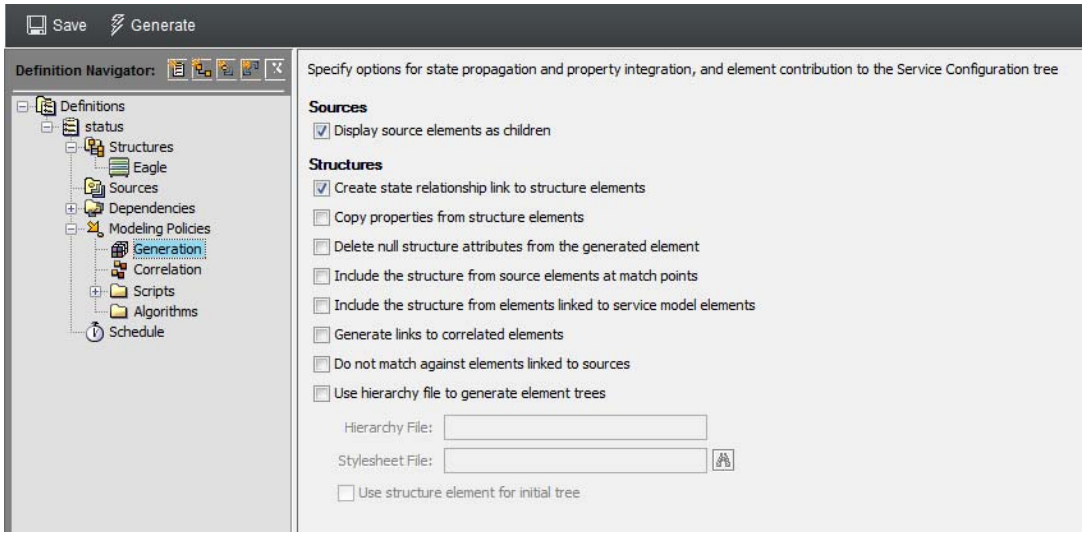
Use *Modeling Policies* > *Generation* options to specify advanced features that define how to generate the new configuration structures and how to apply source and property data. These include options for propagating state, integrating property data, and deciding how definitions contribute to the configuration.

For example, source elements can automatically display as children in the new configuration. Under certain circumstances, choose to derive property, and state calculations from structures instead of sources.

To set options for generation of the service configuration hierarchy:

1 In the Definition Navigator pane, expand *Modeling Options*.

2 Click *Generation*.



3 Select the *Display Source Elements As Children* option to include source elements in the new configuration hierarchy as linked children of the service model.

This feature is the same as the *Display as Children* option in the Elements property page for a service model element. For more information, see Section 3.1.1, "Assigning Elements to a Service Model," on page 26.

4 Select one or more of the following check boxes, to indicate how the structure of the configuration is to be generated and what attributes are copied:

Create state relationship link to structure elements: Select this option to apply state propagator mechanisms from structure elements. When selected, configuration objects are linked to the original structure elements in order to inherit state information from them.

Copy properties from structure elements: Select this option to copy properties from original structure elements. When selected, but not the above `Create state relationship link to structure elements` option, the properties, but not states, are inherited from the structure's elements.

IMPORTANT: If copying properties from structure elements AND you have selected *Require Match Between Structure and Source Elements* for any Source join rule, Sources must be defined in a new and separate definition from Structures. In this case, this option is selected in the generation options for the first definition set up for Structures.

For more information on defining sources, see "STEP 3: Define Sources" on page 132. For more information on join rule settings, see "Using Join Rules" on page 134.

Delete null structure attributes from the generated element: When selected, element properties with a null value are not copied from structure elements.

Include the structure from source elements at match points: Select this option to build a structure from both source and structure elements when the join rule matches. When selected, at the point the join rule is found to be true, the new configuration structure is built by combining the object hierarchies from the base elements as defined in both *Structures* and *Sources*.

Include the structure from elements linked to service model elements: Select this option to include any linked elements found under the base elements defined in *Structures*. This applies only when a Service Model hierarchy is identified in the *Structures* subdefinitions. Keep this option unselected to include only service models elements that are not linked objects.

Generate links to correlated elements: Select this option to create linked objects of correlated elements instead of copying them to build the new configuration.

Do not match against elements linked to sources: Select this option to exclude any linked children under source elements. When selected, only direct children are matched.

5 Select the *Use Hierarchy File to Generate Element Trees* check box to implement a MODL file to build the configuration hierarchy.

A MODL file can be used to generate matches to target elements using structure or source values. MODL is the Managed Object Definition Language.

For more information on using MODL, see the *Operations Center 5.5 Adapter and Integration Guide*.

5a **Hierarchy File:** Select a hierarchy file that contains an XML description of the hierarchy of elements to build below the configuration element. Specify a relative filename in the `/OperationsCenter_install_path/database` directory.

Sample hierarchy files can be found in the `/OperationsCenter_install_path/database/examples/BSCM` directory.

5b **Stylesheet File:** Click (*Browse*) to select a stylesheet file.

5c **Use Structure Element for Initial Tree:** Select the check box to apply hierarchy file elements to bottom of structure elements.

6 Click *Save*.

8.3.6 STEP 6: Select Element Correlation Options

Modeling Policies > *Correlation* uses matching rules to determine which elements are included in the element hierarchy generated by a service configuration.

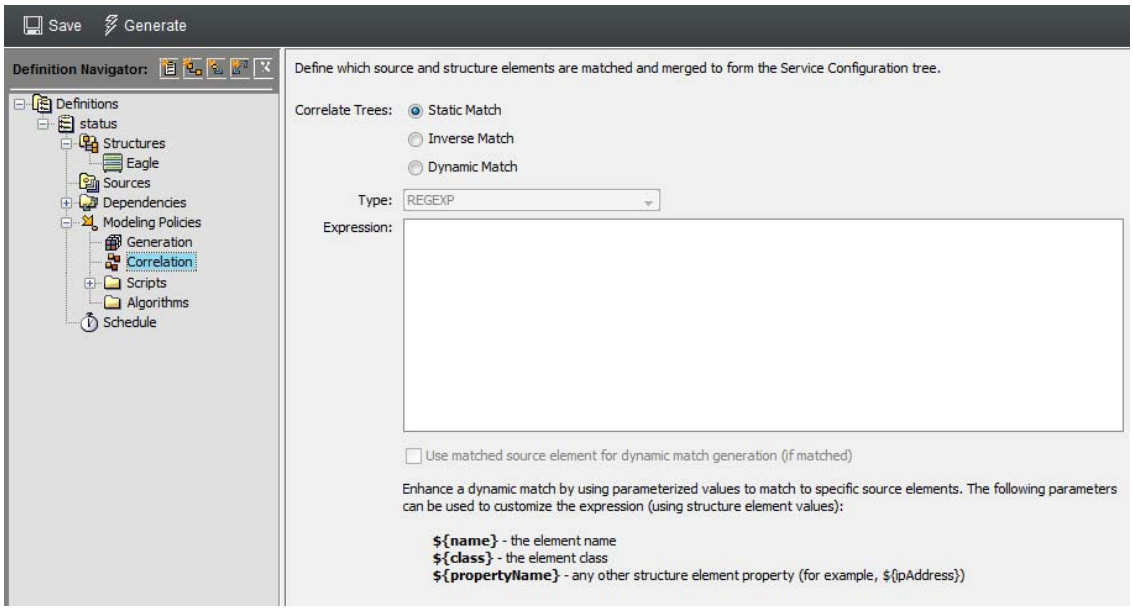
By default, *Static* correlation is used and applied to all elements that meet the *Structures*, *Dependencies*, and other *Modeling Policies* rules in the definition of the service configuration. Meaning, all matched elements are included.

As an alternative, correlation type can be set to *Inverse* so that the new configuration is built including only those elements that do not matching rules. All matched elements are excluded.

For more complex correlation needs, *Dynamic* matching uses expressions that generate matches based on values extracted from structure and source elements, to determine which elements are included in the output hierarchy.

To define how elements in the new configuration are correlated:

1 In the Definition Navigator pane, click the *Correlation* element under *Modeling Policies*.



2 Select one of the following options under *Correlate Trees* to define how the new configuration is correlated:

- **Static:** *Static* matching is the default. All elements that meet the *Structures*, *Dependencies*, and other *Modeling Policies* rules in the service configuration definition are included in the outputted hierarchy.
- **Inverse:** *Inverse* matching is basically the opposite (or "not" version) of static matching. It means that all elements that meet the Structures (and optionally Sources) rules but do not meet all other rules of the definition are included in the output hierarchy. This option is typically used to create or find elements that are missing.
- **Dynamic:** Select *Dynamic* matching to define an expression template that generates a match based on values extracted from structure and source elements, to determine which elements are included in the output hierarchy. The types of expression templates are as follows:
  - **REGEXP:** Selects elements based on a regular expression.
  - **LDAP:** Selects elements based on an LDAP expression.

- **Script:** Selects elements based on a script written using the NOC Script language. The script must evaluate to either True or False for each element.
- **Fixed:** Performs the same as static matching except that you can add parameters to further define elements included in the output hierarchy.

When *Dynamic* matching is selected, there is an option to restrict matching to only the elements specified under *Sources* (for more information on sources, see "STEP 3: Define Sources" on page 132). To match only the base elements, select the *Use Matched Source Element for Dynamic Match Generation* check box.

For more information on defining expression templates, see "Understanding Expression Template Parameters" on page 146.

3 Click *Save*.

Understanding Expression Template Parameters

The following parameters can be used in an expression template:

- ${name} for element name
- ${class} for element class
- ${propertyName} for any other property of an element for which correlation applies, such as ${IPAddress} for the IPAddress property of an element
- ${parent.name} to look up the value of the parent to further define the expression
- ${parent} to allow for the template to introduce parent values to the resulting match for any property, such as ${parent.class} or ${parent.propertyName}

Parameters must be specified using Velocity syntax. Operations Center uses Velocity, which is a project of the Apache Software Foundation, as a macro processor and template language. Documentation for Velocity is available at http://velocity.apache.org/engine/devel/user-guide.html (http://velocity.apache.org/engine/devel/user-guide.html).

Some of the syntax rules are:

- No spaces are used in macro expressions.
- The notation for a variable consists of a leading "$" character followed by a Velocity Template Language (VTL) identifier. A VTL identifier must start with an alphabetic character.
- Characters in a macro expression are limited to:
  - Alphabetic characters (a–z or A–Z)
  - Numeric characters (0–9)
  - Hyphen (–)
  - Underscore (_)

Because the template language is Velocity, the template can use standard java string manipulation functions to customize the output. A common use is to apply a regular expression to transform a part of the string:

```
${name.replaceFirst('\..*', '' )}
```

This macro takes the value of the element name, and replaces all values from the first period to the end of string with the empty string (''). This is useful for shortening the name to an unqualified hostname, for example.

In addition, a common use of the `${formula.util()}` macro expansion is to encode a DName component with URL encoding syntax:

```
${formula.util.encodeURL($name)}
```

This is required for formatting a DName in the proper match output to escape spaces into + characters. This, however, must be combined with a `replaceAll()` macro to turn the + characters into non regular expression characters:

```
${formula.util.encodeURL($name).replaceAll('+','\\+')}
```

In this example, if the input `${name}` has the value of `One, Two, Three`, the output should end up as `One,\+Two,\+Three`, first by passing through the URL encoding to produce `One,+Two,+Three`, then the `replaceAll` to achieve the final result.

Multi-generational SCM definitions now allow the destination element of a SCM definition to appear in the structure position. This enables additional correlation steps after the first pass through a generation.

8.3.7 STEP 7: Apply Scripts

Modeling Policies > *Scripting* allows scripts to be run on configuration elements any time the configuration is generated.

Scripts are often used to:

- Issue alerts that allow proactive management when a view has changed or something is actionable. Script alerts can create alarms, send e-mail notifications, play a sound, or issue any other automation event.
- Create a trouble ticket when an element is critical, but does not have an associated ticket.

For example, if a configuration looks at both test and production systems and then overlays a change management system, a script might signal an alert when something is different between test and production systems, but a change order does not exist.

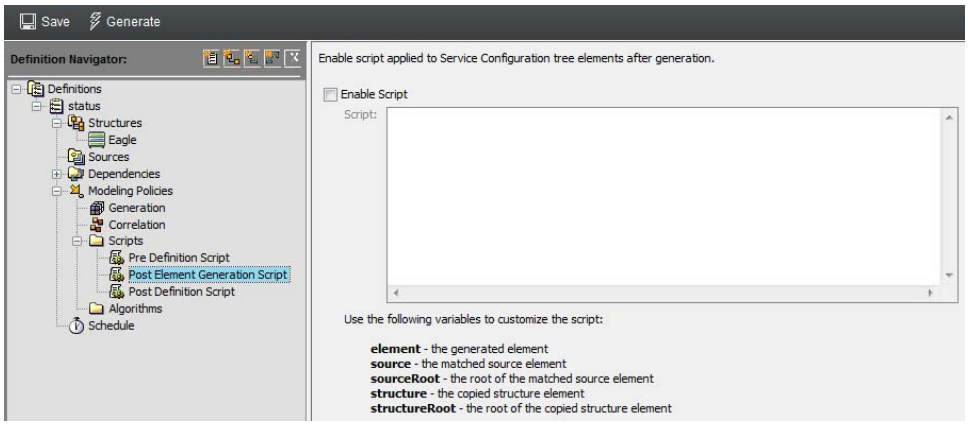
Under *Modeling Policies* > *Scripting*, define scripts to run:

- **Pre Definition:** the script runs before any elements are generated by the definition.
- **Post Element Generation:** the script runs after an element is generated by the definition.
- **Post Definition:** the script runs after all elements are generated by the definition and is the last thing executed by the configuration's job.

IMPORTANT: Never use configuration scripts to invoke another SCM configuration job.

To process newly created elements with a script:

1 In the Definition Navigator pane, click to expand the *Scripts* element under *Modeling Policies*. to define a script for use with new elements within the service configuration.



2 Do one of the following:

- Click *Pre Definition* to define a script to run before any elements are generated by the definition.
- Click *Post Element Generation* to define a script to run after each element is generated by the definition.
- Click *Post Definition* to define a script to run after all elements are generated by the definition and is the last thing executed by the configuration's job.

3 Select the *Enable Script* check box.

4 Enter or paste the script code in the Script text area.

The util class, (scripting engine), supports `$formula.util.encodeURL($objectClass)`, and access to `$formula.Root` and `$formula.Elements`, and so on.

If defining a Pre Definition script, return the string "abort" to cancel definition generation.

For information about scripting, see the *Operations Center 5.5 Scripting Guide*.

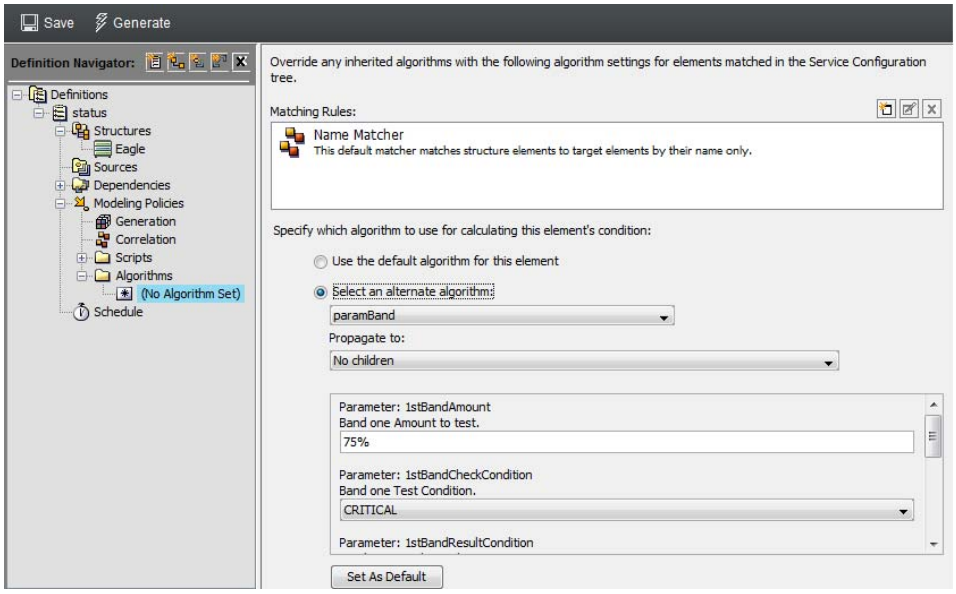
5 Click *Save* to save the definition settings.

8.3.8 STEP 8: Configure Custom Algorithms

By default, a configuration inherits algorithms from Structure definition elements. However, it is possible to use algorithm policy definitions to override inherited algorithms for matched elements. An algorithm definition uses matching rules to select the elements to which an algorithm is applied.

To create an algorithm for elements in the new service tree:

1 In the Definition Navigator pane, click the *Algorithms* element, then click the Create Algorithm link in the right pane.



2 To add, edit, or remove Matching Rules for the algorithm, do any of the following:

- To define a new rule, click (*New Rule*).
- To edit a rule, double-click it.
- To delete a rule, select a rule, then click (*Delete Rule*).

Matching rules are used in multiple areas of Operations Center. For a complete explanation of creating and applying matching rules, see "Assigning Elements by Matching Properties" on page 26.

3 To specify the algorithm to use in calculating matching element conditions, select one of the following options:

- Select the *Use the Default Algorithm For This Element* radio button to apply all default algorithms.
- Select the *Choose an Alternate Algorithm* radio button to customize the algorithm setting for the matched elements.

  If you select an alternate algorithm, click the drop-down list, select an algorithm type, then specify options based on the algorithm type.

For detailed information on algorithms, see *Using Algorithms to Calculate Element State* in the *Operations Center 5.5 Server Configuration Guide*.

4 Click *Save* to save definition settings.s

8.3.9 STEP 9: Test and Generate the Configuration

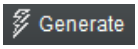
Generating a configuration builds a new hierarchy beneath the service level element where the configuration is defined. As a general practice, generate the new configuration and check the results before allowing users access to it.

Once in production, schedules can be used to automatically update the configuration for users. Regeneration on a timely basis ensures that new information and elements are accessible from the service hierarchy.

TIP: Generate service configurations directly from the Operations Center console. Right-click the element where the configuration is defined, select *Service Configuration*, then select *Generate Now*.

Definition options allow you to enable configuration element locking during generation. When enabled, the service configuration may not be generated manually or by scheduled process as long as a user or another process has the service configuration element locked. For more informaton about enabling or disabling the element locking feature during generation, see Section 8.4, "Enabling Auditing, Debug, and Element Locking During Generation," on page 151.

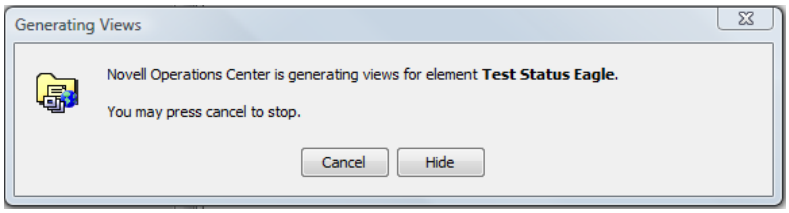
To generate the configuration:

1 On the toolbar, click 

Generate.

The service configuration is generated based on definition settings.

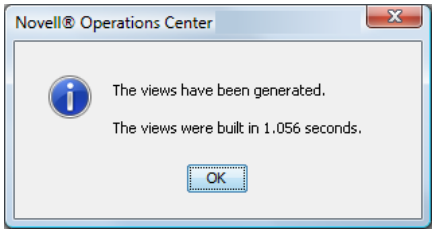
As the configuration generates, the following prompt asks if you want to hide the generation process in the background:



2 Do one of the following:

- Click *Hide* to allow the generation to continue in the background.
- Click *Cancel* to stop the configuration generation in progress.

The following message displays when the generation process is complete. It displays the total time used to complete the generation.



8.3.10 STEP 10: Schedule Updates (Regenerations) of the Configuration

Setup a schedule in the service configuration definition to run automatic updates.

If you have multiple configurations that affect any of the same elements, use a script to run the SCM jobs in a specific sequence. This alleviates the risk of any of these SCM jobs conflicting with each other if they happen to run at the same time. For information, see *Scheduling Multiple SCM Jobs by Using a Script* in the *Operations Center 5.5 Scripting Guide*.

NOTE: Scheduled updates cannot run if the *Lock definition element during generation* is enabled for the definition and another user or process has the configuration element locked. If the scheduled process cannot run, it attempts to regenerate the configuration at the next scheduled time.

To schedule updates (regenerations) of the service configuration:

1. In the Definition Navigator, click the *Schedule* element.
2. Select the *Enable Schedule for This Definition* check box to generate the configuration based on the schedule selections.
3. Click the *Schedule Type* drop-down list, then select the type of schedule to create:

   Cron String: Uses a regular expression to define dates and times.

   Regular Interval: Runs on a regular basis with optional start and end dates.

   Daily Interval: Runs at a regular interval each day or every *x* number of days (such as *Every 3 days*).

   Weekly Interval: Runs once or at regular intervals during a specified range of time (such as *Between 8am and 5pm*) for certain days in a week (such as *Mondays through Fridays*).

   Optionally, set start and end dates or every x number of weeks (such as *Every 2 weeks*).

   Monthly Interval: Runs once or at regular intervals during a specified range of time (such as *Between 8am and 5pm*) for certain days in a week (such as *Mondays thru Fridays*).

   Optionally, set start and end dates or day interval (such as *Every 2 days*) or specific day of the month (such as *the 15th of the month*).
4. Click *Save* to save definition settings.

8.4 Enabling Auditing, Debug, and Element Locking During Generation

Auditing and debug features are available while working in the Service Configuration Editor. After a service configuration definition is defined and established, it could be helpful to enable auditing to be alerted of any changes or activity in service configuration definitions. Likewise, while in development, it can be helpful to have debug features enabled.

TIP: To determine when a configuration was last generated, enable auditing. This can be helpful when a configuration is generated manually, not using a schedule.

Enabling the *Lock definition element during generation* feature works to ensure that no other user or process is editing or removing the configuration element in the Operations Center console while the service configuration is being generated. Likewise, if another user or action has the service configuration element locked (for example, a user is editing the Layout view drawing), the generation process, either manual or scheduled, is unable to run.

To enable auditing or debug features:

1. In the Service Configuration Editor, click the *Definitions* root element.
2. To enable auditing, select the *Enable Auditing* check box.
3. To enable debug logging, select the *Enable Debug Logging* check box.
4. To enable element locking during service configuration generation, select the *Lock definition element during generation* check box.

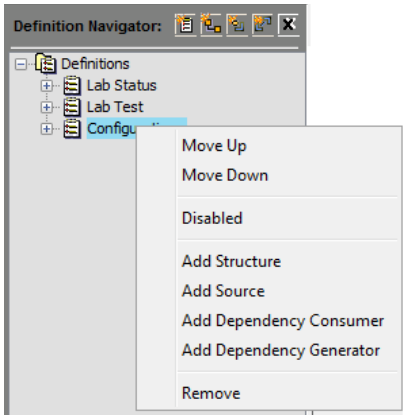
   During generation, `(Locked)` displays next to the configuration element in the Explorer pane in the Operations Center console.
5. Click *Save* to save definition settings.

8.5 Managing Multiple Definitions

When multiple definitions exist, the generations build in the order that they display in the editor.

To use the definition right-click menu options to change the order of definitions, and to disable or remove them:

1. Use the *Move Up* or *Move Down* option options to change the order in which a definition is processed:



2. Select the *Disabled* option to render the definition inactive.
3. To restore the definition, reselect the *Disabled* option.
4. To delete a definition, select the *Remove* option.

8.6 Backing Up, Restoring and Deleting Definitions

SCM definitions can be exported as XML files. This allows you to back up SCM definitions. The *Import* option enables restoring a definition to the same service model or adding the definition to a different service model. Use the *Delete* option to remove obsolete SCM definitions.

- Section 8.6.1, “Exporting a SCM Definition,” on page 153
- Section 8.6.2, “Importing a SCM Definition,” on page 153
- Section 8.6.3, “Deleting a SCM Definition,” on page 153

8.6.1 Exporting a SCM Definition

To export a SCM Definition:

1. Right-click a service model that has an existing definition, then select *Service Configuration* > *Export*.
2. Specify a filename, then export the definition as an XML file.

8.6.2 Importing a SCM Definition

To import a SCM Definition:

1. Right-click a service model that does not have a definition, then select *Service Configuration* > *Import*.
2. Specify a filename, then import the definition as an XML file.

8.6.3 Deleting a SCM Definition

To delete a SCM Definition:

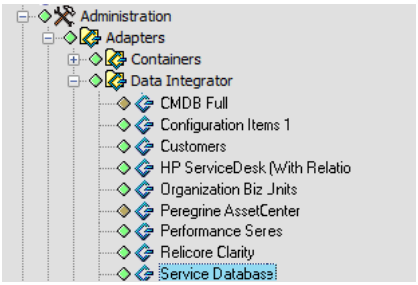
1. Right-click a service model that has an existing definition, then select *Service Configuration* > *Delete*.

9 Use Case: Merging Configuration Management and Asset Databases

Valuable data about an organization and its services is often contained in configuration or asset databases. By integrating this data into Operations Center, you can provide an integrated view of operations and the services they affect.

The basic steps to leverage database information are:

1. Use the Operations Center Data Integrator to create a database definition that mines database information and creates rules that expose data.

   For information about using the Data Integrator, see the *Operations Center 5.5 Data Integrator Guide*.

2. Deploy a new adapter from the newly created Data Integrator adapter definition, then create an adapter using the new adapter type.

   For information about using the Data Integrator, see "Deploying an Adapter Definition" in the *Operations Center 5.5 Data Integrator Guide*.

   Once created, the adapter automatically exposes database builds out all the elements, properties, and alarms as defined. The adapter and its hierarchy are available in the *Elements* hierarchy of the Operations Center console.

3. Correlate views and link to technology and data sources by creating a Service Configuration.

Although asset and management databases can provide unique information to build structures for the configuration, these databases often do not have information on how technology works together. By integrating discovery tools into the service configuration, these relationships can be integrated into a view. For an example of how discovery information is integrated with a configuration, see Chapter 10, "Use Case: Mapping Application Dependencies," on page 163.

Technology sources can be provided from any network or application management adapter integrations, discover tool, or trouble ticketing systems.

The following scenarios illustrate some useful applications of SCM in leveraging information from configuration management and asset databases:

- Section 9.1, "Scenario: Building a View of Services and Supporting Technology," on page 156
- Section 9.2, "Scenario: Building an Organization Structure Based on Discovered Items," on page 159

9.1 Scenario: Building a View of Services and Supporting Technology

In this example, the goal is to build a view that shows services and the technology that supports them. This view displays current condition and state information from the technology components, to enable analysis of how the corresponding services are affected.

Information from an internal, IT-maintained database has already been mined using a Data Integrator adapter. The result is a hierarchy that shows all business services with the servers that support them.

Use SCM to merge this hierarchy with information from three disparate network management systems that manage the servers that these services rely on. Correlation with technology state provides insight to the database information.

Most of the scenarios in this guide begin with a description of a hypothetical system containing a specific element hierarchy originating from a specific set of adapters. The steps used to perform various SCM functions use these hypothetical names. It is assumed that you will apply the examples to your own system, substituting the example element names with actual elements from your hierarchy.

The scenario described in this section uses the following:

- A custom Data Integrator adapter that pulls in database information about IT services and the technology that supports them
- Two Tivoli T/EC Adapters that provide state information about related technology
- An HP OpenView Network Node Manager Adapter that provides state information about related technology

This example updates the view with new database information on a timely basis. In this situation, it is unnecessary to constantly update the configuration. Instead, the schedule runs daily to coincide with a shift change.

To merge elements from these three hierarchies into a single, new hierarchy:

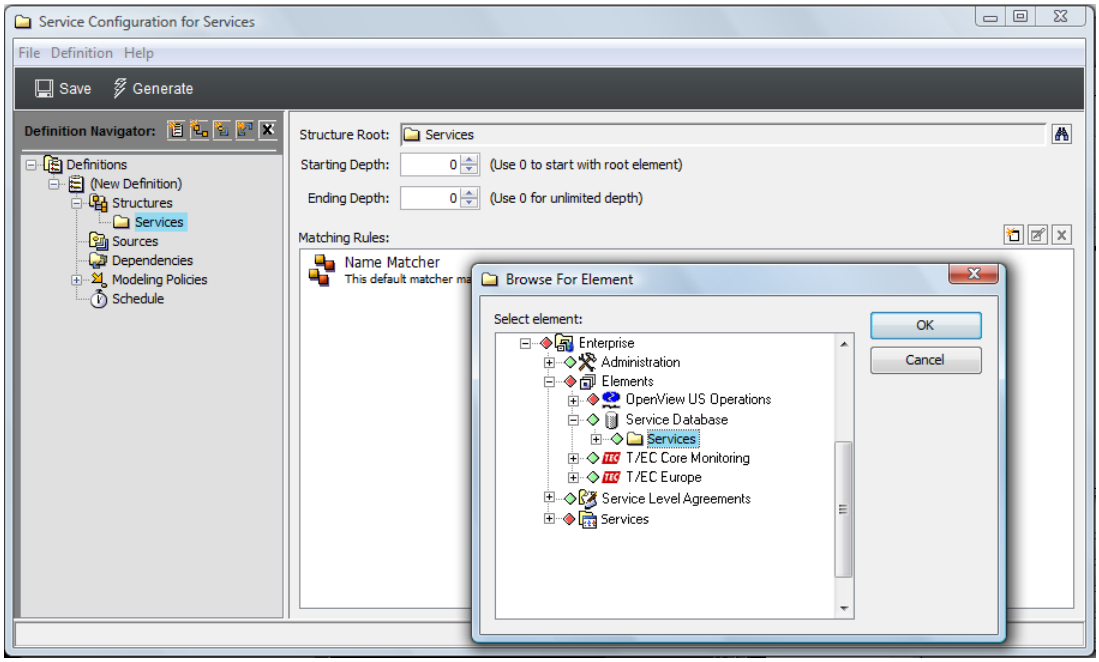
1. In the Explorer pane under *Services*, use the *Add Element* option to create an element that is the parent of the new hierarchy.

   In this example, name it `Service Database`.
2. Right-click the new element, then select *Service Configuration* > *Create* to open the Service Configuration Editor.

3 Define the element structures by identifying an existing hierarchy that drives the shape of the new tree.

In this example, create a Structure definition that adds the *Services* element as the *Structure Root*, as shown in the illustration below.

The *Services* hierarchy is a branch from a Data Integrator adapter that taps into an internal IT-maintained database. It organizes technology (server information) by the service that it supports.



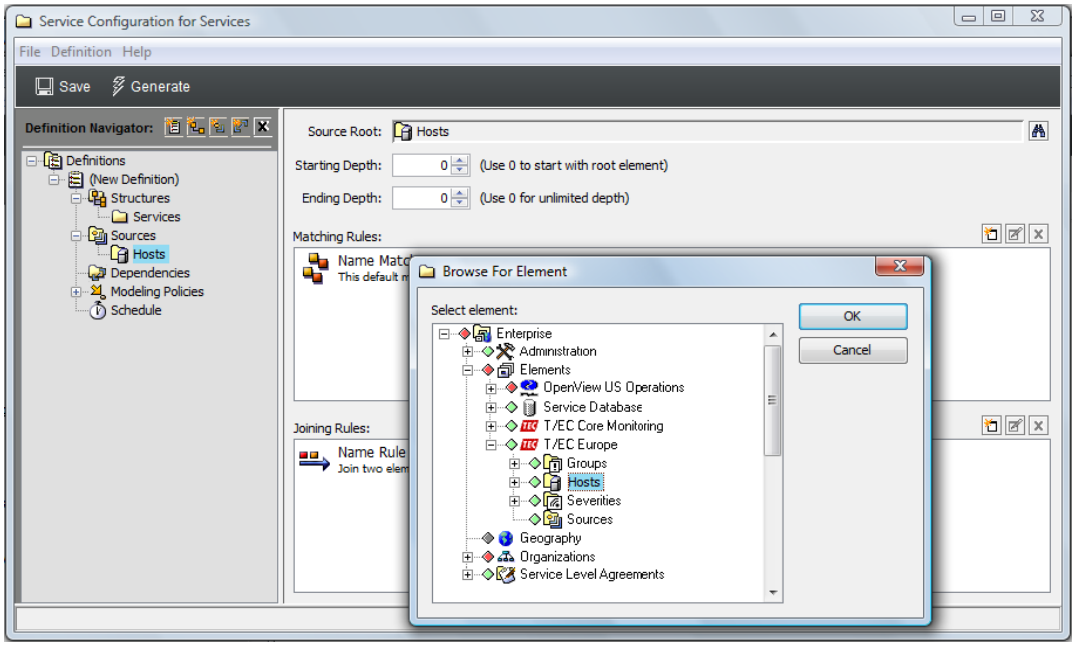
4 To display all elements contained in the *Services* hierarchy, leave *Starting Depth* and *Ending Depth* at `0`.

5 Use the default Matching Rule to apply a simple name match to select elements.

6 Define the sources for the configuration by identifying an existing hierarchy that provides state and information to the new elements in the configuration.

This example creates three Source definitions that link to two different Tivoli T/EC adapters and an HP OpenView adapter that can provide state information about technology elements.

The first two Source definitions link to the *Hosts* folders of the Tivoli T/EC adapters as the *Source Root*. Use the default join rule to link state information to structure elements using a simple name match.

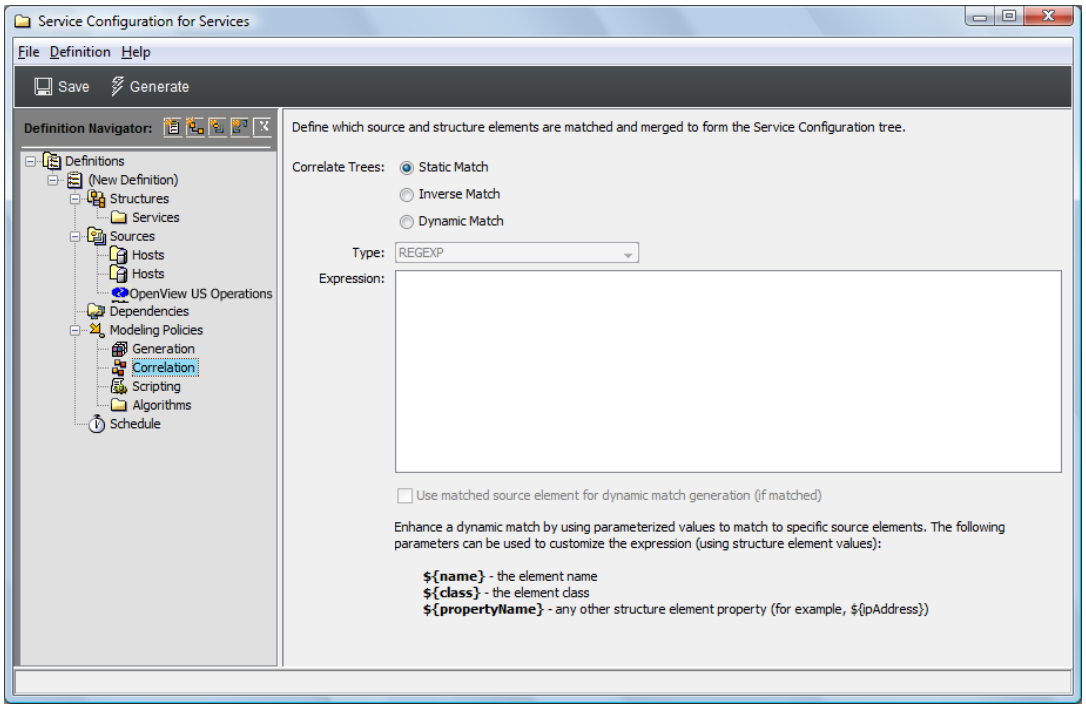


The third Source definition links directly to the root of an HP OpenView adapter by selecting *OpenView US Operations* as the *Source Root*. Use the default join rule on both definitions to link state information to structure elements using a simple name match.

7 To define *Generation* and *Correlation* policy rules to specify how the configuration is generated and elements are correlated, expand *Modeling Policies*, then click *Generation* or *Correlation*.

This example does not add any *Generation* settings. The configuration is generated using the *Structure* definition to define the hierarchy structure and the *Source* definitions setting to integrate source information.

The *Correlation* defaults are retained to correlate configuration objects using a basic *Static Match*. Those objects that are matched are included in the new configuration tree.



8 Click *Save* to save configuration settings, then click *Generate* to generate the new tree.

9 After the configuration is run and results are verified, create a schedule for the generation of the service configuration on a routine basis.

9.2 Scenario: Building an Organization Structure Based on Discovered Items

This example scenario involves building a new Organization structure based on a discovered item from a discovery tool. When working with discovery tool elements, special actions are required.

The following sections describe the two procedures required for this scenario:

- Section 9.2.1, “Defining a Custom Element Menu Option,” on page 159
- Section 9.2.2, “Defining the Service Configuration,” on page 160

9.2.1 Defining a Custom Element Menu Option

The first objective is to create a right-click option that displays a discovered attribute. In this example, the discovered operating system name should be displayed.

It is necessary to define a custom element option to display the discovered operating system. For steps to create custom element options, see Modifying Element and Alarm Menus in the *Operations Center 5.5 Server Configuration Guide*.

Use the following syntax to create the option:

```
Info("OS Name: " + element['CI.Operating System.Name']);
```

This string displays the discovered operating system name. To view attributes that can be displayed, look at Configuration Information on the property page of a discovered item; always start the property name with CI, as shown above.

9.2.2 Defining the Service Configuration

The second goal is to build a SCM Organizational structure based on the operating system with the host name listed beneath, such as the following the structure:

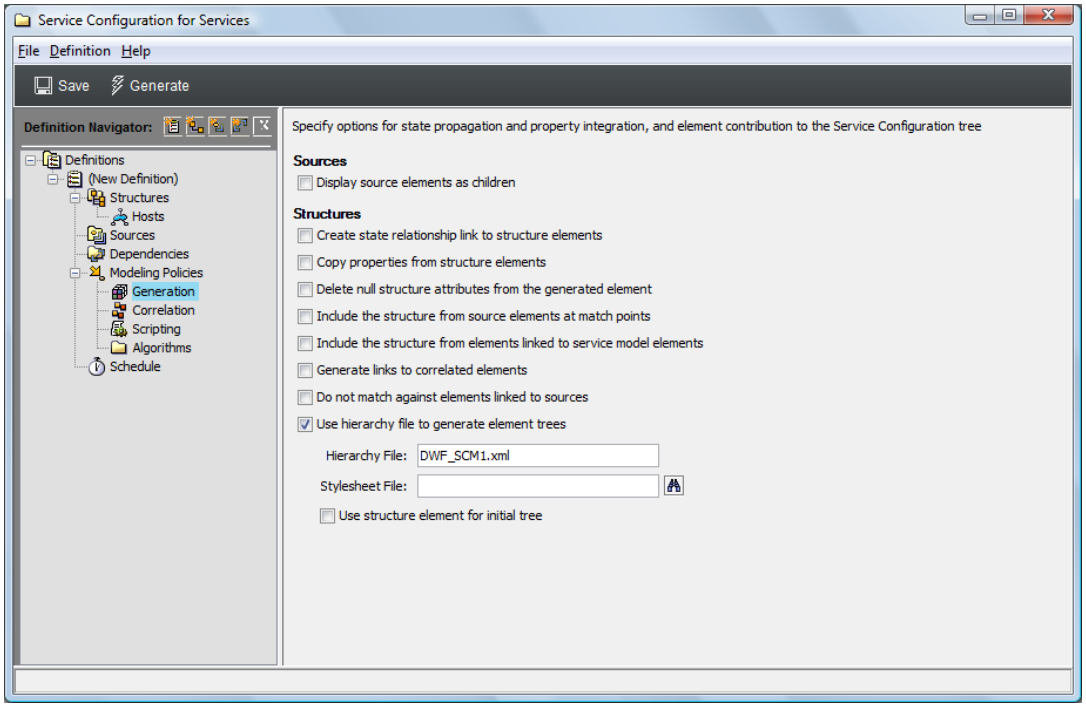
```
Windows 2000
    Labs.net
```

To generate the new organizational structure:

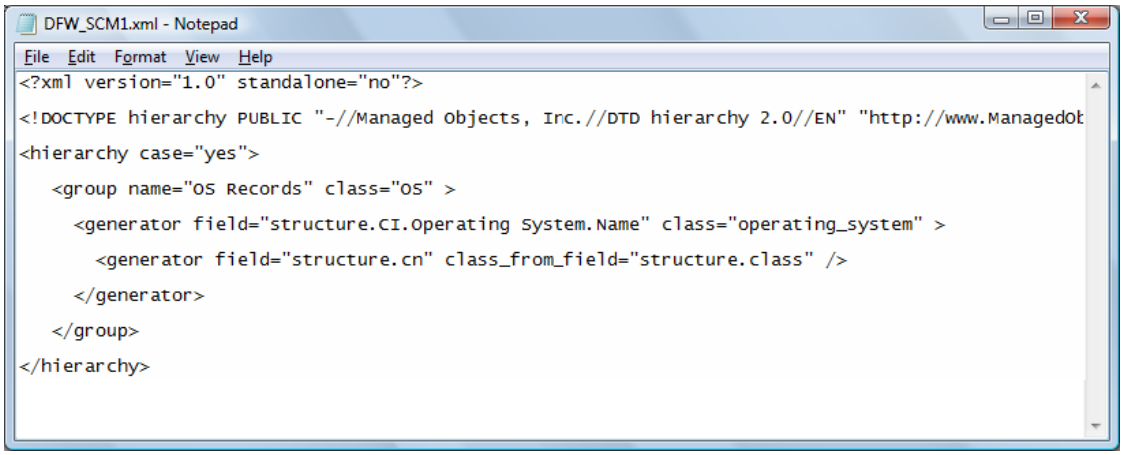
1. Open the Service Configuration Editor and define a Structure definition using the discovered hosts tree from the discovery tool.

   If you are unfamiliar with the Service Configuration Editor, see Chapter 8, "Creating a Service Configuration," on page 125.

2. Define a *Generation* policy similar to the one shown in the following illustration:



The following illustration shows the contents of the `DWF_SCM1.XML` hierarchy file:



3 Click *Generate* to generate the new tree.

Be patient when generating the element tree because the process must query every discovered host for the operating system name.

10 Use Case: Mapping Application Dependencies

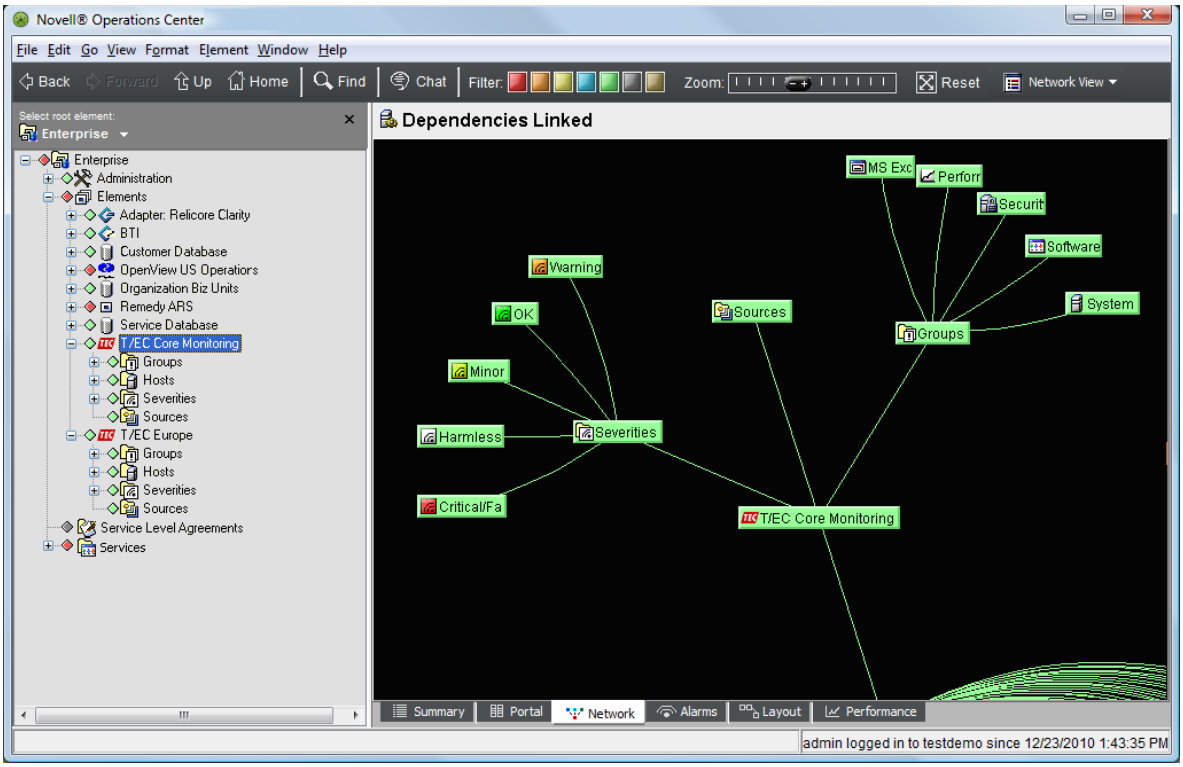
When applications fail, it might be difficult to identify what else is dependent on the application and affected by the failure. For this reason, auto-discovery is a powerful tool that automatically discovers and shows an application's place in a network structure.

- Section 10.1, "Understanding Dependency Mapping," on page 163
- Section 10.2, "Scenario: Displaying Network Dependencies," on page 164
- Section 10.3, "Showing Impacted Services," on page 168

10.1 Understanding Dependency Mapping

When an application fails, all its dependencies might not be known. Standard network tools can provide a picture of the network and its inner workings. SCM's auto-discovery feature, used in the *Network* view, provides additional insight to a network by showing the network health and relationships:

Figure 10-1 *Network View*



10.2 Scenario: Displaying Network Dependencies

This example builds a configuration that displays dependencies in the network.

Assume that a *Service Catalog* hierarchy built using SCM shows all key services, but doesn't provide any state information. A configuration can provide structure for other configurations without adding to the state information that is defined in subsequent configurations.

This scenario uses SCM to leverage a discovery tool adapter that runs discovery and links dependencies. The resulting list of supporting technology from a Tivoli T/EC adapter displays directly under the services. This enables analysis of affected services and discovery of previously unknown relationships.

This scenario uses:

- A *Service Catalog* hierarchy that shows a simple listing of all key services
- A discovery tool adapter to provide discovery on networks and dependency information
- A Tivoli T/EC Adapter that provides state information to newly discovered objects found as a result of discovery by the discovery tool adapter

The following sections are specific to this scenario, but use features of the Service Configuration Editor that were described earlier in Chapter 8, "Creating a Service Configuration," on page 125:

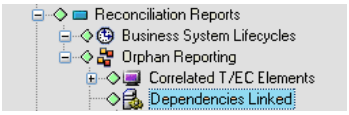
- Section 10.2.1, "Scenario Step 1: Creating a Parent Element in Services," on page 164
- Section 10.2.2, "Scenario Step 2: Identifying an Existing Hierarchy," on page 165
- Section 10.2.3, "Scenario Step 3: Defining the Sources of State Information," on page 165
- Section 10.2.4, "Scenario Step 4: Defining Dependencies," on page 165
- Section 10.2.5, "Scenario Step 5: Configuring Generation Under Modeling Policies," on page 166
- Section 10.2.6, "Scenario Step 6: Generating a New Hierarchy & Schedule Configuration Regeneration," on page 167

10.2.1 Scenario Step 1: Creating a Parent Element in Services

To map dependencies among elements, the first step is creating the parent element of the new hierarchy:

1 In the Explorer pane under *Services*, use the *Add Element* option to create an element that is the parent of the new hierarchy.

In this example, name it `Dependencies Linked`.



10.2.2 Scenario Step 2: Identifying an Existing Hierarchy

In this step, we identify an existing hierarchy that drives the shape of the new tree:

1. Right-click the `Dependencies Linked` element we created in Section 8.3.1, "STEP 1: Create the Service Configuration Definition," on page 129, then select *Service Configuration* > *Create* to open the Service Configuration Editor.
2. Create a Structure definition that adds the *Service Catalog* element as the *Structure Root*.

   The *Service Catalog* hierarchy contains a series of elements that represent key services.

   For more information about creating a structure definition, see Section 8.3.2, "STEP 2: Define Structures," on page 131.
3. To see all elements contained in the *Service Catalog* hierarchy, leave *Starting Depth* and *Ending Depth* at 0.
4. Use the default Matching Rule to apply a simple name match to select elements.

   For more information on matching rules, see "Creating a Matching Rule" on page 65.

10.2.3 Scenario Step 3: Defining the Sources of State Information

In this step, we define the sources of state information for the new configuration, select an existing hierarchy:

1. In the Definition Navigator pane, click  (*New Source*).

   For more information on sources, see Section 8.3.3, "STEP 3: Define Sources," on page 132.
2. Click  (*Browse for Root Element*) to select the *Source Root*.

   In this example, select the *Hosts* branch of the Tivoli T/EC adapter.
3. Use the default join rule to link state information to structure elements using a simple name match.

   This definition makes available state and property information for all new elements found when the configuration searches for dependencies.

   For more information on join rules, see "Using Join Rules" on page 134.

10.2.4 Scenario Step 4: Defining Dependencies

In this step, we define Dependencies for the configuration by specifying dependency rules to apply to a related hierarchy:

1. In this example, create a Dependency Consumer definition, then select the *By Network* branch of a discovery tool adapter as the *Dependency Root*, as shown in the illustration in Step 3 on page 166.

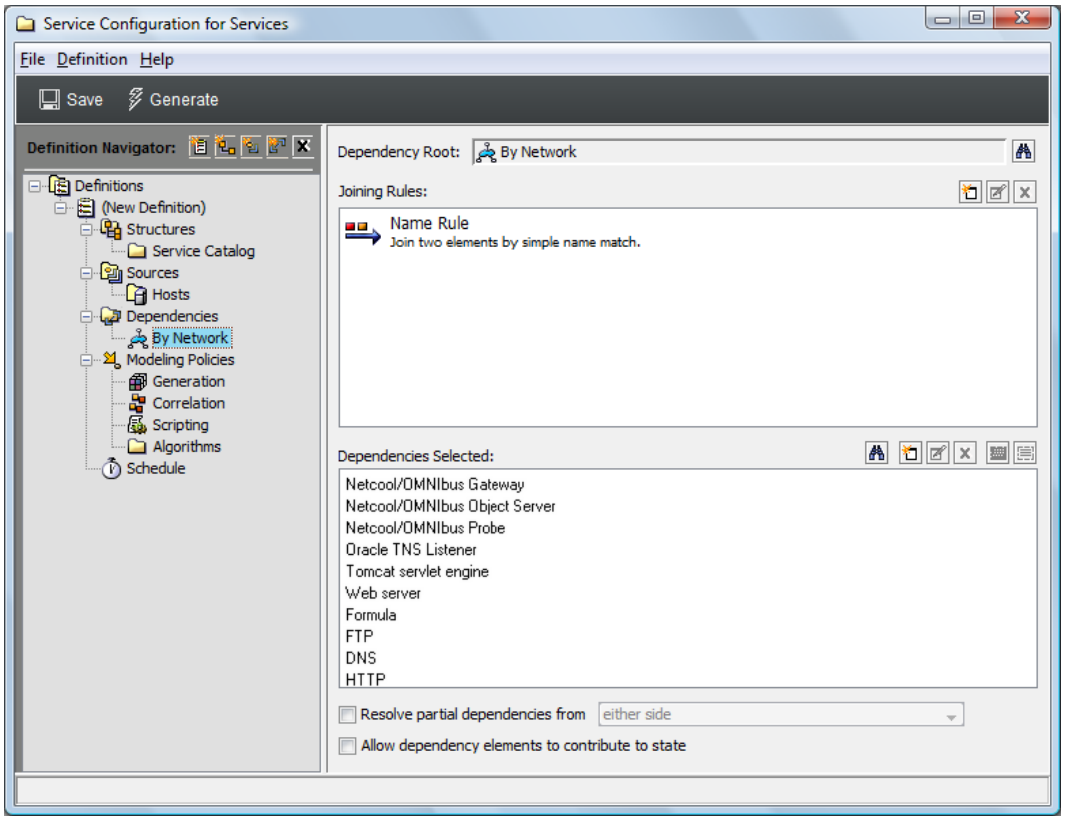
   The *By Network* hierarchy is from a discovery tool adapter that has run discovery on the networks. It contains listings of servers and applications organized by network.

   For more information on Dependency Consumers, see "Using Dependency Consumers" on page 139.

   For more information on defining Dependencies, see Section 8.3.4, "STEP 4: Define Relationship Dependencies," on page 138.
2. Use the default join rule to link discovered elements to structure elements using a simple name match.

3 In the *Dependencies Selected* section, browse the known dependencies for the selected *Dependency Root*, then select one or more dependencies.

The selected dependencies display in the *Dependencies Selected* list as shown in the following illustration:



The selected dependencies and the join rules together determine which relationship information is included in the new configuration.

10.2.5 Scenario Step 5: Configuring Generation Under Modeling Policies

In this step, under *Modeling Policies*, we will define *Generation* and *Correlation* policy rules:

1 Expand *Modeling Policies*, then click *Generation* or *Correlation*.

For more information on *Generation* modeling policies, see Section 8.3.5, "STEP 5: Select Generation Options," on page 143.

For more information on *Correlation* modeling policies, see Section 8.3.6, "STEP 6: Select Element Correlation Options," on page 145.

2 In the *Generation* policy page, select the *Display Source Elements As Children* check box.

This creates a link to source branches where matches occur and displays source elements as children.

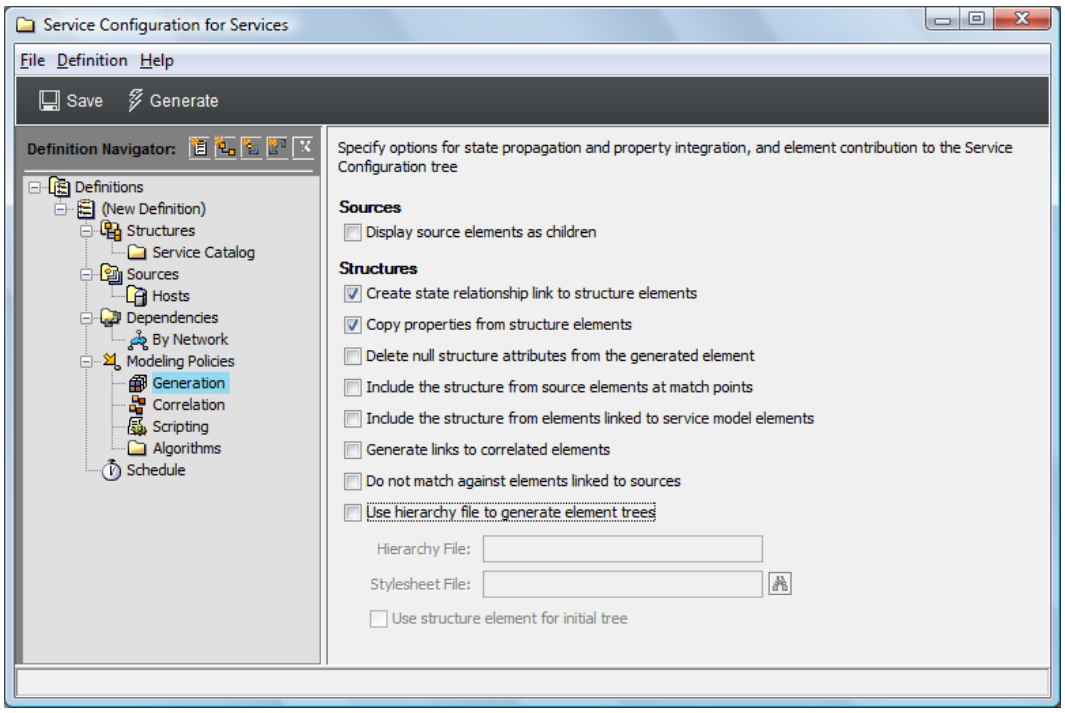
For more information about *Generation*, see Section 8.3.5, "STEP 5: Select Generation Options," on page 143.

3 To include state and property information from the source Tivoli T/EC adapter for structure elements, select the *Create State Relationship to Structure Elements* and *Copy Properties from Structure Elements* check boxes.

Retain *Correlation* defaults to correlate configuration objects using a basic Static Match. Those objects that are matched are included in the new configuration tree.

For more information about *Correlation*, see Section 8.3.6, "STEP 6: Select Element Correlation Options," on page 145.

In this example, defined sources provide state information to the new objects discovered from the search for dependencies:



10.2.6 Scenario Step 6: Generating a New Hierarchy & Schedule Configuration Regeneration

In this step, we generate the new hierarchy and after verification, we create a schedule to keep the hierarchies updated:

1 Click *Save* to save configuration settings.

2 Click *Generate* to generate the new hierarchy.

3 Verify the results of the new configuration are what you expected.

4 Create a schedule to generate the service configuration on a routine basis.

Because this view provides insight into the entire environment, this configuration should be regenerated when changes occur, which is sometimes predictable. For example, set up a schedule to automatically generate this configuration to coincide with the end of each scheduled shift.

In some environments, it could be necessary to schedule this type of configuration to run more often—as frequently as every five minutes.

For more information on creating a schedule, see Section 8.3.9, "STEP 9: Test and Generate the Configuration," on page 150.

10.3 Showing Impacted Services

The Show Impacted function displays the elements that are affected by the performance of a selected element. This function helps identify the business services impacted by the failure of lower-level services or components. The Show Impacted feature and dialog box are described in *Identifying Impacted Services* in the *Operations Center 5.5 User Guide*.

- Section 10.3.1, "Understanding Impacted Services," on page 168
- Section 10.3.2, "Excluding an Element from Impact," on page 169

10.3.1 Understanding Impacted Services

The information displayed in Show Impacted for an element depends on the actual location of the impacted elements, permissions and whether the element is excluded from impact reporting.

- **For elements in the *Service Models* hierarchy:** If an element exists only in the *Service Models* hierarchy, then service models elements are identified as impacted by the element if the following conditions are met:
  - The element has not been excluded from impact reporting (there is an option to exclude elements from impact reporting. For more information, see Section 10.3.2, "Excluding an Element from Impact," on page 169.)
  - The user has View permissions for the element
- **For elements in the *Generational Models* hierarchy:** If an element exists only in the *Generational Models* hierarchy, then *Generational Models* elements are not identified as impacted by the element regardless of the setting for exclude from impact reporting.
- **For elements in both the *Service Models* and the *Generational Models* hierarchies:** If an element appears in both the *Service Models* and *Generational Models* hierarchies:
  - *Generational Models* elements are not identified as impacted regardless of the setting for exclude from impact reporting
  - *Service Models* elements are identified only if the element has not been excluded from impact reporting and the user has View permissions for the element

10.3.2 Excluding an Element from Impact

The option to exclude an element from impact reporting is set on the property page for that element. The default is to include the element in impact reporting (if it does not appear only in the *Generational Model* hierarchy).

To set the exclude from impact reporting option:

1. Right-click an element in the *Services* hierarchy, then select *Properties* to open the Status property page.
2. Select the *Exclude from Impact Reporting* check box.
3. Close the property page.
4. Click *Yes* when asked to save changes.

11 Use Case: Correlating Network Objects

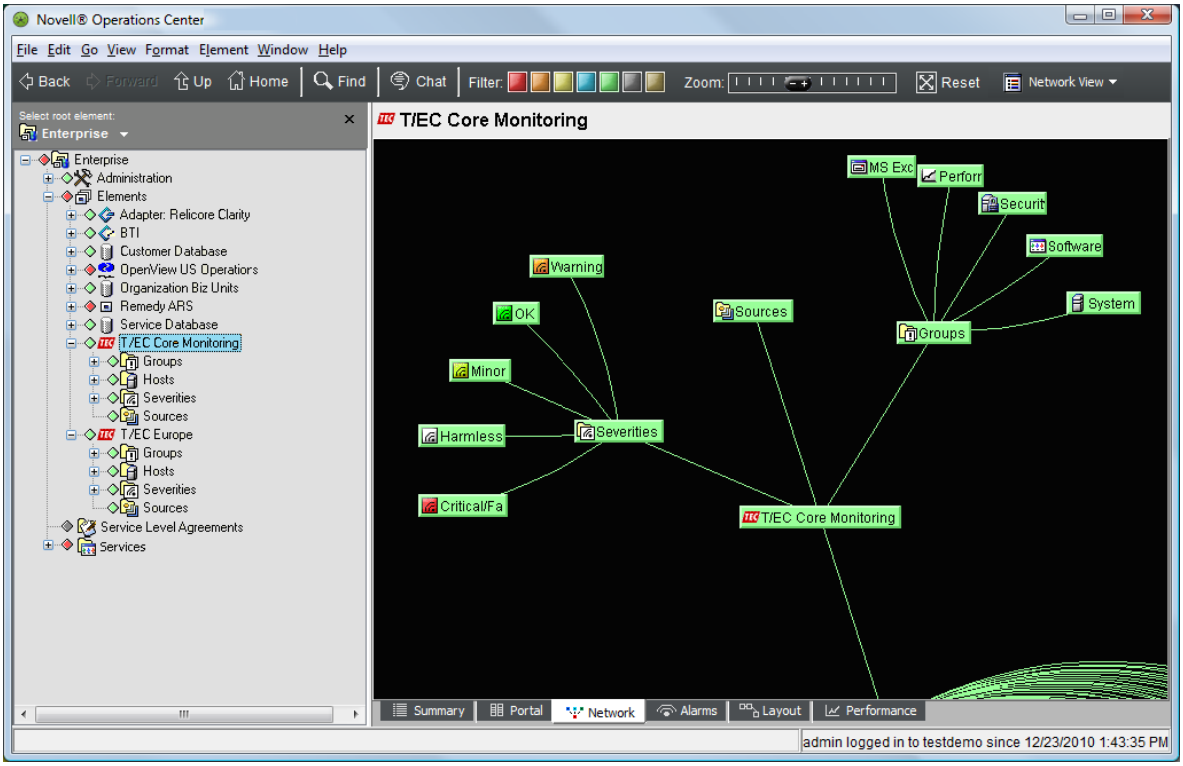
Network management systems can monitor servers in various ways and provide specialized information. Instead of using these separate views of network management systems, consider merging related views so that all information is available in a single view and is correlated for easy interpretation. SCM can correlate network objects to provide a unified view.

Use the SCM to merge similar objects and views into a single view. For example, different applications can provide host data, change management data, trouble tickets, and/or impact data. Correlated views enable you to merge all objects and views into a unified view.

This example scenario combines two *Network* views into one correlated view that can expose and combine information.

Use SCM to combine two T/EC adapter views into one integrated view that represents objects monitored by both adapters as one correlated object in the new configuration. Figure 11-1 shows the resulting correlated *Network* view.

Figure 11-1 *Network View: Two Network views are combined into one correlated view.*



This scenario uses:

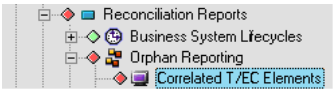
- Two Tivoli T/EC adapters that monitor various servers on the network

  Many servers are found in both views.

Use SCM to correlate two T/EC adapter views into a unified view:

1. In the Explorer pane, add an element that is the parent of the new hierarchy.

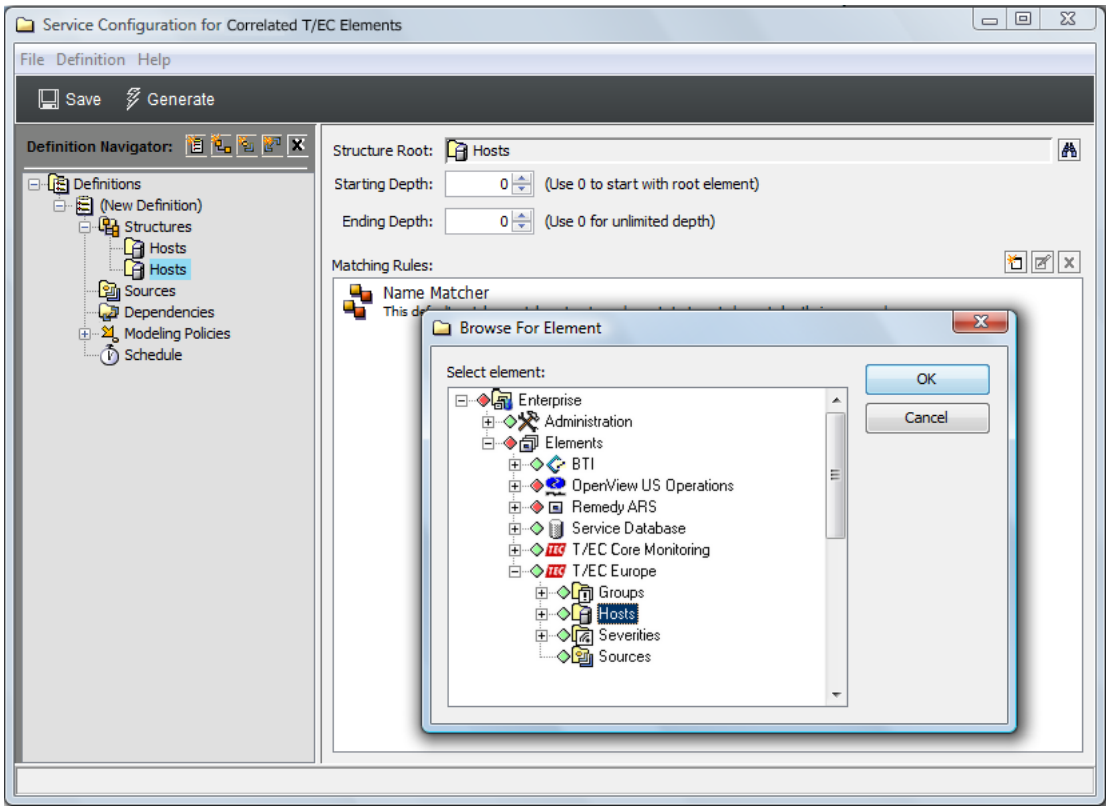
   In this example, the new element is named `Correlated T/EC Elements`.



2. Right-click the element, then select *Service Configuration* > *Create* to open the Service Configuration Editor.
3. Define the Structures for the element by identifying an existing hierarchy that drives the shape of the new tree.

   In this example, define two Structures definitions, each with a *Structure Root* pointing to the *Hosts* element of one of the two T/EC Adapters.

   For more information on defining structures, see Section 8.3.2, "STEP 2: Define Structures," on page 131.
4. To see all elements contained in the *Hosts* hierarchy for both definitions, leave *Starting Depth* and *Ending Depth* at O.
5. Use the default Matching Rule to apply a simple name match to select elements.



For more information on matching rules, see "Creating a Matching Rule" on page 65.

6 Define the Sources by identifying an existing hierarchy that provides state and information to the elements in the new configuration.

This scenario assumes there is no interest in obtaining state or element information from other sources; it is possible to rely solely on the elements and state information from the T/EC adapters. Therefore, do not create any Source definitions.

For more information on defining sources, see Section 8.3.3, "STEP 3: Define Sources," on page 132.

7 To define *Generation* and *Correlation* policy rules to further define how the configuration is generated and elements are correlated:

7a In the left pane, expand *Modeling Policies*, then click *Generation* or *Correlation*.

The right pane updates.

7b Because state is integrated directly from *Structures*, in the *Generation* page, select the *Copy Properties from Structure Elements* and *Create State Relationship to Structure Element*s check boxes.

For more information about *Generation*, see Section 8.3.5, "STEP 5: Select Generation Options," on page 143.

7c Retain the *Correlation* defaults to correlate configuration objects using a basic Static Match.

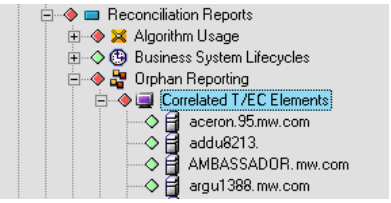
Those objects that are matched are included in the new configuration tree.

For more information about *Correlation*, see Section 8.3.6, "STEP 6: Select Element Correlation Options," on page 145.

8 Click *Save* to save configuration settings.

9 Click *Generate* to generate the new tree.

The new hierarchy generated under *Correlated T/EC Elements* combines the two Tivoli T/EC adapter views into a unified view, as shown:



10 After the configuration is run and results are verified, create a schedule for the generation of the service configuration on a routine basis.

For an up-to-date view of technology, schedule this configuration to generate every 5 minutes.

12 Use Case: Automating Change Management

In today's global business environment, it is critical to have an accurate view of the state of all locations. However, things can change rapidly in remote locations, causing data to be out of data and inaccurate.

Using Service Configuration Manager, gain insight into your environment and the changes that are occurring.

- Section 12.1, "Lifecycle Support: Time-Based Snapshots to Manage Changes," on page 175
- Section 12.2, "Drift Reporting to Highlight Changes," on page 182
- Section 12.3, "Use Case: Creating a Configuration Management Database," on page 186
- Section 12.4, "Reconciliation," on page 194

12.1 Lifecycle Support: Time-Based Snapshots to Manage Changes

Supporting the Change Management lifecycle can be a daunting task. Using SCM, views can be created to show the technology and state of the production environment. If a snapshot can be taken of these views, in order to create a proper baseline, then as the production environment view updates and changes, it can be compared to the baseline snapshot in order to expose any changes.

- Section 12.1.1, "Scenario Description: Production Environment Baseline," on page 175
- Section 12.1.2, "PART 1: Set Up a Production Candidate," on page 177
- Section 12.1.3, "PART 2: Set Up the Production Snapshot," on page 179

12.1.1 Scenario Description: Production Environment Baseline

This scenario uses:

- A *Service Catalog* hierarchy that shows a simple listing of all key services
- A discovery tool that provides discovery on networks and dependency information
- A Tivoli T/EC Adapter that provides state information to newly discovered objects found as a result of discovery by the discovery tool
- Two different service configuration definitions that build the *Production Snapshot* and *Production Candidate* views

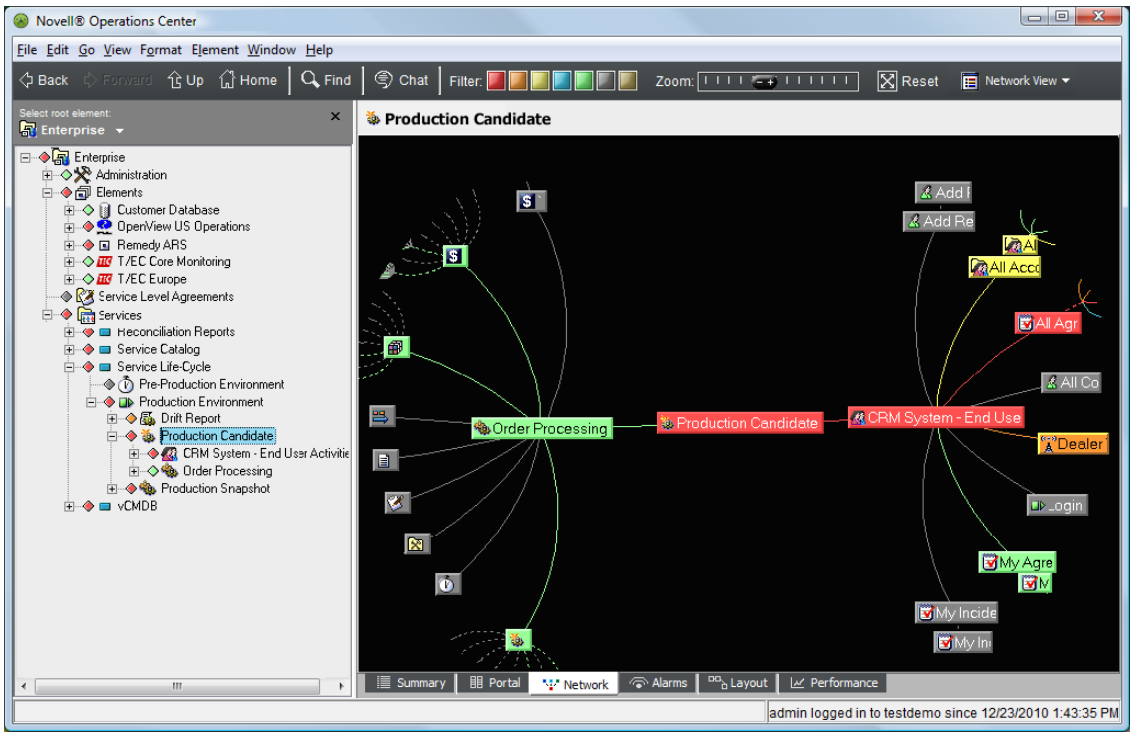
In this example, the goal is to create an accurate view of a production environment, to use as a baseline snapshot of all the networks. As changes occur in the network environment, identify the services affected by these changes by comparing the snapshot baseline with the current *Network* view.

1. Start by taking a snapshot or baseline of the environment.

   To do this, define a service configuration that creates a *Production Candidate* hierarchy, which shows key business services and the technology that supports them. This view remains an active and constantly updated view of the network.

2. Create another configuration named *Production Snapshot*.

3. Take a "picture" of this view, then save it as *Production Snapshot*, which becomes the baseline view:



4. As the *Production Candidate* view is updated to reflect current status and change as they occur, compare it to the *Production Snapshot* baseline to identify changes.

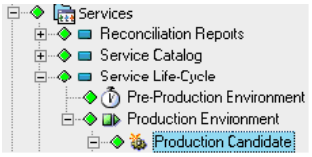
12.1.2 PART 1: Set Up a Production Candidate

This section assumes you are familiar with the Service Configuration Editor. For descriptions of all the steps required to define a service configuration definition, see Section 8.3, "Creating Service Configuration Definitions," on page 128.

To set up a production candidate:

1. In the Explorer pane, create an element that is the parent of the new tree.

   In this example, the new element is named *Production Candidate*:



2. Right-click the element, then select *Service Configuration* > *Create* to open the Service Configuration Editor.
3. To define the Structures for the element by identifying an existing hierarchy that drives the shape of the new tree, create a Structure definition that adds the *Service Catalog* element as the *Structure Root*.

   The *Service Catalog* hierarchy contains a series of elements that represent key services.
4. To see all elements contained in the *Service Catalog* hierarchy, leave *Starting Depth* and *Ending Depth* at 0.
5. Use the default Matching Rule to apply a simple name match to select elements.

   For more information on Matching Rules, see "Creating a Matching Rule" on page 65.
6. Define the Sources for the configuration by identifying an existing hierarchy that provides state and information to the new elements in the configuration.

   In doing this, create a Source definition that uses the *Hosts* branch of the Tivoli T/EC adapter.

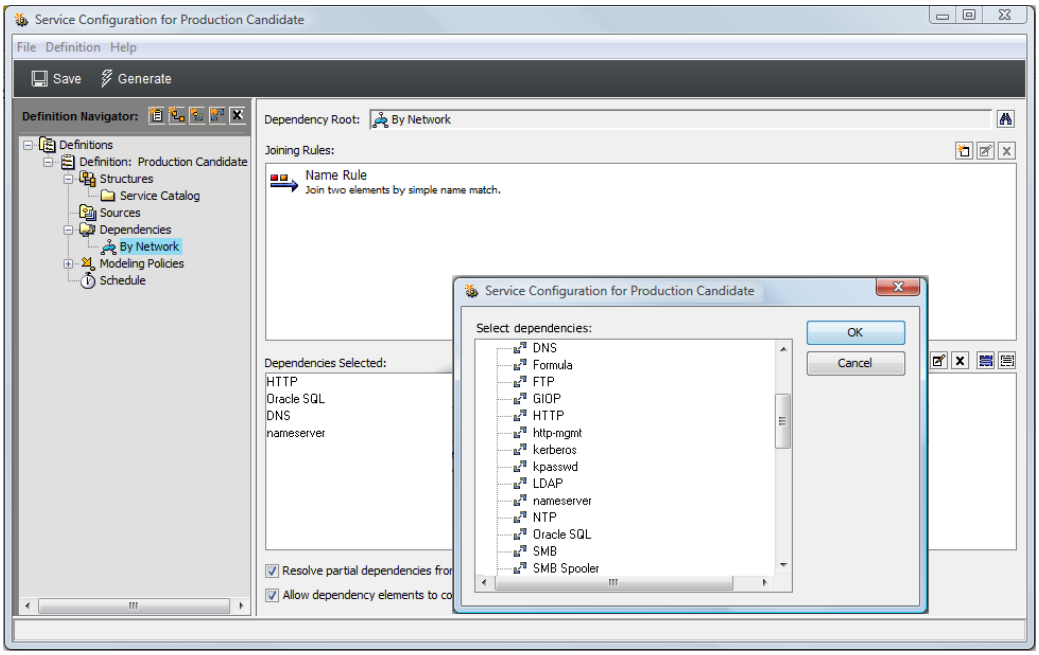
   For more information on defining sources, see Section 8.3.3, "STEP 3: Define Sources," on page 132.
7. Use the default join rule to link state information to structure elements using a simple name match.

   This definition allows state and property information to be available for any new elements found when the configuration searches for dependencies.

   For more information on join rules, see "Using Join Rules" on page 134.

8 To define Dependencies for the configuration by specifying dependency rules to apply to a related hierarchy:

8a Create a Dependency Consumer definition, then select the *By Network* branch of a discovery Adapter to be the *Dependency Root*, as shown in the following illustration:



For more information on creating dependencies, see Section 8.3.4, "STEP 4: Define Relationship Dependencies," on page 138.

8b Use the default join rule to link discovered elements to structure elements using a simple name match.

The *By Network* hierarchy is from a discovery tool adapter that has run discovery on networks. It contains listings of servers and applications organized by network.

9 Browse and select *Dependency Types* as shown in the illustration in Step 8.

The selected dependencies along with the join rules select the information for the new configuration.

10 To define *Generation* and *Correlation* policy rules to further define how the configuration is generated and elements are correlated:

10a In the left pane, expand *Modeling Policies*, then click *Generation* or *Correlation*.

The right pane updates.

For more information on *Generation*, see "STEP 5: Select Generation Options" on page 143.

10b To create a link to source branches where matches occur (displays source elements as children), select the *Display Source Elements As Children* check box.

10c If you do not want to include source information from the Tivoli T/EC adapter for structure elements (in this case, sources provide state to the new objects discovered from the search for dependencies), select the *Copy Properties from Structure Elements* and/or *Create State Relationship to Structure Elements* check boxes.

10d Retain *Correlation* defaults to correlate configuration objects using a basic Static Match.

Those objects that are matched are included in the new configuration tree.

For more information on *Correlation*, see "STEP 6: Select Element Correlation Options" on page 145.

11 Click Save on the toolbar to save configuration settings.

12 Click Generate on the toolbar to generate the new tree.

13 After the configuration is run and results are verified, create a schedule for the generation of the service configuration on a routine basis.

For an up-to-date view of the network, schedule this configuration to update every 5 minutes.

12.1.3 PART 2: Set Up the Production Snapshot

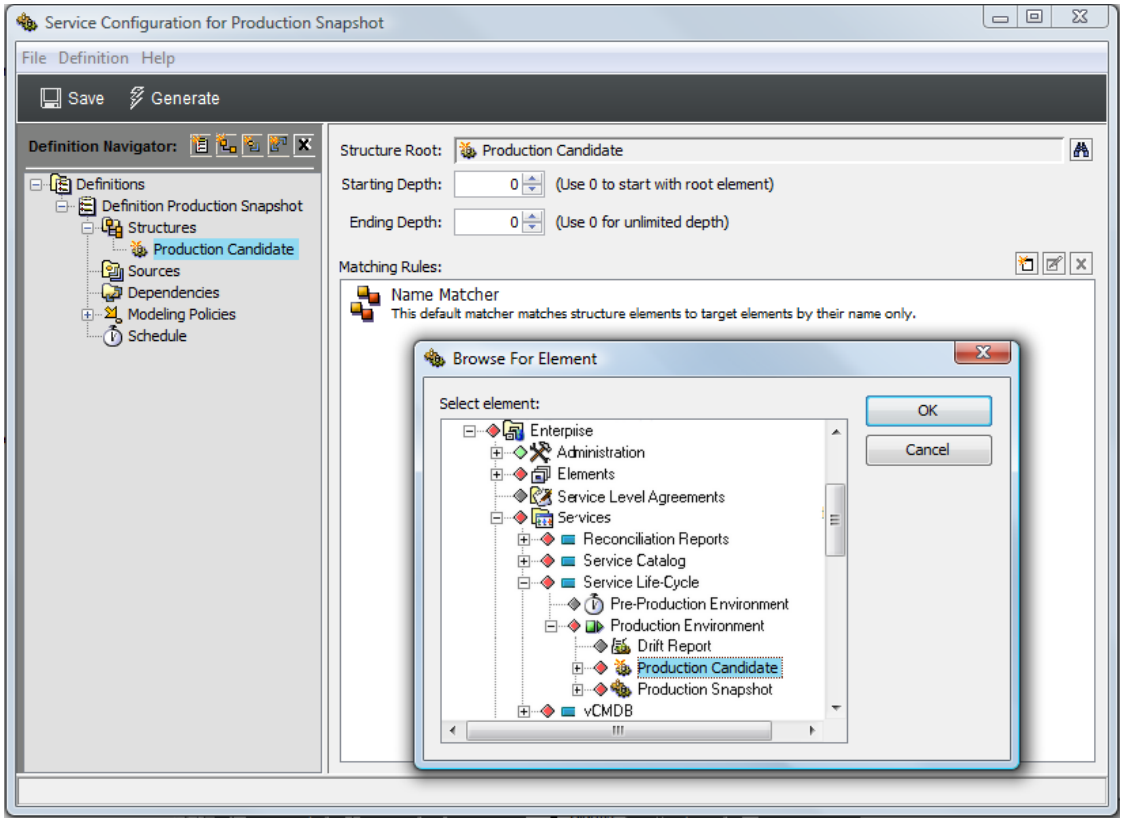
1 In the Explorer pane, create an element that is the parent of the new tree.

In this example, the new element is named *Production Snapshot*:



2 Right-click the element, then select *Service Configuration* > *Create* to open the Service Configuration Editor.

3 To define the Structures for the element by identifying an existing hierarchy that drives the shape of the new tree, create a Structure definition that adds the *Production Candidate* element as a *Structure root*, as shown in the illustration:



The *Production Candidate* hierarchy is the same view created by the configuration that was defined in PART 1: Set Up a Production Candidate of this scenario.

4 Define the Sources for the configuration by identifying an existing hierarchy that provides state and information to the new elements in the configuration.

Because this configuration is meant to be a snapshot of the *Production Candidate* hierarchy, it is desirable to copy the configuration as it exists. It can be assumed there is greater interest in the current state information as provided by the Structure. Instead of creating any Source definitions, use *Generation* policy options to capture properties and state calculations from the *Structure* elements.

5 To define *Generation* and *Correlation* policy rules to further define how the configuration is generated and elements are correlated:

5a In the left pane, expand *Modeling Policies*, then click *Generation* or *Correlation*.

The right pane updates.

5b To obtain the state directly from the structure, select the *Create State Relationship to Structure Elements* check box.

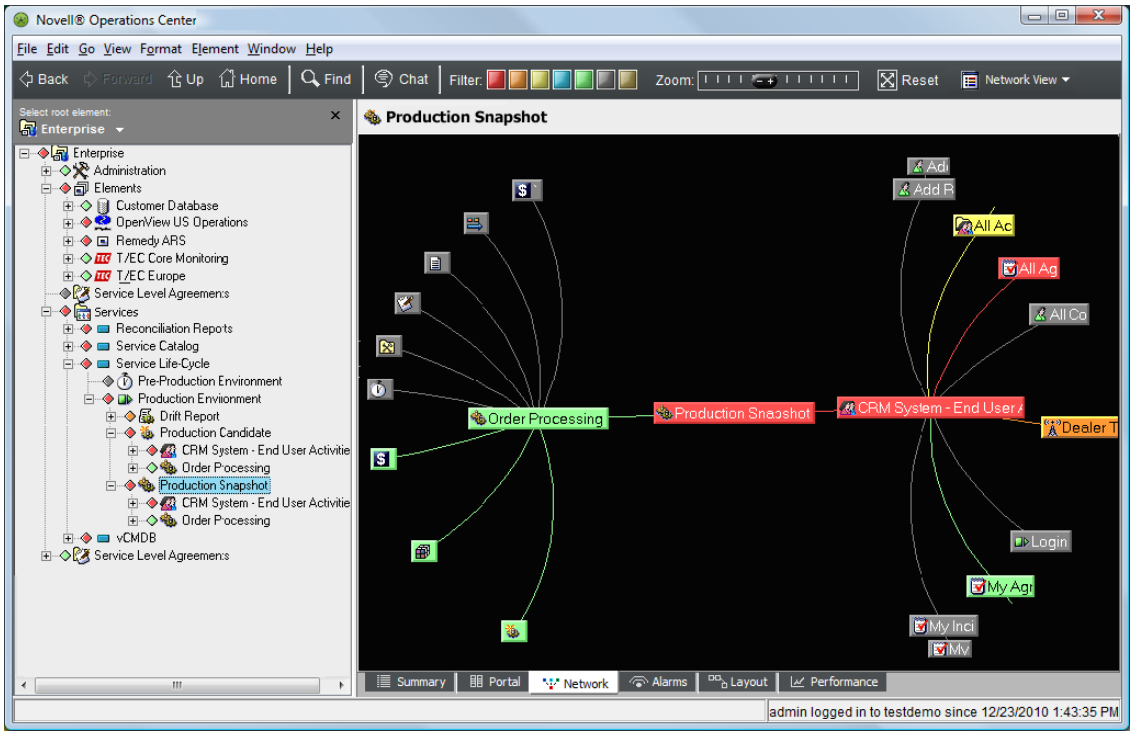
5c To have the hierarchy correlate directly as defined, accept the default *Static Match* setting in *Correlation* policies.

Those elements that are matched are included in the new configuration tree.

6 Click Save on the toolbar to save configuration settings.

7 Click *Generate* to generate the new tree.

The following illustration shows the *Production Snapshot Network* view:



8 After the configuration is run and results are verified, create a schedule for the generation of the service configuration on a routine basis.

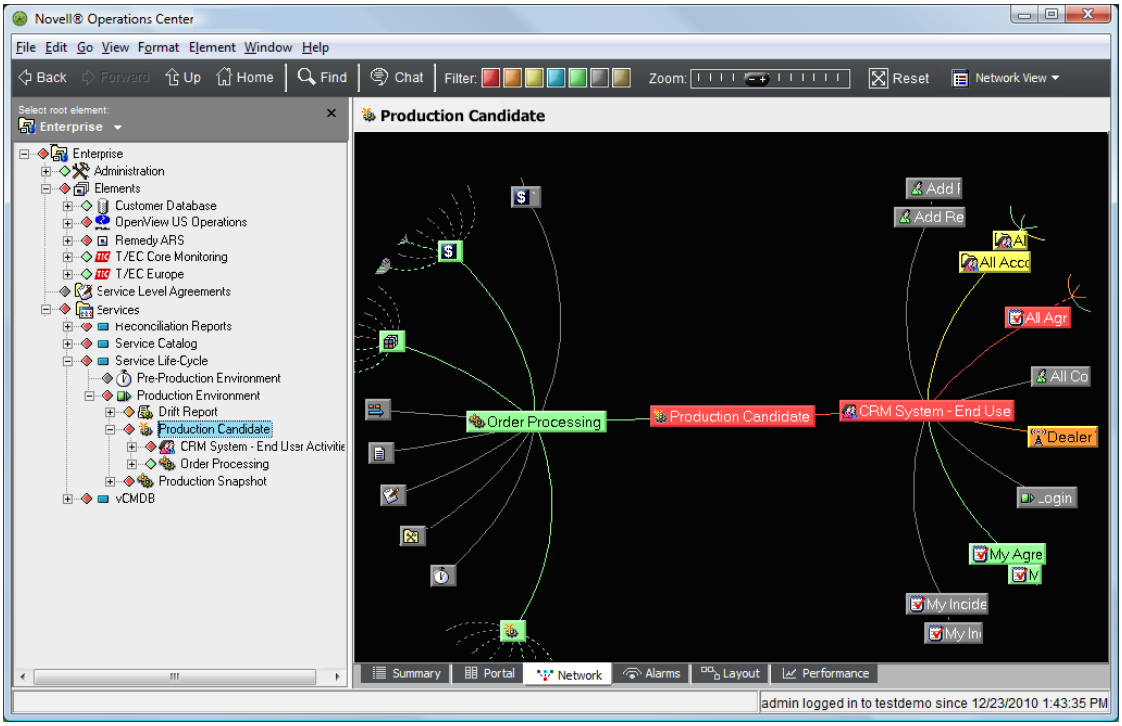
Because this configuration is the baseline used to compare against the *Production Candidate* configuration, it could be desirable to update the configuration manually when deemed necessary. A schedule is not created to update the configuration.

12.2 Drift Reporting to Highlight Changes

SCM can highlight changes between hierarchies for production and proposed infrastructure. This can produce a type of drift reporting view that allows isolating the elements that have changed.

Figure 12-1 illustrates changes between hierarchies for production and proposed infrastructure are highlighted (colored) in this view:

Figure 12-1 *Network View*



- Section 12.2.1, “Scenario Description: Shows Only Changed Elements,” on page 182
- Section 12.2.2, “Scenario Steps,” on page 183

12.2.1 Scenario Description: Shows Only Changed Elements

This scenario creates a view that shows only the elements that have changed between production and proposed views.

Use SCM to analyze the two hierarchies, production (*Production Snapshot*) and proposed (*Production Candidate*), and use the configuration to run an inverse match to highlight the delta (change).

This scenario uses:

- A *Production Candidate* hierarchy that shows key business services and the technology that supports them

  It is created by a service configuration that is scheduled to update on a regular basis.

- A *Production Snapshot* hierarchy that shows a baseline state for key business services and the technology that supports them

  It is a baseline snapshot created by a service configuration that takes a "copy" of the *Production Candidate* hierarchy at a certain point in time. It is not scheduled to run on a regular basis, but is generated manually when there is a need to do so.

12.2.2 Scenario Steps

To use SCM to review the changes between a production and proposed hierarchy:

1 In the Explorer pane, create an element that is the parent of the new tree.

In this example, the new element is named *Drift Report*:

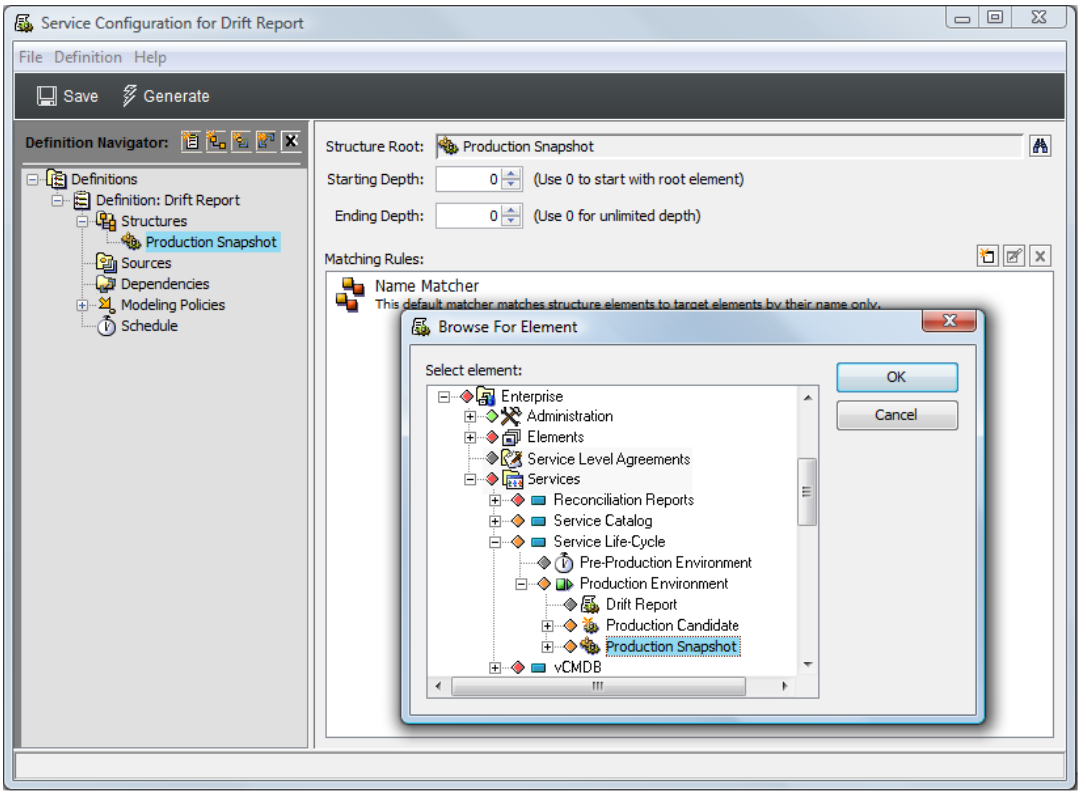


2 Right-click the element, then select *Service Configuration* > *Create* to open the Service Configuration Editor.

3 Define the Structures for the element by identifying an existing hierarchy that drives the shape of the new tree:

3a Because this scenario requires multiple Service Configuration definitions, we named the default definition `Drift Report`.

3b Create a Structure definition that adds the *Production Snapshot* element as the *Structure Root*, as shown in the illustration below.



Production Snapshot is a hierarchy that shows a baseline state for key business services and the technology that supports them. For more information how this hierarchy was created, see Step 2 in Section 12.1, "Lifecycle Support: Time-Based Snapshots to Manage Changes," on page 175.

3c To see all elements contained in the *Production Snapshot* hierarchy, leave *Starting Depth* and *Ending Depth* at 0.

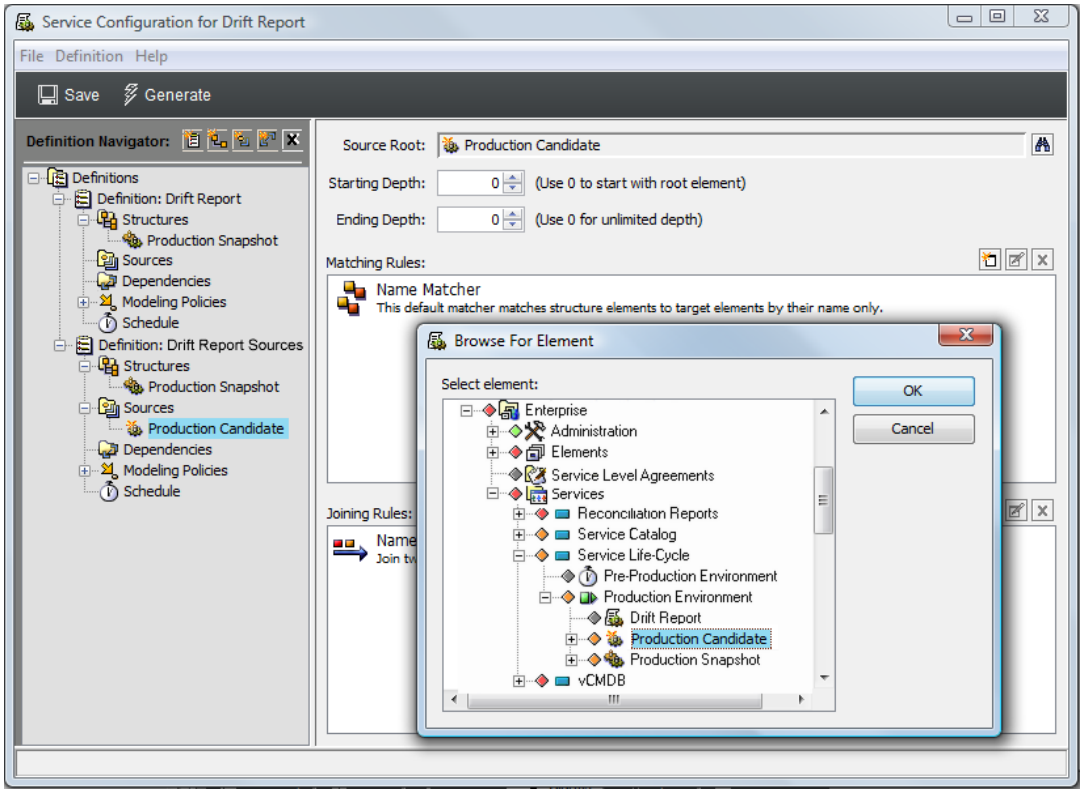
3d Use the default Matching Rule to apply a simple name match to select elements.

4 Define the Sources for the configuration by identifying an existing hierarchy to compare with the Structure. Normally sources are defined to provide state calculations but in this scenario, we are using it as another structure for comparison to the actual structure.

NOTE: Because this scenario uses the *Require Match Between Structure and Source Elements* option in the Source join rule AND copies properties from the Structure elements, we need to create a new definition to define Sources.

4a Create a new Service Configuration definition and name it `Drift Report Sources`. Make sure it is listed below the `Drift Report` definition.

4b Create a Source definition that adds the *Production Candidate* element as the *Source Root*.

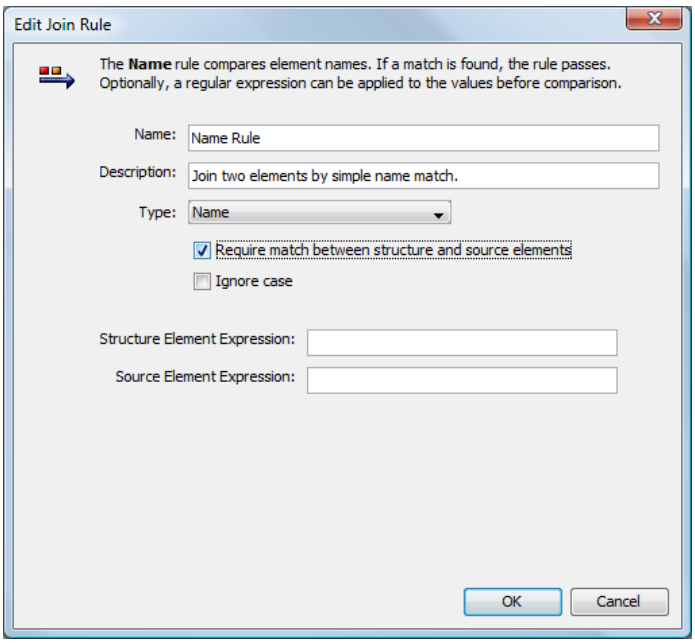


For more information how this hierarchy was created, see Step 1 in Section 12.1, "Lifecycle Support: Time-Based Snapshots to Manage Changes," on page 175.

Note that because of the selected *Generation* policies, the *Production Candidate* does not actually provide state to the configuration. In this example, it is used as another structure for comparison to the actual structure (*Production Snapshot*):

4c Modify the default join rule to match source elements to structure elements using a simple name match by selecting the *Require Match Between Structure and Source Elements* check box.

A specific and accurate match is important to this configuration in order to determine the specific changes between the two trees:



4d To define *Generation* and *Correlation* policy rules to further define how the configuration is generated and elements are correlated:

4d1 In the left pane, expand *Modeling Policies*, then click *Generation* or *Correlation*.

The right pane updates.

4d2 To see all new elements after the baseline was run, select the *Display Source Elements As Children* check box in *Generation* policies.

All new elements in the *Production Candidate* tree is selected (over what is found in the *Production Snapshot*).

4d3 To compare against the *Production Snapshot* tree, select the *Copy Properties from Structure Elements* check box in *Generation* policies.

It is assumed that state calculations, normally provided by the *Source* tree (*Production Candidate*), can be ignored.

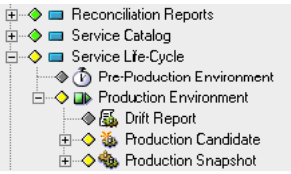
4d4 Because the focus is on the items that are different (have changed) between the *Production Snapshot* and the *Production Candidate* trees, select the *Inverse Match* radio button in *Correlation* policies.

Only the elements that are not matched (not the same) are shown in the new configuration tree.

5 Click Save on the toolbar to save configuration settings.

6 Click *Generate* to generate the new tree.

The new hierarchy tree is built under the *Drift Report* element. The generated configuration now found under the *Drift Report* element shows only the items that are different between the *Production Snapshot* and *Production Candidate* trees.



If the resulting configuration shows no elements (as shown in the previous illustration), then no drift (or change) in the production tree occurred after the last snapshot was taken.

7 After the configuration is run and results are verified, create a schedule to generate the service configuration on a routine basis.

Assume that a decision is made not to schedule the automatic generation of the Drift Report. This configuration is run manually when you want to promote change into the baseline (the *Production Snapshot* view).

12.3 Use Case: Creating a Configuration Management Database

How do you handle configuration management when there are multiple data sources?

SCM is ideal for creating a Service Catalog that consolidates and organizes data. It is ideal because it doesn't require physically building a new CMDB360° to store all the newly organized and filtered data.

Use SCM to create a CMDB360° to unify data and knowledge from various views and databases. Only information about new relationships that are inferred is persisted in Operations Center.

- Section 12.3.1, "Scenario Description: Creating a View from Multiple Data Sources," on page 186
- Section 12.3.2, "LAYER 1: Set Up an Initial Structure to be Leveraged for the CMDB," on page 187
- Section 12.3.3, "LAYER 2 – Define a Second Definition That Provides State Information and Generate the CMDB360° View," on page 189

12.3.1 Scenario Description: Creating a View from Multiple Data Sources

In this example, the goal is to create a view that combines data from multiple data sources.

Use SCM to create multiple configuration definitions that are merged together to accomplish the goal. The first configuration leverages a CRM database that enables creating a hierarchy that shows the services that support customers. Leverage this configuration to correlate with technology by normalizing, reconstituting and correlating the various technology elements. The end result is the CMDB360°.

Because you are working with an existing element hierarchy unique to your environment, you should read through this example, then apply the steps to the relevant elements in your system.

This scenario uses:

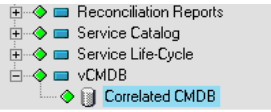
- A custom Data Integrator adapter that mines a CRM database and provides information about what services support various customers

  In this example, *Customers* is the element hierarchy created by the Business Data Integrator adapter that queries a CRM database and organizes data by service name.
- A second custom Data Integrator adapter that pulls in database information about IT services and the technology that supports them
- A correlated *CMDB* hierarchy created by a service configuration (Layer 1 of this scenario) and reused for the subsequent configuration (Layer 2 of this scenario)

12.3.2 LAYER 1: Set Up an Initial Structure to be Leveraged for the CMDB

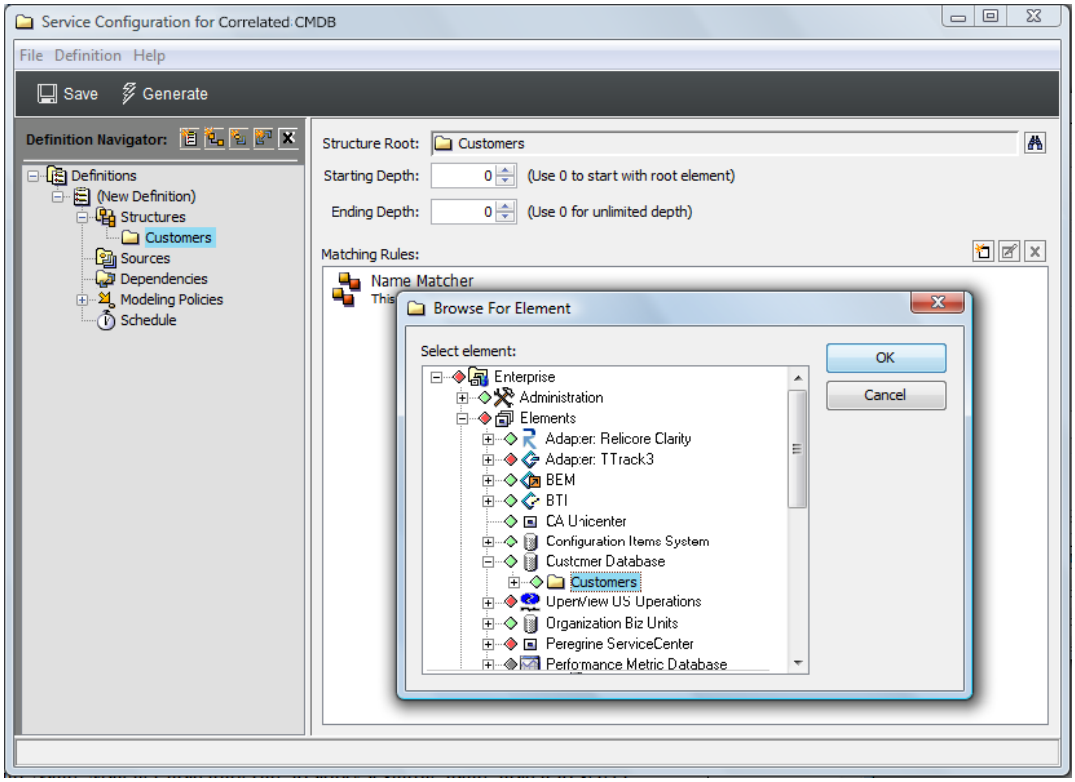
1. Under *Service Models*, add an element named *vCMDB*.

   This is the parent of the new tree.
2. Add an element named *Correlated CMDB* as a child of *vCMDB*:



3. Right-click the *Correlated CMDB* element, then select *Service Configuration* > *Create*.
4. When prompted to add a new definition, click *Yes* to open the Service Configuration Editor.
5. Specify the definition name *Customers*.
6. Click the *Create Structure* link in the right pane.

7 To define the Structures for the element by identifying an existing hierarchy that drives the shape of the new tree, click the *Browse* icon, then select the *Customers* element as a *Structure Root*, as shown in the following illustration:



Assume that *Customers* is a hierarchy that is created by a Data Integrator adapter that queries a CRM database and organizes data by service name.

8 To see all elements contained in the *Customers* hierarchy, leave *Starting Depth* and *Ending Depth* at 0.

9 Use the default Name Matcher matching rule to apply a simple name match to select elements.

10 To create a *Customers* hierarchy that can be leveraged with future layers or generations of configurations, without having state information that might confuse the view, do not create any Source definitions.

11 To define *Generation* and *Correlation* policy rules to further define how the configuration is generated and elements are correlated, in the left pane, expand *Modeling Policies*, then click *Generation* or *Correlation*.

The right pane updates.

In this example, the configuration is generated using the *Generation* and *Correlation* defaults.

Structure is determined solely by the *Customers* hierarchy as leveraged by the Structures definition defined earlier.

12 Click Save on the toolbar to save configuration settings.

13 Click Generate to generate the new tree.

The plan is to use this configuration with another configuration defined for the same element. Generating this configuration makes the hierarchy available for reuse.

12.3.3 LAYER 2 – Define a Second Definition That Provides State Information and Generate the CMDB360° View

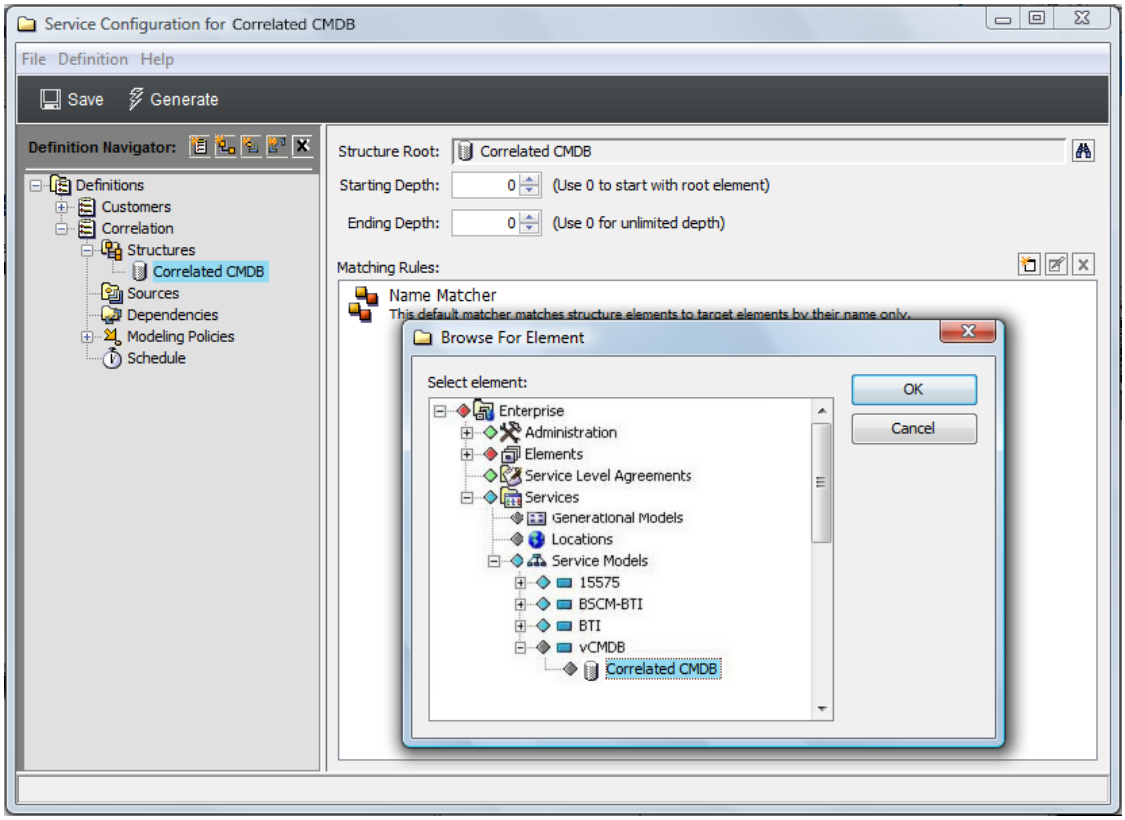
1 In the Service Configuration Editor, click (*New Definition*) to create another configuration definition.

The new definition element is added beneath the existing definition (*Customers*) in the *Definition Navigator* tree.

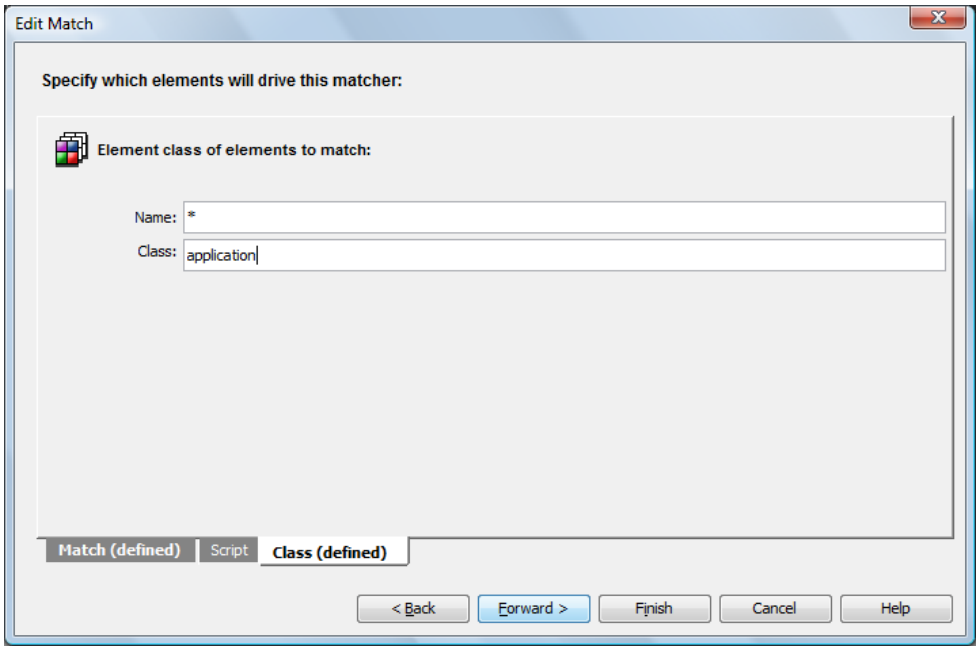
2 Enter the new definition name: *Correlation*.

3 Click the Create Structure link to select an existing hierarchy to drive the shape of the new tree.

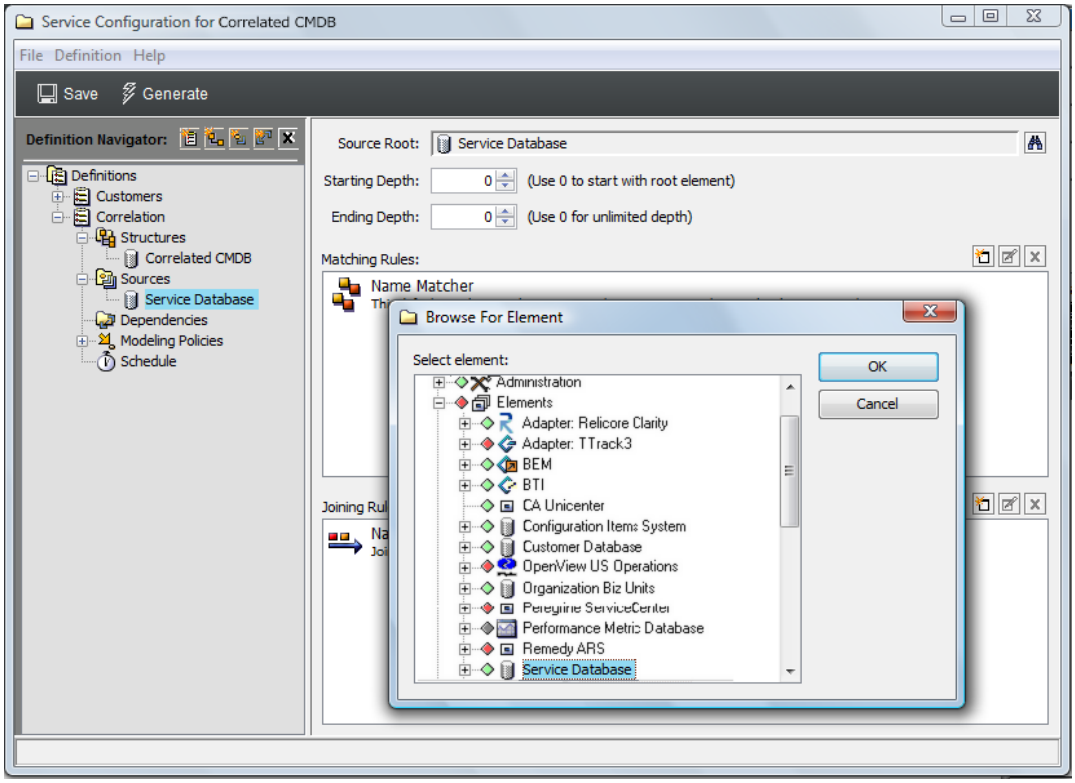
4 Click *Browse*, then select *Services* > *Service Models* > *vCMDB* > *Correlated CMDB*:



5 To match only those elements that are in the application class, define a Matching Rule to match by class as shown in the following illustration:



6 To define the Sources for the configuration by identifying an existing hierarchy that provides state and property information to the new elements in the configuration, create a Source definition that uses *Service Database*, which is an element originating from the Data Integrator adapter:

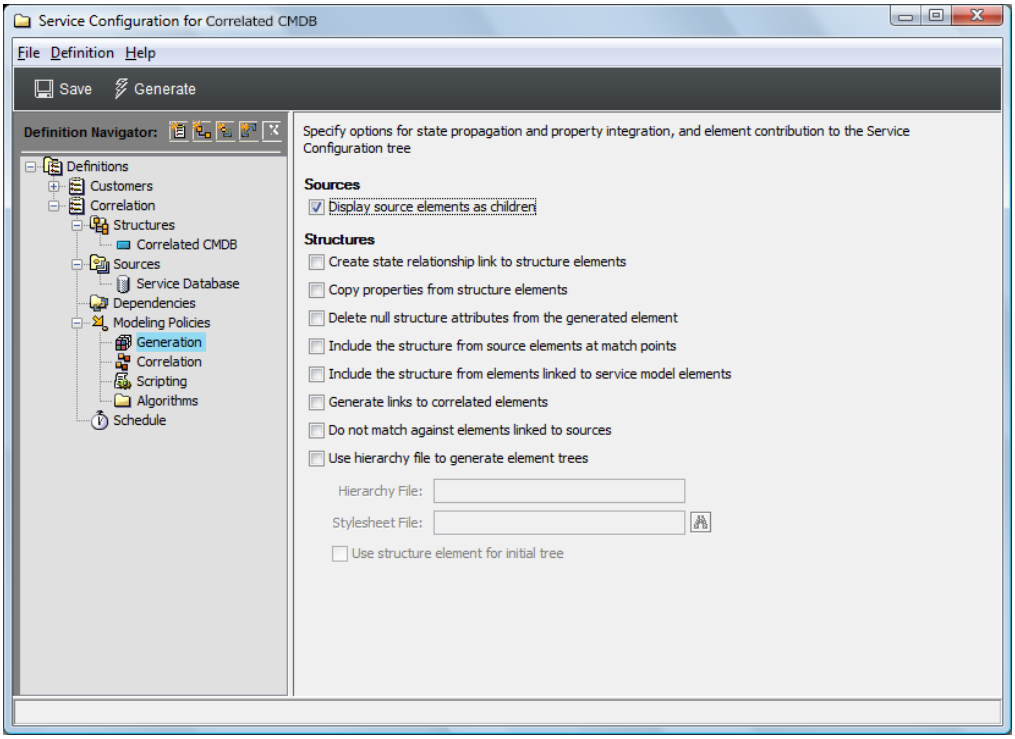


7 Use the default join rule to link state information to structure elements using a simple name match.

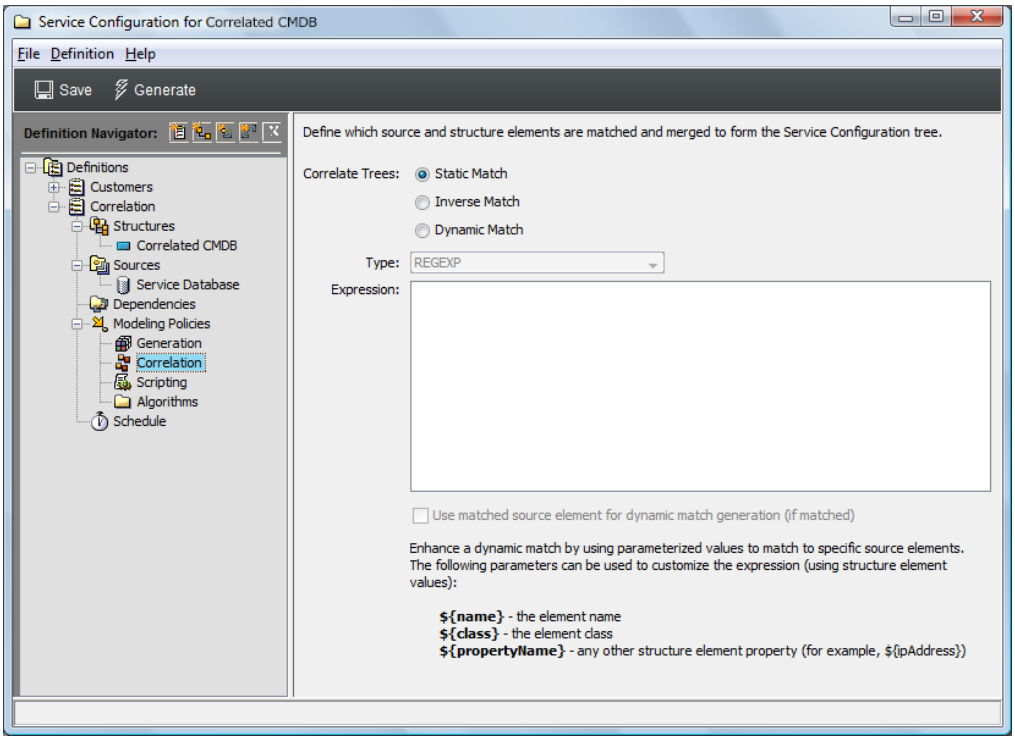
This definition links state and property information found in the *Service Database* to elements in the *Correlated CMDB* structure.

8 To define the *Generation* and *Correlation* policy rules to specify how the configuration is generated and elements are correlated:

8a To display all children found beneath match points in the *Service Database* hierarchy, select the *Display Source Elements As Children* check box in *Generation* policies:



8b To generate the hierarchy and match elements as specified in the *Structures*, *Sources*, and *Generation* definitions, retain the default *Static Match* selection in *Correlation* policies:

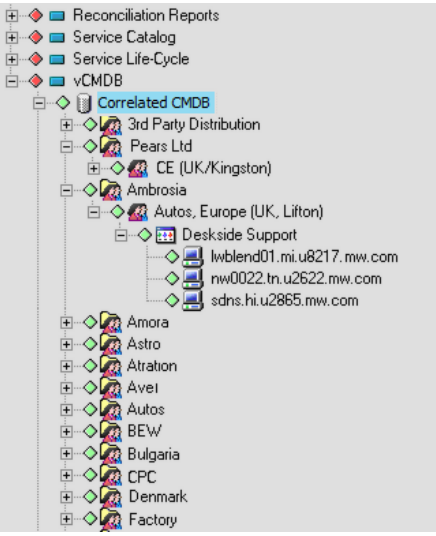


9 Click *Save* on the toolbar to save configuration settings.

10 Click *Generate* to generate the new tree.

The new hierarchy is built under the *Correlated CMDB* element. It is the result of two service configuration definitions: *Customers* and *Correlation*. *Customers* uses as its structure the *Customers* hierarchy created by a Data Integrator adapter that queries a CRM database and organizes data

by service name. *Correlation* uses as its source the *Service Database* element hierarchy, which also originates from a different Database Integrator adapter. *Service Database* provides state and property information to the elements in the new configuration.



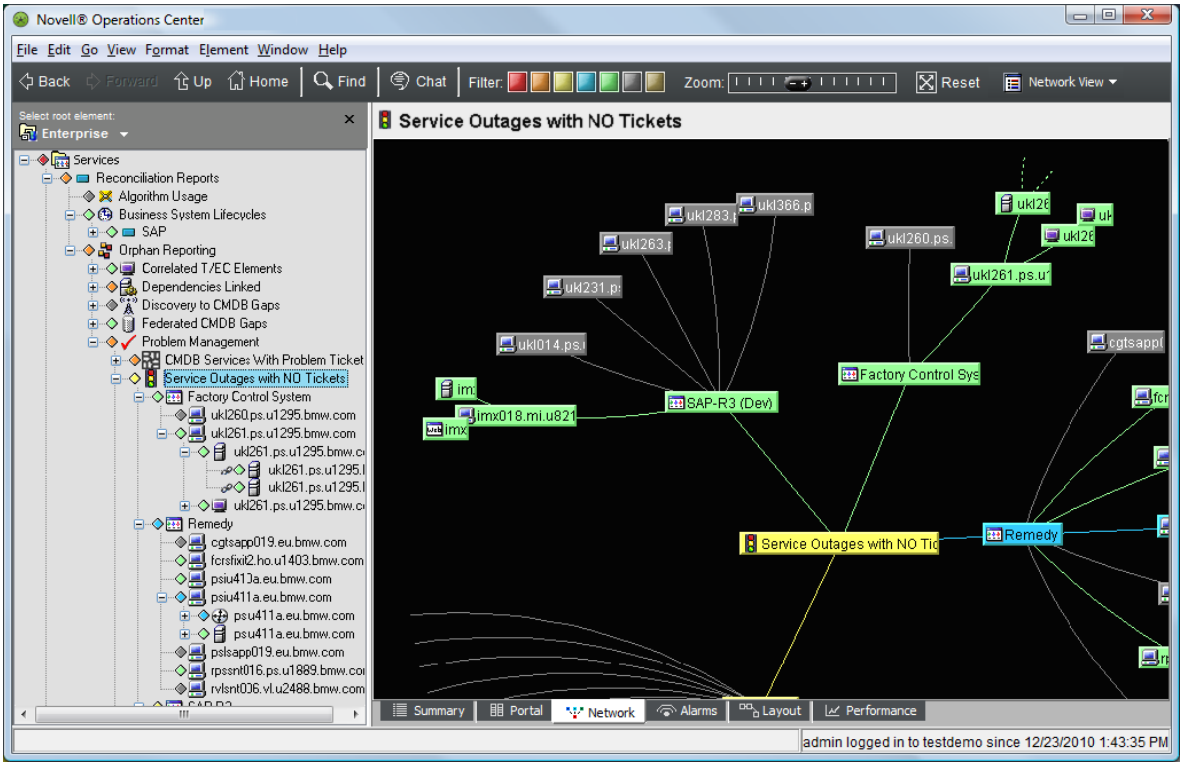
11 After the configuration is run and results are verified, create a schedule for the generation of the service configuration on a routine basis.

12.4 Reconciliation

Assume you have correlated views to gain visibility into your environment. What is the next step? It might be useful to see what might not be integrated in the *SCM* tree. Use discovery tools to determine what is over or under managed.

Figure 12-2 illustrates a view which displays elements that are not integrated in the SCM tree:

Figure 12-2 *Network View*



- Section 12.4.1, “Scenario Description: Show Critical Services (Outages) with No Matching Trouble Tickets,” on page 194
- Section 12.4.2, “Scenario Description: Show Critical Hosts Only,” on page 199

12.4.1 Scenario Description: Show Critical Services (Outages) with No Matching Trouble Tickets

In this example, the goal is to build a view that shows critical services (outages) that do not have matching trouble tickets.

Using SCM, tickets are matched to the services by using a simple hostname match. Only those services without matches are included (an inverse match).

The resulting *Services* hierarchy shows all critical services without an associated trouble ticket.

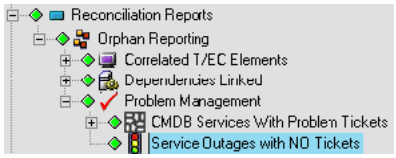
This scenario uses:

- A custom Data Integrator adapter that pulls in database information about IT services and the technology that supports them
- A Remedy ARS adapter that provides trouble ticket information

To use SCM to identify service outages without an associated trouble ticket:

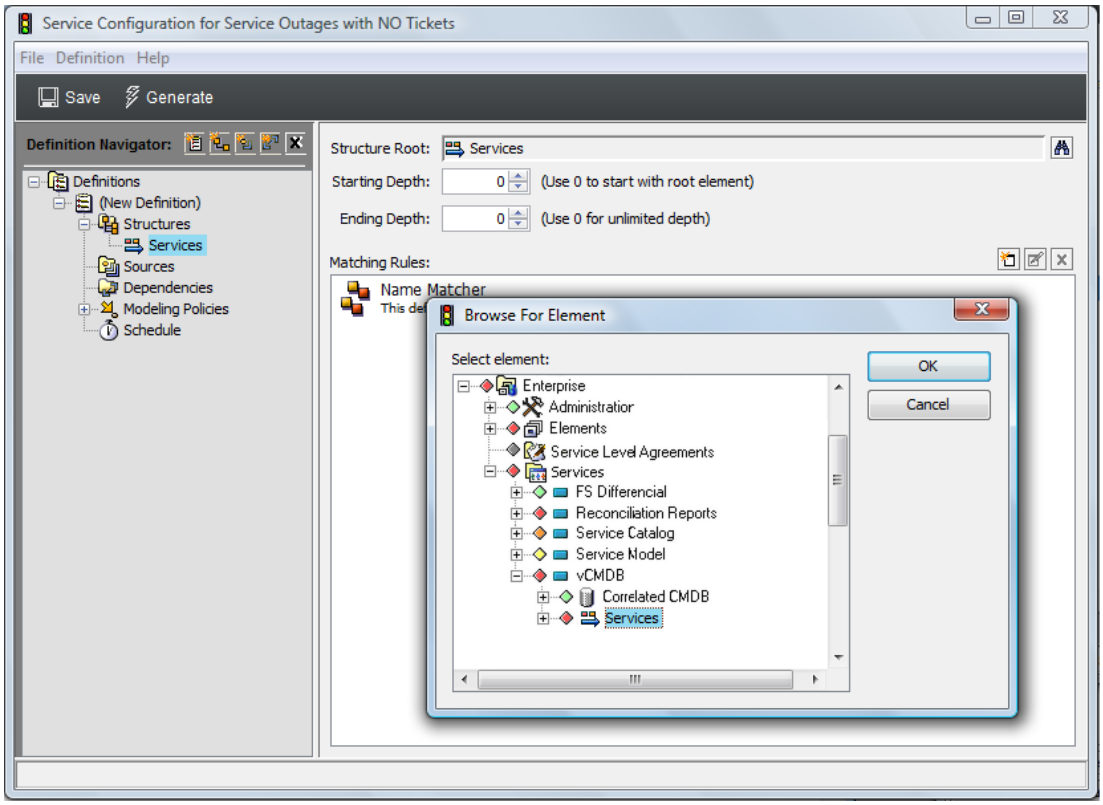
1. Add an element that is the parent of the new tree.

   In this example, the *Service Outages with NO Tickets* element is created under the *Problem Management* element:



2. Right-click the element, then select *Service Configuration* > *Create* to open the Service Configuration Editor.
3. To define the Structures for the element by identifying an existing hierarchy that drives the shape of the new tree, create a Structure definition that adds the *Services* element as a *Structure Root*, as shown in the illustration below.

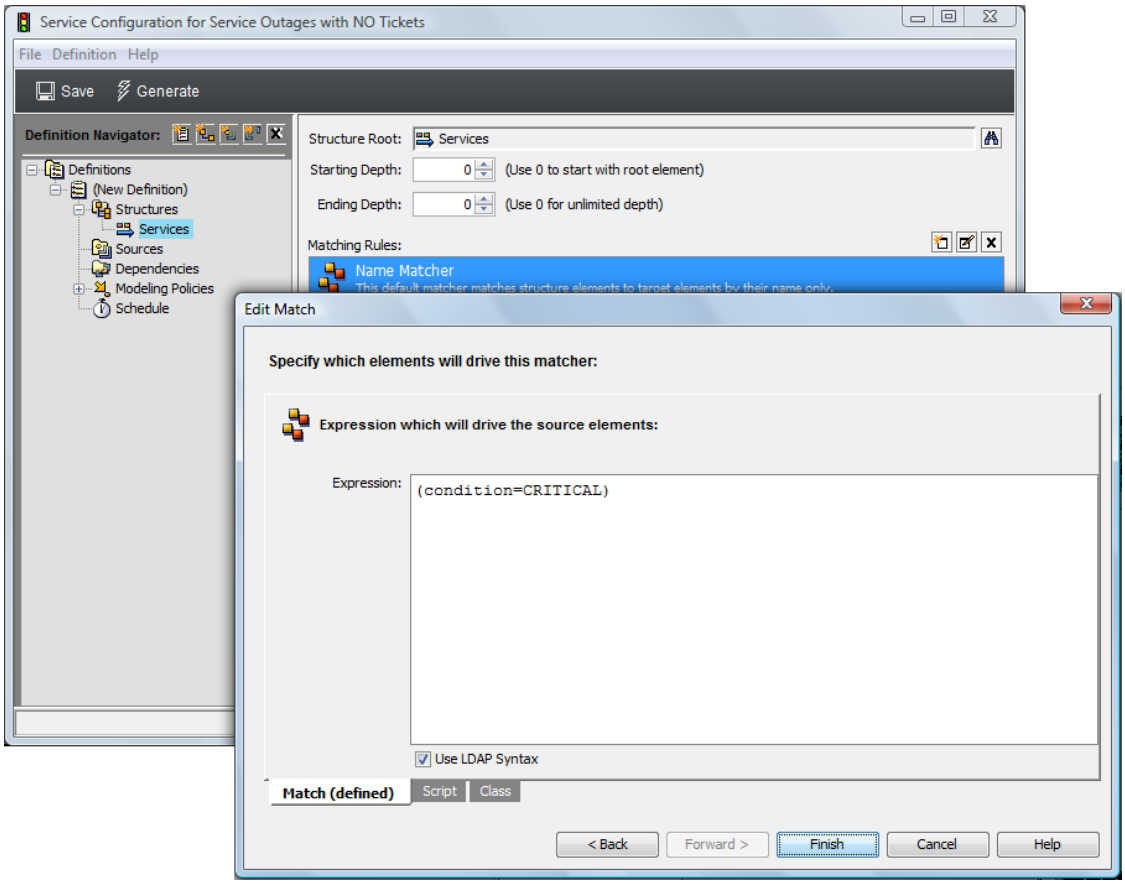
   Services is a hierarchy created by a Data Integrator adapter that pulls in database information about IT services and the technology that supports them.



4 To see all elements contained in the *Services* hierarchy, leave *Starting Depth* and *Ending Depth* at O.

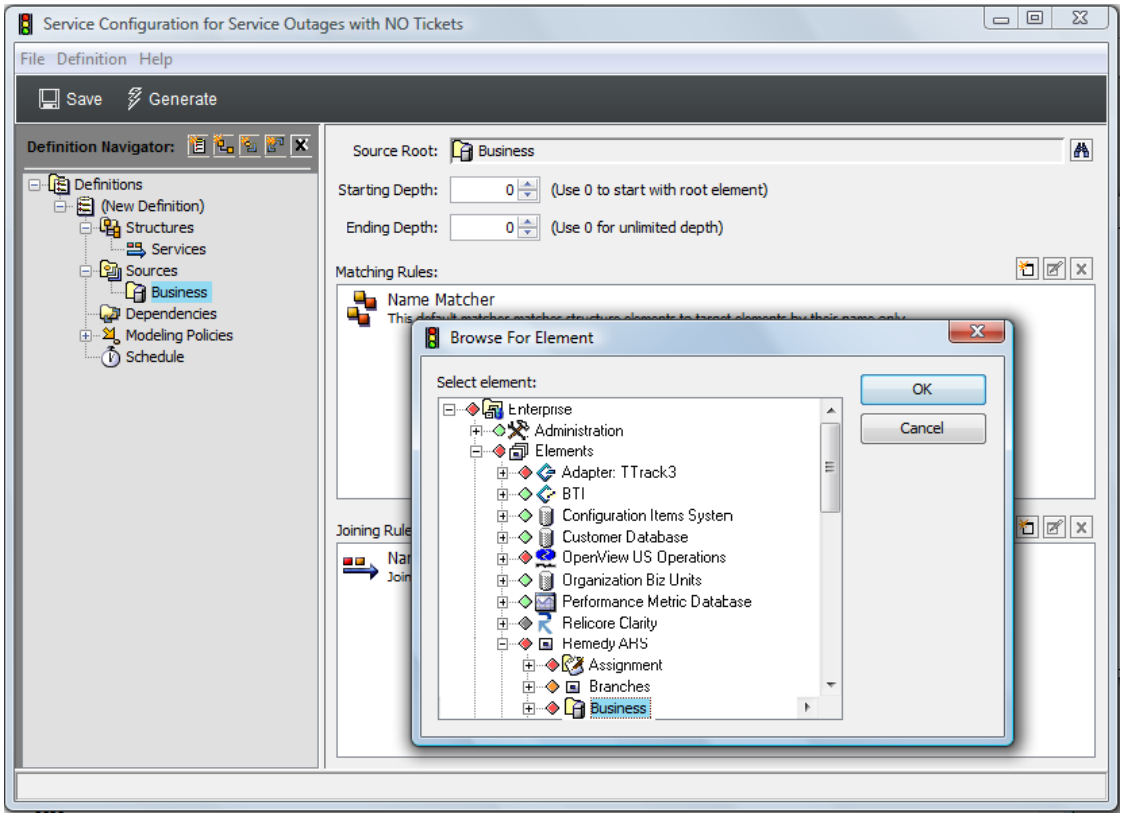
Only outages (elements that are down) without an associated trouble ticket should display.

5 Define a Matching Rule that uses an expression (`condition=CRITICAL`) to replicate only those structure elements that have a CRITICAL condition level.



6 To define the Sources for the configuration by identifying an existing hierarchy that provides state and information to the new elements in the configuration, create a Source definition that adds the *Business* element as the *Source Root*.

Business is a hierarchy from a Remedy ARS adapter integration. It provides trouble ticket information to the new configuration elements based on a join rule using a simple name match.



7 To define *Generation* and *Correlation* policy rules to further define how the configuration is generated and elements are correlated:

7a To use elements found in the *Source* tree to populate the new configuration tree, select the *Display Source Elements As Children* check box in *Generation* policies.

7b To carry over state and properties from *Structure* elements, select the *Create State Relationship to Structure Elements* and *Copy Properties from Structure Elements* check boxes in *Generation* policies.

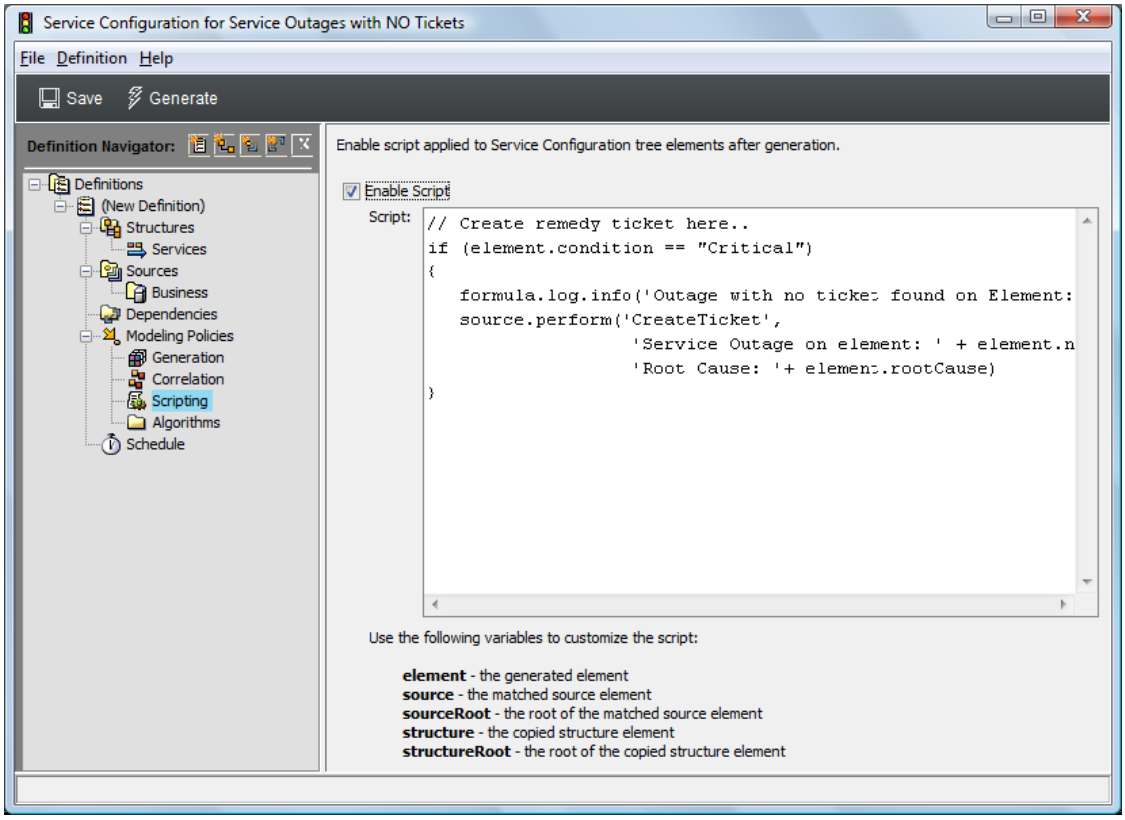
7c To view service outages without tickets, we will want to specify that the configuration show only those services that do not have matching trouble tickets.

For the *Correlation* policy, select the *Inverse Match* radio button to pick up only the elements without matching trouble tickets.

8 (Optional) Define scripts to apply to configuration elements after generation.

In the left pane, click *Scripting*, select the *Enable Script* check box, then enter the script directly in the text area provided.

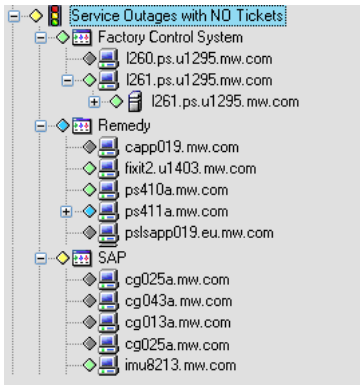
Because this configuration looks for service outages without associated trouble tickets, it could be very useful to then use the configuration to automatically generate a trouble ticket. In this scenario, we use the scripting feature to activate a script that dynamically creates a trouble ticket for services that do not have an existing trouble ticket as shown in the following illustration:



For information on scripting, see the *Operations Center 5.5 Scripting Guide*.

9 Click Save to save configuration settings, then click Generate to generate the new tree.

In this instance, elements displayed with state/condition are no longer considered outages after the configuration was generated.



10 After the configuration is run and results are verified, create a schedule for the generation of the service configuration on a routine basis.

In this case, schedule the configuration to generate every 5 minutes, as it is critical to identify outages that are not being managed properly.

For information on defining the schedule, see Section 8.3.9, "STEP 9: Test and Generate the Configuration," on page 150.

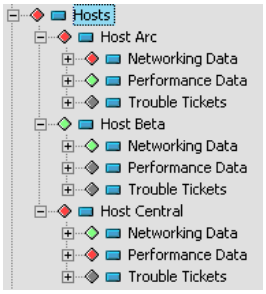
12.4.2 Scenario Description: Show Critical Hosts Only

A scenario similar to the previous one involves selecting specific information from multiple hosts in different geographic locations.

Assume there is a correlated list of hosts, as shown in Figure 12-3. Each host has the following folders:

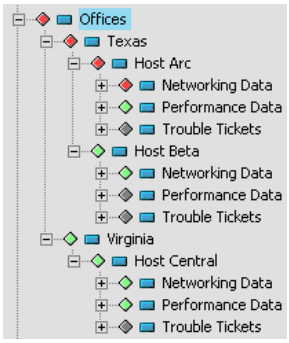
- **Networking:** Links to an adapter with networking alarm data
- **Performance:** Links to an adapter with performance metric data
- **Trouble Tickets:** Links to a trouble ticket system adapter

Figure 12-3 *Correlated Hosts Hierarchy*



Assume there is also a geographic correlation hierarchy that shows the hosts located in two different offices, Texas and Virginia, as shown in Figure 12-4:

Figure 12-4 *Element hierarchy showing hosts in two office locations*



Assume you want to create an element hierarchy named *Critical Hosts*, which shows only the hosts that have a CRITICAL condition. The hosts can be in either the Texas or Virginia office. The list changes over time, as host conditions change.

Using the previous scenario as a model, to create a configuration that shows only CRITICAL hosts:

1. In the Explorer pane, add an element that is the parent of the new tree.

   In this example, the *Critical Hosts* element is created under the *Services* element.

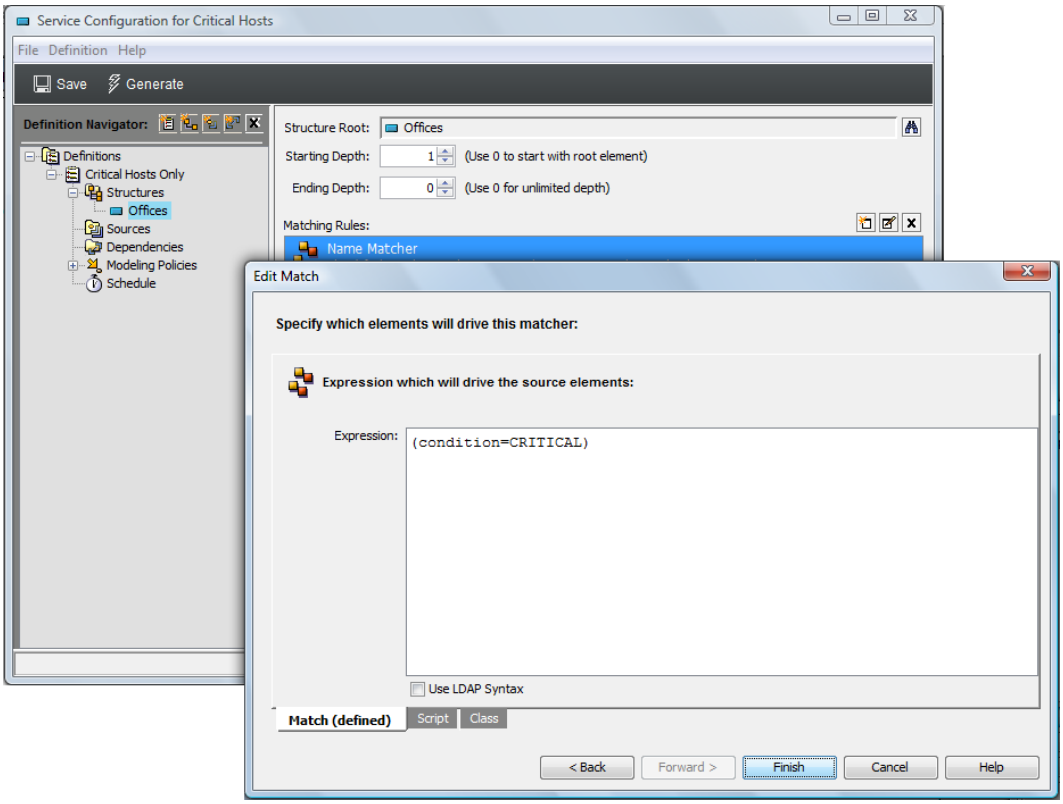
2. Right-click the element, then select *Service Configuration* > *Create* to open the Service Configuration Editor.

3. To define the Structures for the element by identifying an existing hierarchy that drives the shape of the new tree:

   3a Create a Structure definition that adds the *Offices* element as a *Structure Root*.

   3b To display the elements contained in the *Offices* hierarchy, starting with the office location names, which are 1 level beneath the root, enter 1 in the *Starting Depth* field and leave *Ending Depth* at O.

   3c Because the new configuration should contain only those elements that are in a CRITICAL state, define a Matching Rule that uses an expression (`condition=CRITICAL`) to replicate only those structure elements that have a CRITICAL condition level.



4. To define the Sources for the configuration by identifying an existing hierarchy that provides state and information to the new elements in the configuration, create a Source definition that adds the *Offices* element as the *Source Root*.

   Offices is a hierarchy that lists hosts by office location. Use the default join rule for a simple name match.

5 To define *Generation* and *Correlation* policy rules to further define how the configuration is generated and elements are correlated:

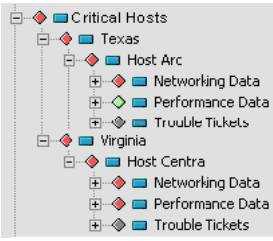
5a To use elements found in the *Source* tree to populate the new configuration tree, select the *Display Source Elements As Children* check box in *Generation* policies.

5b To carry over state and properties from *Structure* elements, select the *Create State Relationship to Structure Elements* and *Copy Properties from Structure Elements* check boxes in *Generation* policies.

5c To display all elements that meet the *Structures*, *Dependencies*, and other *Modeling Policies* rules, leave *Static Match Correlation* policy.

6 Click Save to save configuration settings, then click Generate to generate the new tree.

The following illustration shows the generated *Critical Hosts* hierarchy, which displays only those host elements with a conditon of `CRITICAL`, using the location structure of the existing *Offices* element hierarchy:



7 Expand the hosts to identify the `CRITICAL` elements that contribute to the host condition.

13 Use Case: Reporting Service Configuration Problems

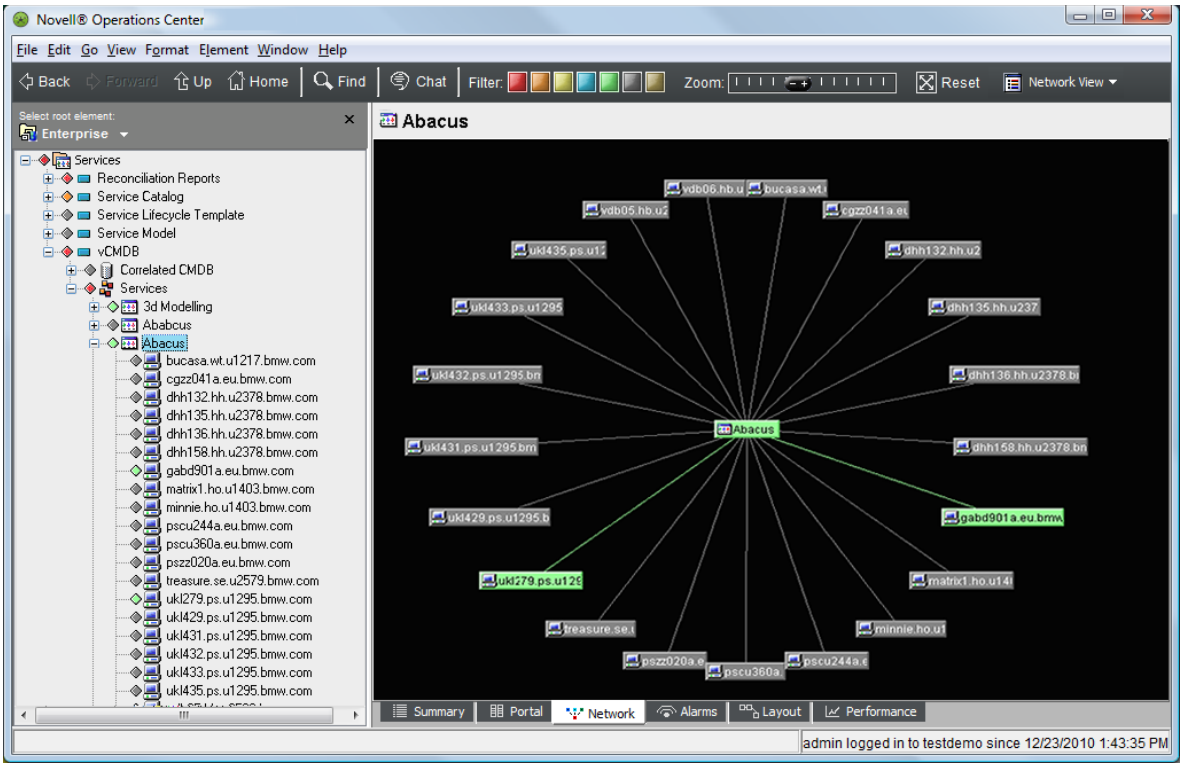
SCM provides custom and correlated views and can also help identify services that are under-instrumented, gaps in your service views, areas that lack data. Use SCM to create views that show what is not being managed or provide information for problem management.

- Section 13.1, “Identifying Unmanaged Services,” on page 203
- Section 13.2, “Problem Management,” on page 208

13.1 Identifying Unmanaged Services

Use SCM to identify services that are unmanaged. By creating a SCM definition that links all sources to your services, it becomes obvious which services are not managed. These services display a gray or UNMANAGED condition state, as shown in Figure 13-1:

Figure 13-1 *Network View: Services that are not being managed have a gray (unmanaged) condition state.*



The following section steps through an example of how SCM is set up to identify services that are not managed in a network.

- Section 13.1.1, “Scenario Description: View All Services That are Not Managed,” on page 204
- Section 13.1.2, “Scenario Steps,” on page 205

13.1.1 Scenario Description: View All Services That are Not Managed

In this example, a basic view of services and the technology that supports them is created. Then, services that are not being managed are identified.

SCM leverages two T/EC adapters and one HP OpenView adapter, to feed source information to a *Services* hierarchy. Data matches the services using a simple hostname match. The resulting *Services* hierarchy shows services with state information. Services that have an UNKNOWN or UNMANAGED state are identified as undermanaged.

Figure 13-2 *Elements Tree View*



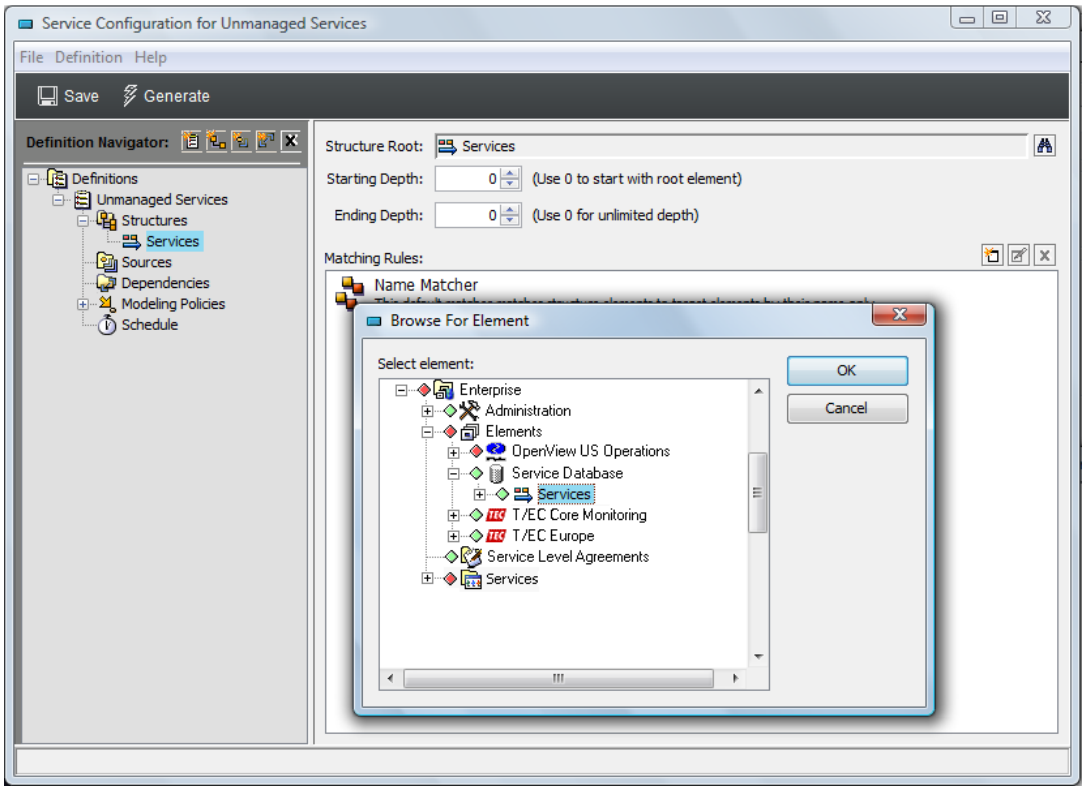
This scenario uses:

- A custom Data Integrator adapter that pulls in database information about IT services and the technology that supports them
- Two Tivoli T/EC Adapters that provide state information about related technology
- An HP OpenView Network Node Manager Adapter that provides state information about related technology

13.1.2 Scenario Steps

To use SCM to identify services that are not under management:

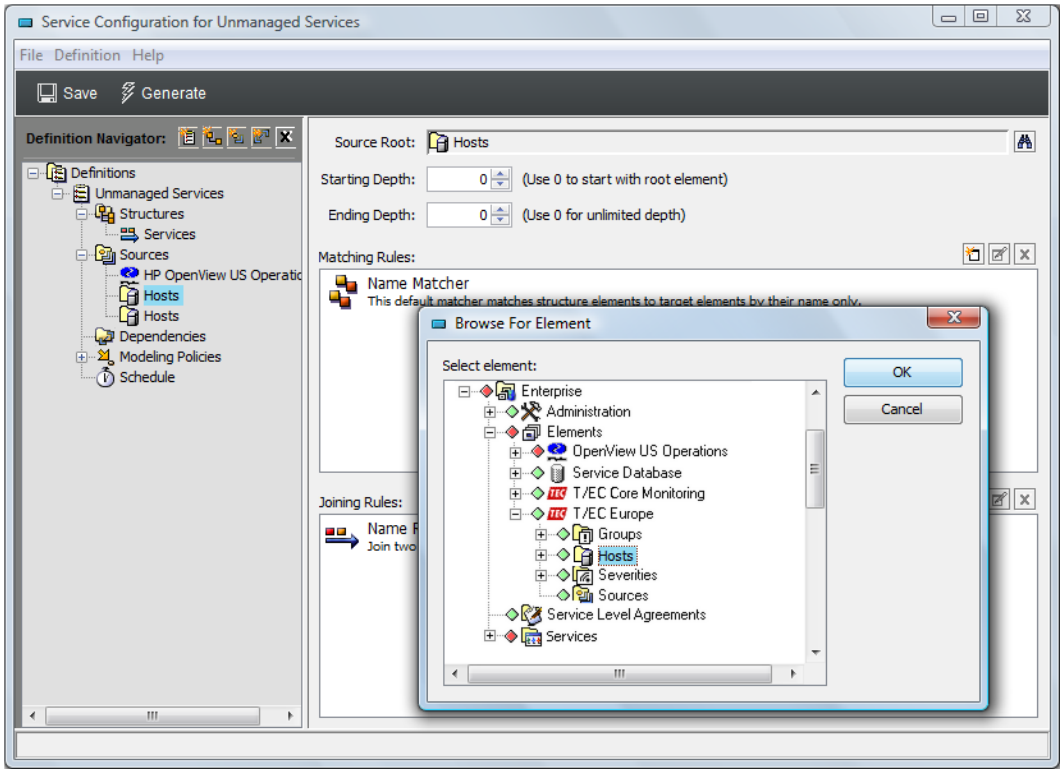
1. In the Explorer pane, add an element that is the parent of the new tree.
2. Right-click the element, then select *Service Configuration* > *Create* to open the Service Configuration Editor.
3. To define the Structures for the element by identifying an existing hierarchy that drives the shape of the new tree, create a Structure definition that adds the *Services* element as a *Structure Root*, as shown in the following illustration:



Services is a hierarchy that is created by a Data Integrator adapter that pulls in database information about IT services and the technology that supports them.

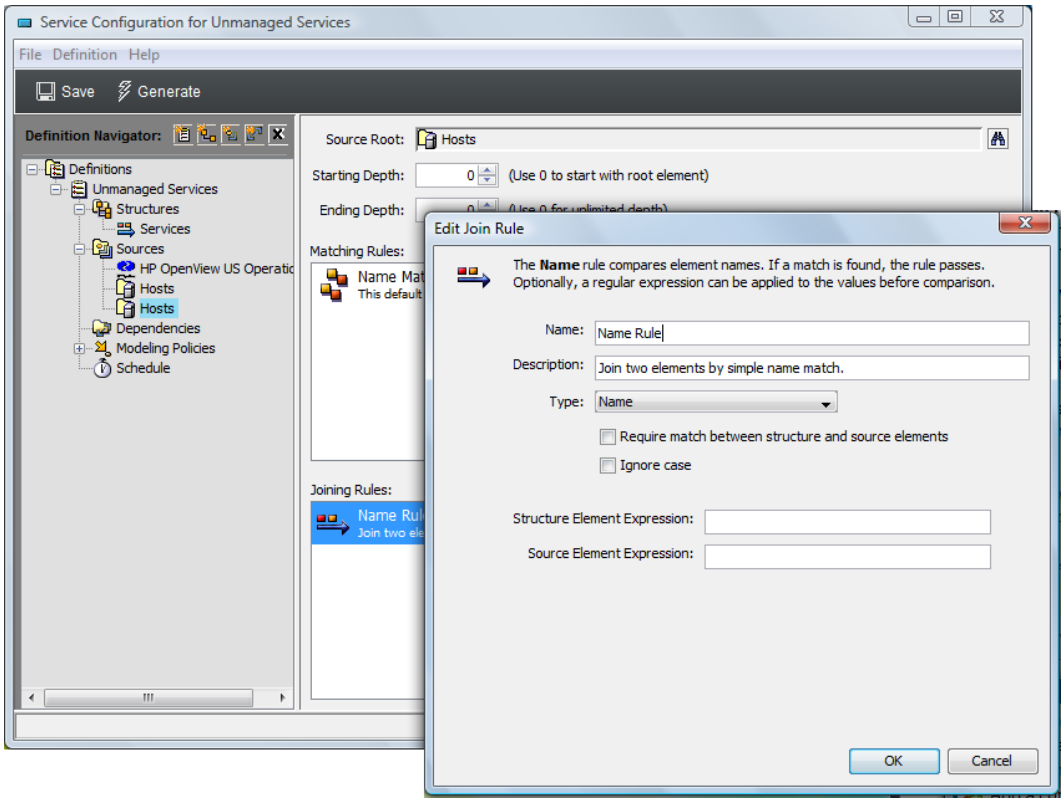
4 To define the Sources for the configuration by identifying an existing hierarchy that provides state and information to the new elements in the configuration:

4a For this example, add two Source definitions that specify Hosts from T/EC adapters as the *Source Root* element:



Hosts is a hierarchy from a T/EC adapter integration. It provides source information to the new configuration elements based on a simple name join.

4b Also add a Source definition that uses OpenView US Operations (an HP OpenView adapter) as a *Source Root* element:



5 Define *Generation* and *Correlation* policy rules to further define how the configuration is generated and elements are correlated.

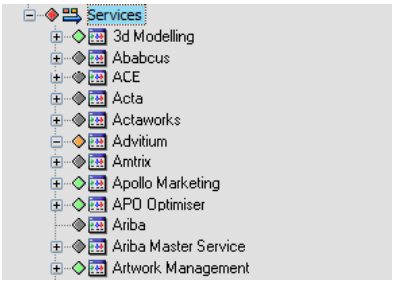
In this scenario, default *Generation* and *Correlation* modeling policies are sufficient to generate the configuration tree as required.

The default Static Match correlation allows elements and information to match as specified within the Structure and Source definitions.

6 Click Save to save configuration settings.

7 Click Generate to generate the new tree.

Looking at the new *Services* tree, it is easy to identify services that do not have any related technology monitoring them. These are the services that are undermanaged (gray state).



8 After the configuration is run and results are verified, create a schedule for the generation of the service configuration on a routine basis.

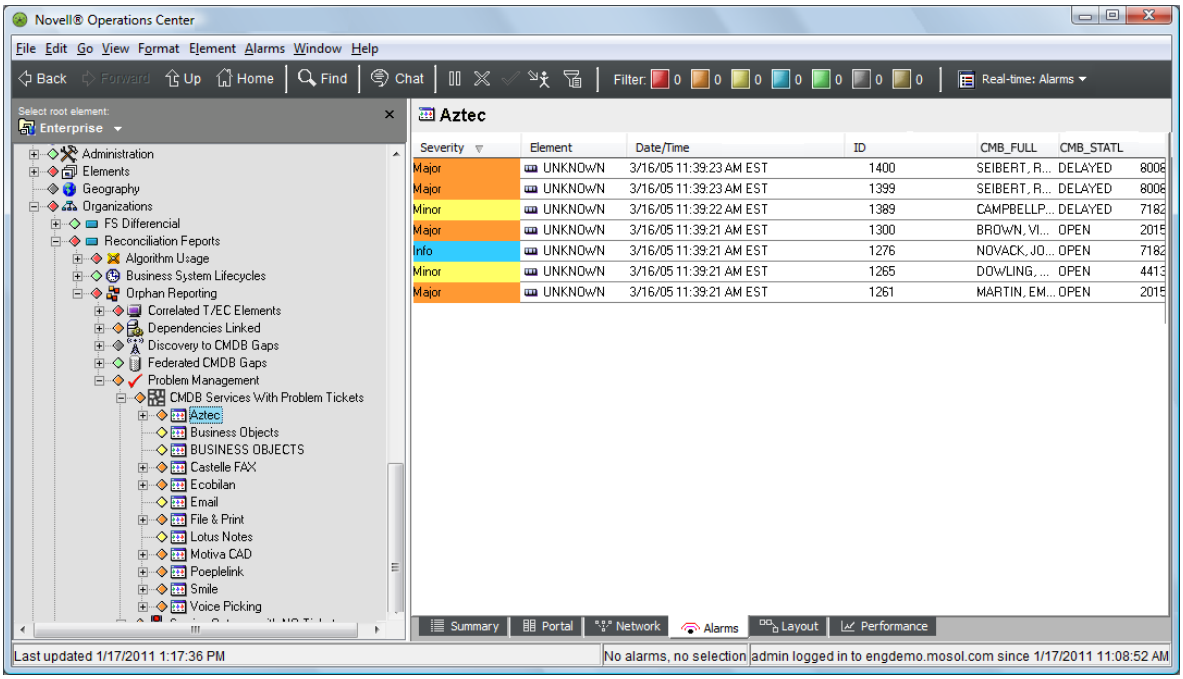
In this case, run the report manually on an ad-hoc basis when there is demand for asset and instrumentation coverage.

13.2 Problem Management

How do problems affect the state of services? Which services are affected by a particular problem? Are there related problems?

Use SCM to create a tree that shows all services with corresponding trouble tickets. Figure 13-3 shows services with problem tickets:

Figure 13-3 *Alarms View*



The following scenario shows how a SCM definition is created to overlay services with trouble tickets from a Remedy ARS system.

- Section 13.2.1, “Scenario Description: Managing Trouble Tickets by Service,” on page 209
- Section 13.2.2, “Scenario Steps,” on page 209

13.2.1 Scenario Description: Managing Trouble Tickets by Service

Feed trouble ticket information directly to the *Services* hierarchy. Tickets are matched to the services by using a simple hostname match. The resulting *Services* hierarchy shows services with trouble ticket information. Trouble tickets can then be managed by associated service.

This scenario uses:

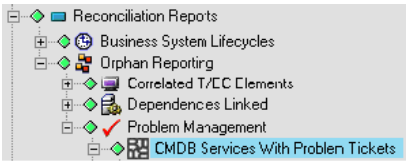
- A Data Integrator adapter that taps into an internal IT-maintained database

  It organizes technology (server information) by the service that it supports.
- A Remedy ARS adapter that provides trouble ticket information

13.2.2 Scenario Steps

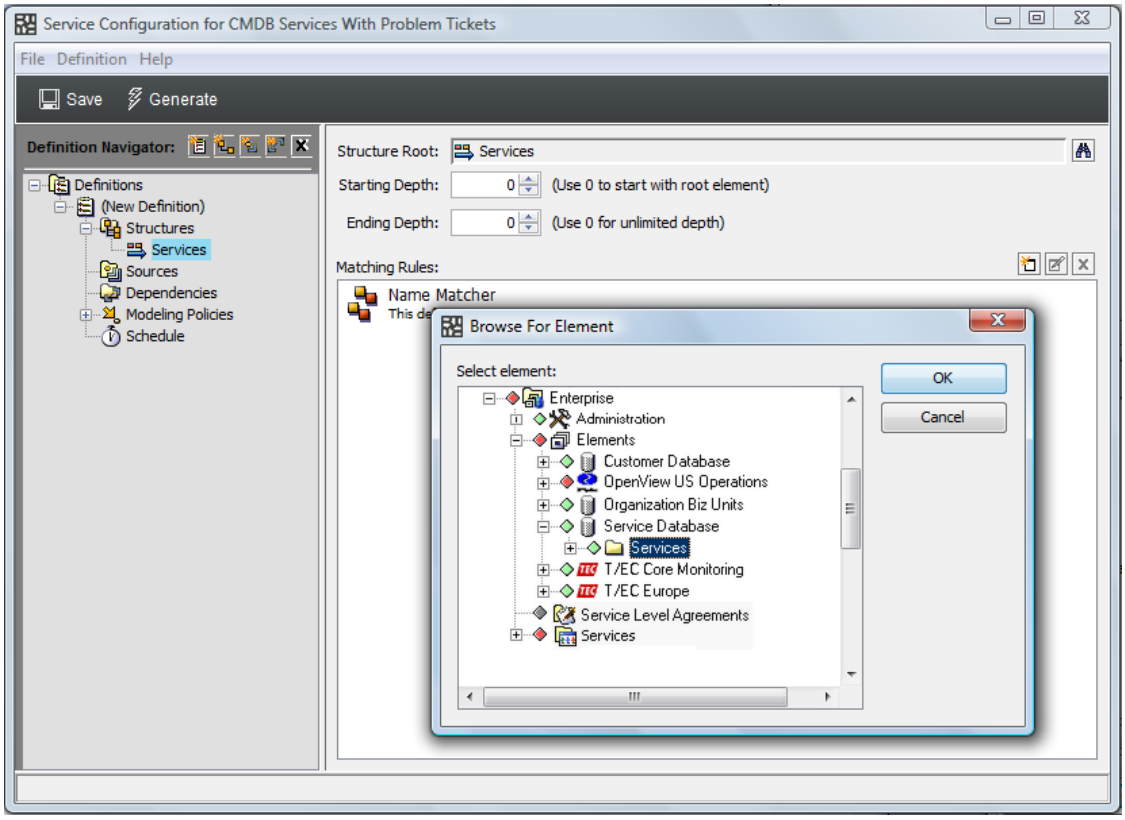
To use SCM to create a configuration to manage problem tickets:

1. In the Explorer pane, add an element that is the parent of the new tree.
2. In this example, create a *CMDB Services With Problem Tickets* element under the *Reporting* element:



3. Right-click the element, then select *Service Configuration* > *Create* to open the Service Configuration Editor.

4 To define the Structures for the element by identifying an existing hierarchy that drives the shape of the new tree, create a Structure definition that adds the *Services* element as a *Structure Root*, as shown in the illustration below.



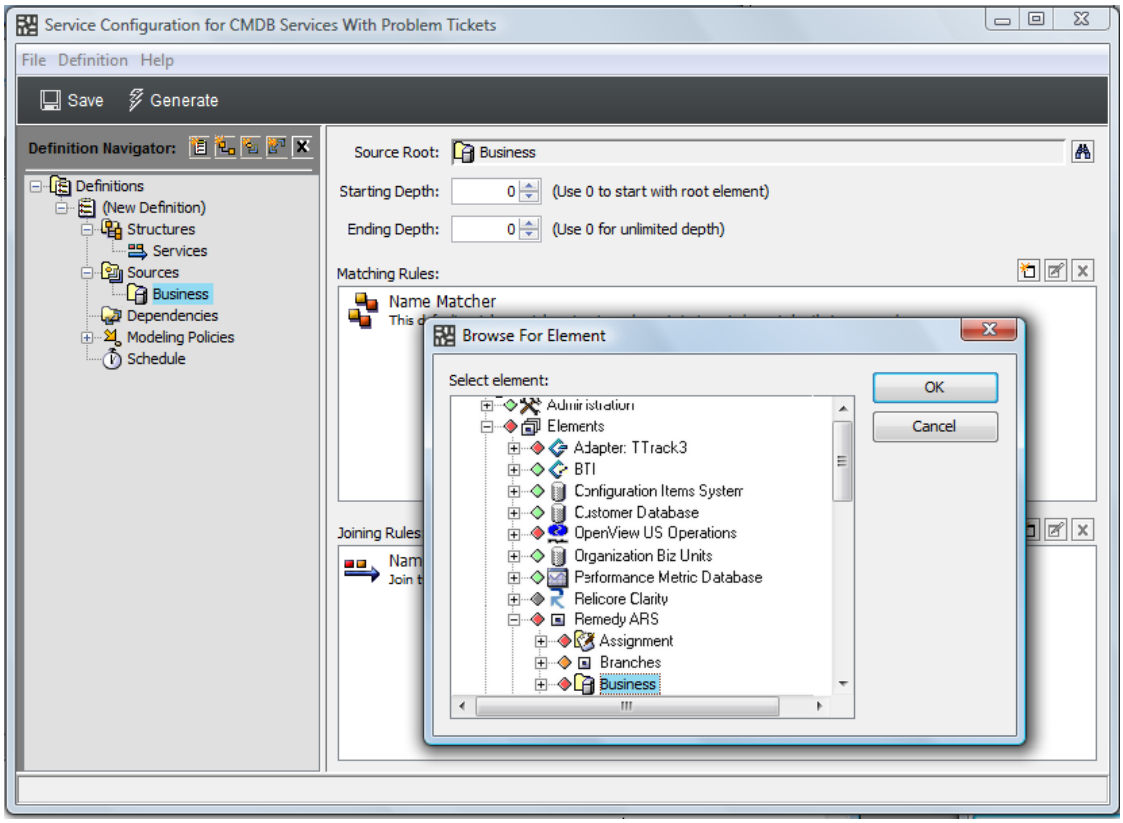
In this case, the *Services* hierarchy is a branch from a Data Integrator adapter that taps into an internal IT-maintained database. It organizes technology (server information) by the service that it supports.

5 To view all elements contained in the *Services* hierarchy, leave *Starting Depth* and *Ending Depth* at 0.

6 Use the default Matching Rule to apply a simple name match to select elements.

7 To define the Sources for the configuration by identifying an existing hierarchy that provides state and information to the new elements in the configuration, create a Source definition that uses the *Business* element as the *Source Root*.

Business is a hierarchy resulting from a Remedy ARS adapter integration. It provides trouble ticket information to the new configuration elements based on a simple name join.



8 Use the default join rule to link state information to structure elements using a simple name match.

This definition allows state and property information to be available for any new elements found when the configuration searches for dependencies.

9 To define *Generation* and *Correlation* policy rules to further define how the configuration is generated and elements are correlated:

9a To specify that Structures define the elements in the new configuration and Source data provide state, retain all the generation defaults, but do not select any *Generation* policy options.

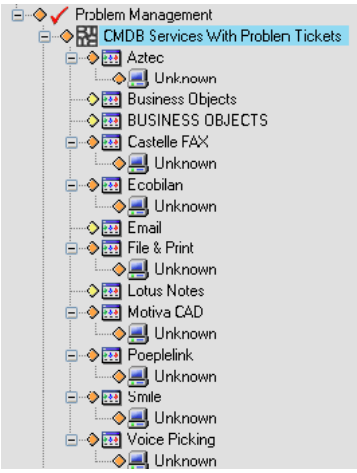
9b To match all elements and sources, retain the default *Static Match* selection in *Correlation* policies.

Those objects that are matched are included in the new configuration tree.

10 Click Save to save configuration settings.

11 Click Generate to generate the new tree.

The new hierarchy is built under the *CMDB Services With Problem Tickets* element in the *Operations Center Services* hierarchy:



12 Use the *Alarms* tab to review problem tickets for the appropriate elements.

13 After the configuration is run and results are verified, create a schedule for the generation of the service configuration on a routine basis.

Schedule this configuration to generate every five minutes as problem management is always an immediate requirement.

III Using The View Builder Repository

The View Builder Repository allows you to create, edit, and share element hierarchies in the *Services* > *Service Models* tree. Use the View Builder to dynamically create or maintain branches in the *Service Models* hierarchy instead of manually creating them element by element.

The View Builder Repository is an XML store of element hierarchy and information. An XML stream dynamically takes that information and populates the hierarchies. After the XML file is created, use the *Run* option in the View Builder Editor to build the hierarchy.

- Chapter 14, “Editing View Builder Repository Files,” on page 215
- Chapter 15, “Importing and Exporting Element Hierarchies,” on page 221
- Chapter 16, “Executing XML Files to Update Elements,” on page 225

14 Editing View Builder Repository Files

The View Builder Repository uses XML files to store element information and dynamically create element hierarchies. These XML files can be edited and maintained using any text editor, XML editor or the View Builder Editor.

Any structure in the *Service Models* hierarchy can be saved as an XML file using the View Builder Repository's `Export` command. The file can then be edited for required changes or shared with other installations.

This section examines a sample XML file and provides instructions on how to edit and maintain the files using the View Builder Editor.

- Section 14.1, "Understanding XML Tags for Operations Center Elements and Views," on page 215
- Section 14.2, "Editing XML Files," on page 217
- Section 14.3, "Adding Elements or Comments," on page 218
- Section 14.4, "Moving Elements," on page 219
- Section 14.5, "Cutting, Copying, and Pasting Tags," on page 219
- Section 14.6, "Deleting Tags," on page 219

14.1 Understanding XML Tags for Operations Center Elements and Views

By examining any XML file, you begin to see from the various tags it contains that data is consistently categorized and identified within a hierarchical structure.

The well-formed XML file for Operations Center hierarchies contains basic tags and commands. At the simplest level, the XML file must contain this basic structure:

```
<views>
  <tree>
    <element>element definition tags</element>
  </tree>
</views>
```

Looking the following sample XML file that would be used to create a hierarchy of elements in Operations Center, we can see that the hierarchy created would contain a *Test Element* top element with a native child (*Child Element*) and a link to an existing technology branch (Tivoli TEC+):

```
<!DOCTYPE views PUBLIC "-//NetIQ, Inc.//DTD views 1.0//EN" "http://www.NetIQ.com/
dtds/views_1.0.dtd">

<views destroy="no">
    <tree start_at="root=Organizations">
      <element>
        <name>Test Element</name>
        <class>org</class>
        <displaySourceElements>false</displaySourceElements>
        <contact>Jim</contact>
        <company>Enigma Corp</company>
        <address>394 First Street, Fairfax, VA</address>
        <phone>703-555-1212</phone>
        <email>test@mo.com</email>
        <secure name="simple" related="yes" self="yes" children="no">
          <grant names="GroupOne,GroupTwo" permissions="view,manage,access,define"/
>
          <deny names="GroupThree" permissions="access,define"/>
        </secure>
        <element>
          <name>Child Element</name>
          <class>org</class>
          <displaySourceElements>true</displaySourceElements>
          <contact>John</contact>
          <company>Enigma Corp</company>
          <address>394 First Street, Fairfax, VA</address>
          <phone>703-555-1212</phone>
          <fax>703-555-3939</fax>
          <pager>322-325-3565</pager>
          <sref name="simple"/>
          <relate kind="ORG">script=TivoliTec+/root=Elements</relate>
        </element>
      </element>
     </tree>
</views>
```

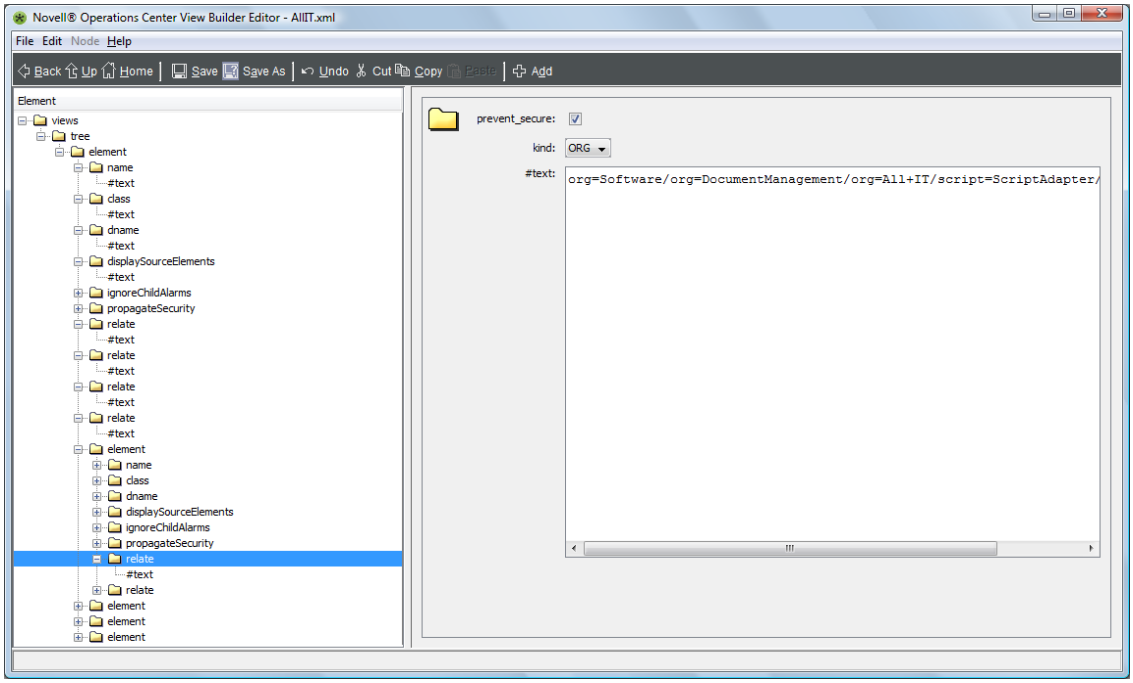
For more information about the various XML tags used by Operations Center for the creation of element hierarchies, see *Understanding XML Tags* in the *Operations Center 5.5 Server Configuration Guide*.

14.2 Editing XML Files

View Builder Repository files are XML files that can be edited using any text editor or with the View Builder Editor. Because the View Builder Editor is an XML editor that is provided with the Operations Center installation, it is easier to edit and maintain XML files from the View Builder Repository.

To edit a View Builder XML file:

1. In the *Explorer* pane, expand *Administration* > *Server* > *View Builder Repository*.
2. Right-click the XML file, then select *Edit* to open the View Builder Editor:



3. Make the required modifications to the file:

Option	Function
Back	Returns focus to the element that was previously selected.
Up	Select the parent of the currently selected element. Continue clicking to move up the hierarchy.
Home	Select the top of the hierarchy.
Save	Saves the XML file using the current name.
Undo	Undo any updates or modifications made on the hierarchy.
Cut	Cuts the selected text from the current XML file.
Copy	Copies the selected text into the current XML file.

Option	Function
Paste	Pastes the cut of copied text into the current XML file.
Add	Adds a new element or new text to the XML file.
File > *Revert*	Abandons all changes made in the current session and opens the saved version of the XML file.
File > *Save* or *Save as*	Saves the XML file currently loaded in the editor.
File > *Exit*	Closes the XML editor.

For more information about XML tags and requirements, see Section 14.1, "Understanding XML Tags for Operations Center Elements and Views," on page 215 or the `/OperationsCenter_install_path/database/examples/views_1.0.dtd` file.

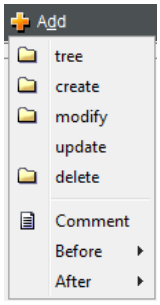
4 Click *File* > *Save*.

14.3 Adding Elements or Comments

To add an element or text comment to a file:

1 Right-click an element in the left pane, click *Add*, then select one of the options from the submenu:

- ***Element type*:** Available elements varies, depending on the DTD associated with the XML file. Only those elements that are valid for the entry point selected display in the menu.
- **Comment:** Allows you to enter comment text that is visible for the element in the XML editor but are not processed when the XML file runs.



2 To place a new element before or after the current element, select *Before* or *After*, then select the desired element or comment option.

3 Specify the attributes for the element or enter comment text in the right pane.

Required attributes are identified by a red flag in the right pane. Most attribute fields allow typing values. Some have drop-down lists.

Attributes with check boxes represent fields with a Yes/No value. Select the check box to indicate Yes (or True) and leave it deselected to indicate No (or False).

4 Save the file.

14.4 Moving Elements

Elements (XML tags) can be moved using drag and drop within the current file.

To move a tag element, drag and drop the tag to the target tag element.

The moved tag is placed as child of the target tag.

14.5 Cutting, Copying, and Pasting Tags

Use the *Copy*, *Cut*, and *Paste* buttons to rearrange tag elements:

- *Copy* and *Paste* are also listed as options under the *Edit* menu
- *Copy* and *Cut* include all of the children of the selected tag
- The *Paste* option provides the choice of pasting before or after the target tag at the same level, or into the target tag as a child tag

14.6 Deleting Tags

To delete a tag and all its children, right-click the element and select *Delete Item*.

15 Importing and Exporting Element Hierarchies

Use the View Builder Repository's import and export features allow you to save and share element hierarchies among Operations Center installations.

The XML file by a View Builder export captures the following information about elements in the hierarchy:

- Children
- Scripts
- Algorithms
- Algorithm parameters
- Match expressions
- General contact information (name, company, address, phone, fax, pager, e-mail, web site)
- Layout graphic references
- Links
- Latitude (*Locations* elements only)
- Longitude (*Locations* elements only)

The following sections describe:

- Section 15.1, "Importing an External XML File into the View Builder Repository," on page 221
- Section 15.2, "Saving (Exporting) an Element Hierarchy Structure," on page 222
- Section 15.3, "Performing a Server-Based Export," on page 223

15.1 Importing an External XML File into the View Builder Repository

Imported XML files need to be well-formed XML View Builder files containing necessary DTD headers, as shown in the example XML code in the example in "Understanding XML Tags for Operations Center Elements and Views" on page 215.

To import an XML file into the View Builder Repository:

1. In the *Explorer* pane, expand the *Administration* root element > *Server*.
2. Right-click *View Builder Repository*, then select *Import* to open the Import dialog box.
3. Click *Browse* to select the XML file.

   The Open dialog box opens.

4 Navigate to the file, then click *Open* to select the file.

The Open dialog box closes.

5 Click *OK* to import the XML file.

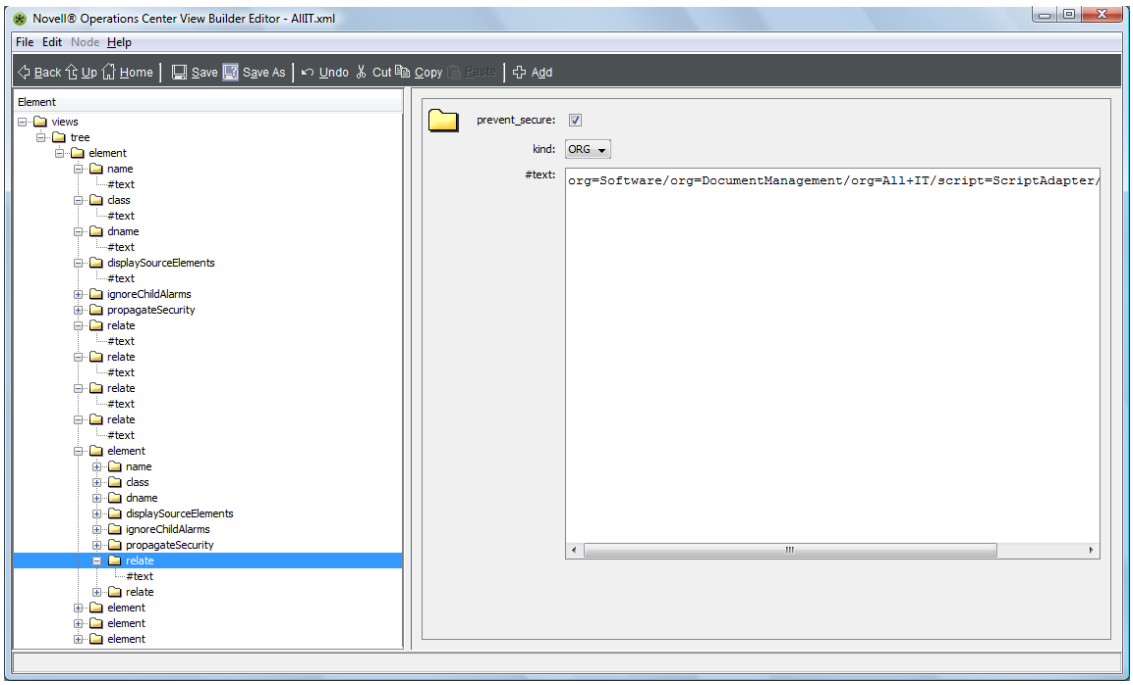
An element corresponding to the filename is created under the *View Builder Repository* element. The imported XML file saves to the `/OperationsCenter_install_path/database/ViewBuilder` directory.

15.2 Saving (Exporting) an Element Hierarchy Structure

To save (export) an element hierarchy structure as an XML file in the View Builder Repository:

1 In the *Explorer* pane, expand the *Administration* root element > *Server*.

2 Right-click *View Builder Repository*, then select *Export* to open the Export dialog box:



3 Click *Browse* to navigate to the element to export.

The Choose Element dialog box opens.

4 Select the element, then click *OK* to close the Choose Element dialog box.

5 Click *Browse* to select a destination file.

The Open dialog box opens.

6 Navigate to the desired directory and specify an XML filename.

7 Click *Open* to accept the destination directory and filename.

8 Select the *Import as a View Builder Repository Element* check box to be able to access, edit or execute the XML file from the View Builder Repository.

   The XML file is saved to the `/OperationsCenter_install_path/database/ViewBuilder` directory during export and can be edited later using the XML editor.

   A new element displays under *View Builder Repository* after export.

9 Click *OK* to create the XML file.

15.3 Performing a Server-Based Export

Because some views can be very large, exporting them through a script on the Operations Center server can be most efficient.

To perform a server-based export:

1 Modify the following script example to save the view to a valid directory:

```
element.perform (session, "ExportToFile"; [], [‘root=Organizations', ‘d;/
formula351/vbtest2.xml']);
```

2 Create an operation on the Operations Center server that calls the script on *Administration* > *Server* > *View Builder Repository*.

3 Execute the script.

16 Executing XML Files to Update Elements

XML files can be executed while Operations Center is running. Elements can be created or modified. If an element already exists, it is updated; otherwise, it is created.

To execute ViewBuilder XML files:

1. In the *Explorer* pane, expand *Administration* > *Server* > *View Builder Repository*.
2. Right-click the desired XML file, then select *Run*. The XML file guides Operations Center software in updating or creating a new element structure.

A LDAP Expression Syntax for Searching and Matching

Regular expressions or LDAP-style syntax is used in Operations Center in two primary functions:

- Filtering searches
- Matching and selecting elements based on element DName

Operations Center filter and match attributes adheres to the standard LDAP syntax defined in RFC2254, The String Representation of LDAP Search Filters. For a comprehensive description of the search and filter syntax, see the RFC2254 document web page (http://tools.ietf.org/rfcmarkup?rfc=2254).

The syntax allows entering one or more attr/value pairs, surrounded by parentheses, and logically combined with "and" (&) or "or" (|) operators. The following operators can be used in the search criteria:

`=, <, >, <= , >=`

For example, to find an object named server1, enter `(cn=server1)`, where cn represents common name, which is a standard way of identifying an object name.

Table A-1 *LDAP Expression Examples*

To find…	Do this….
All objects starting with the same letters.	Specify wildcards using one or more asterisks (*).For example, to find all objects that begin with server, enter the query as: `(cn=server*)`.
All host objects.	Enter `(objectClass=*host*)` to find all hostname objects, all ServerHosts, and so on.
All hosts whose names begin with the letters server.	Combine two `(attr=value)` clauses using the logical "and" operator: `(&(cn=server*)(objectClass=*host*))` Notice how the logical operator appears before the two clauses; this is called prefix notation.
All hosts whose names begin with the letters server and are in a CRITICAL state.	Enter: `(&(&(cn=server*)(objectClass=*host*))(condition=CRITICAL))`
All hosts whose names begin with the letters server and are in either a CRITICAL or WARNING state.	Enter: `(&(&(cn=server*)(objectClass=*host*))(|(condition=CRITICAL)(condition=WARNING)))`

To find…	Do this….
Hostnames that begin with specific letters and are members of a specific object class.	Enter: `&(objectClass=Router)(hostname>B*)`
Elements that are members of a specific class and have a response time greater than or equal to a specified threshold value.	Enter: `&(objectClass=server)(responseTime>=3.4)`