Custom Drawing and Layout Guide

Operations Center 5.0

October 27, 2014

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 9

1.1 Default Layout View . . . 10
1.2 Custom Drawings . . . 10
1.3 Where Do I Start? . . . 11

2 Getting Started with the Layout View 13

2.1 Navigating Inside the Layout View . . . 13
2.2 Creating and Editing Custom Drawings . . . 15
2.3 Understanding Unplaced Elements . . . 16
2.4 Moving Drawing Elements . . . 17
2.5 Clearing a Drawing or Reverting to a Previous Drawing or Default Automatic Layout . . . 17
2.6 Setting Security on Graphic Elements . . . 18
2.7 Troubleshooting Tips . . . 18
2.7.1 Empty Drawings . . . 18
2.7.2 Identifying if a Shape is Associated to an Element . . . 18

3 Using Basic Drawing Tools and Concepts 19

3.1 Understanding the Layout View Panels and Drawing Palette . . . 20
3.2 Selecting Drawing Objects . . . 21
3.3 Adding Shapes and Lines . . . 22
3.4 Changing Fill Colors . . . 24
3.5 Changing the Background Color . . . 24
3.6 Using Undo . . . 25
3.7 Manipulating Shapes and Objects . . . 25
3.8 Using the Grid to Align Objects . . . 26
3.9 Grouping and Ungrouping Objects . . . 27
3.9.1 Working with Grouped Components . . . 27
3.10 Using Containers to Control Objects . . . 28
3.10.1 Grouping Components with a Container . . . 29
3.10.2 Adjusting the Wrap Settings . . . 29
3.10.3 Adjusting Container Layout . . . 30

4 Working with Clip Art and Graphics 33

4.1 Importing Graphics . . . 34
4.2 Using Graphics from the Clip Art Library . . . 35
4.2.1 Linking Clipart from the Graphics Library . . . 35
4.2.2 Embedding Clipart Images . . . 37
4.3 Managing Clipart Files . . . 37
4.4 Creating Graphics for the Clipart Library . . . 38

5 Adding Elements and Updating Drawing Components Using Element Properties 39

5.1 Associating Elements to a Drawing . . . 39

5.2 Dynamically Transforming Text, Shapes and Lines with Element Properties . . . 40
5.2.1 Creating an Element Name Heading . . . 41
5.2.2 Changing the Associated Element . . . 42
5.2.3 Dynamically Updating a Shape's Color based on Element Condition . . . 44
5.2.4 Hiding and Showing Elements Based on Condition . . . 47
5.2.5 Coding Custom Behavior . . . 47
5.3 Changing Default Drill-Down Behavior Using Drawing Shortcuts . . . 48
5.4 Displaying an Element's Children using Child Containers . . . 48
5.4.1 Customizing a Child Container . . . 52
5.5 Integrating Performance Metric Charts and Gauges into a Drawing . . . 54
5.5.1 Types of Charts . . . 55
5.5.2 Types of Gauges . . . 56
5.5.3 Adding Charts and Gauges . . . 58
5.5.4 Configuring Chart and Gauge Properties . . . 58

6 Working with Node Styles and Custom Graphics 65

6.1 Understanding the Node Style and the Node Style Library . . . 65
6.2 Changing the Node Style . . . 67
6.2.1 Applying a Node Style to All Elements in a Drawing . . . 68
6.2.2 Changing the Node Style for One Element Only . . . 69
6.3 Applying the Ambient Node Style . . . 69
6.4 Creating and Applying New Node Styles . . . 69
6.4.1 Creating the Node Style . . . 70
6.4.2 Testing the Node Style Using Preview Elements . . . 72
6.4.3 Applying the Node Style . . . 75
6.5 Importing and Managing Node Styles . . . 76
6.5.1 Importing a Node Style . . . 77
6.5.2 Deleting a Node Style . . . 77
6.5.3 Renaming a Node Style . . . 77
6.5.4 Adding a Node Style Folder . . . 77
6.5.5 Deleting a Folder . . . 78
6.6 Associating Custom Graphics with Classes and Elements . . . 78
6.6.1 Viewing a Drawing or Graphic . . . 78
6.6.2 Creating and Associating a Custom Drawing or Graphic . . . 79
6.6.3 Associating a Node Style or Drawing Template . . . 79
6.7 Stacked Container Example: Toggling Graphics Based on Condition State . . . 79

7 Using Drawing Templates 83

7.1 Applying Templates to Drawings . . . 83
7.1.1 Applying a Template to Elements of a Class . . . 84
7.1.2 Applying a Template to an Element . . . 84
7.1.3 Using a Template to Create a Custom Drawing . . . 85
7.2 Unapplying Templates from Drawings . . . 85
7.2.1 Unapplying a Template for an Element . . . 85
7.2.2 Unapplying a Template for a Class . . . 85
7.3 Creating Templates . . . 86
7.3.1 Creating a New Template . . . 86
7.3.2 Creating a Template from a Custom Drawing . . . 87
7.3.3 Editing Templates . . . 88
7.4 Importing and Managing Templates . . . 89
7.4.1 Importing a Template . . . 89
7.4.2 Organizing Templates Using Folders . . . 89
7.4.3 Deleting a Template . . . 89
7.4.4 Renaming a Template or Folder . . . 90

8 Layout View Examples 91

8.1 Geographical View Example . . . 91
8.2 Tabular Data View . . . 98
8.3 Business Process View Example . . . 101

A Rendering Rule Precedence for Node Styles and Graphics 105

A.1 Element Rendering Look Up Strategy . . . 106
A.2 Creating a Test Model . . . 108
A.2.1 Creating a New Metamodel Class and Elements . . . 108
A.2.2 Creating a Manually Bound Element in the Drawing . . . 111
A.3 Viewing the Layout in the Dashboard . . . 112
A.4 Creating Node Styles for Testing . . . 112
A.5 Applying the Ambient Node Style to the Drawing . . . 113
A.6 Changing the Node Style through Binding . . . 114
A.7 Applying a Node Style to a Class . . . 116
A.8 Applying a Graphic to a Class . . . 117
A.8.1 Changing the Class Node Style to a Custom Graphic . . . 118
A.8.2 Applying a Node Style at the Element Instance Level . . . 119
A.8.3 Applying a Graphic at the Element Instance Level . . . 121
A.8.4 Overriding a Single Element Node Style . . . 122
A.8.5 Clearing Previously Set Node Styles and Graphics . . . 124

B Bind Language API Reference 131

B.1 Bind Task Examples . . . 132
B.1.1 Adding Tooltips or Pop-Ups to Portlets in the Dashboard . . . 132
B.1.2 Tying an Element to a Drawing Shape . . . 138
B.1.3 Changing Shapes or Showing Objects Based on Element Condition . . . 139
B.1.4 Changing Shapes Based on Element Property . . . 139
B.1.5 Displaying Element Name or Element Property . . . 140
B.1.6 Displaying Element Children . . . 141
B.1.7 Formatting Numbers, Dates, and Time . . . 142
B.2 Bind Templates . . . 143
B.2.1 Bind Attributes . . . 144
B.2.2 Template Subcommands . . . 145
B.2.3 Layout Bind Commands . . . 145
B.2.4 Layout Target Element Attributes . . . 146
B.2.5 Generated Copy Variables . . . 146
B.3 Layout . . . 146
B.4 Bind Values . . . 149
B.4.1 Bind Value Attributes . . . 150
B.4.2 Using NOC Script in Bind:Value Tag . . . 151
B.5 Resource Generation . . . 152
B.6 Bind Promotion . . . 152
B.7 General Bind Annotations . . . 153
B.8 Script Objects . . . 153

C Editing the Source SVG Code 155

C.1 Editing the SVG Code and Regenerate the Drawing . . . 155
C.2 Using the Opacity Attribute . . . 156
C.3 Using Relative DNames for Node Attribute Matching . . . 156
C.4 Looking at the Source Code with a Toggling Graphic Node Style Example . . . 158
C.5 Looking at the Source Code with a Child Container Example . . . 168

Glossary of Terms **173**

About This Guide

The *Custom Drawing and Layout Guide* provides a visual analysis of critical relationships and conditions across multiple branches of an element hierarchy.

- Chapter 1, “Introduction,” on page 9
- Chapter 2, “Getting Started with the Layout View,” on page 13
- Chapter 3, “Using Basic Drawing Tools and Concepts,” on page 19
- Chapter 4, “Working with Clip Art and Graphics,” on page 33
- Chapter 5, “Adding Elements and Updating Drawing Components Using Element Properties,” on page 39
- Chapter 6, “Working with Node Styles and Custom Graphics,” on page 65
- Chapter 7, “Using Drawing Templates,” on page 83
- Chapter 8, “Layout View Examples,” on page 91
- Appendix A, “Rendering Rule Precedence for Node Styles and Graphics,” on page 105
- Appendix B, “Bind Language API Reference,” on page 131
- Appendix C, “Editing the Source SVG Code,” on page 155
- “Glossary of Terms” on page 173

Audience

This guide is intended for Operations Center system administrators who manage and create custom drawings in the Layout view.

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 *Custom Drawing and Layout 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

In NetIQ documentation, 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

The Operations Center Layout view can provide a visual analysis of critical relationships and conditions across multiple branches of an element hierarchy. The dynamic linking feature binds graphics in a drawing to element conditions and attributes, thus enabling automatic updates of the drawing when elements change. The Layout view also provides a set of drawing tools for adding shapes, colors, text, and many other features found in standard drawing products.

Planning a Layout view custom drawing requires consideration of its purpose and usage. Typically, administrators create and edit Layout view drawings for various types of users. Before creating a drawing, consider the drawing's audience and the information they need. For example, a Layout view for executives is likely to contain a diagram of a service model's health and compliance. In contrast, an operations department is interested in a network diagram identifying critical performance areas.

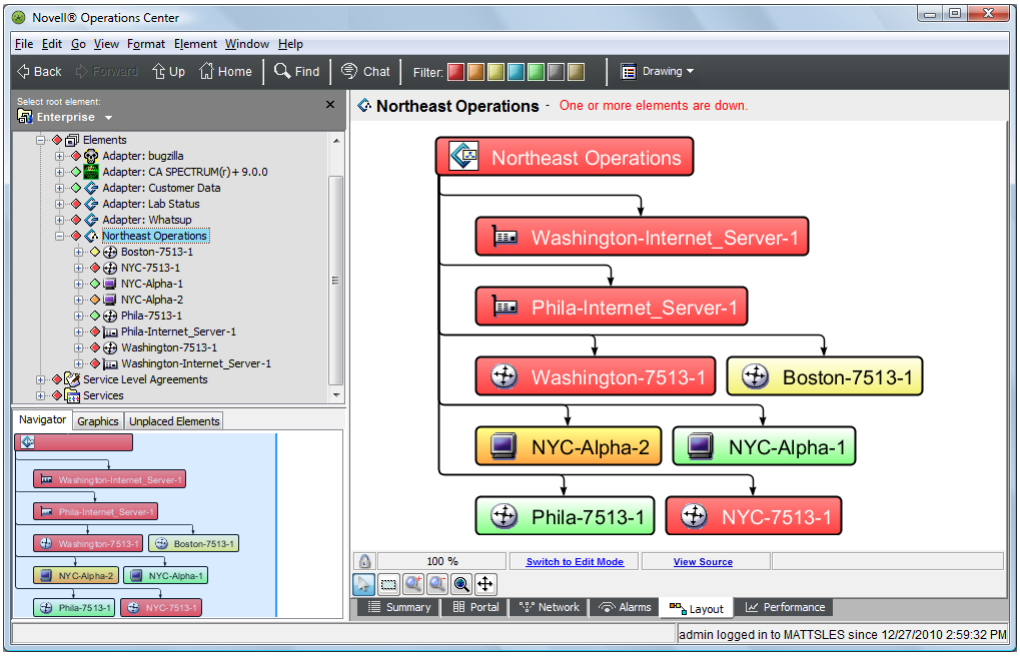
A Layout drawing is built in the Operations Center console by Administrators, but can be published in the Dashboard for your users.

- Section 1.1, "Default Layout View," on page 10
- Section 1.2, "Custom Drawings," on page 10
- Section 1.3, "Where Do I Start?," on page 11

1.1 Default Layout View

By default, the Layout view shows the selected element and its first-level children using a Gradient Bubble node style where fill color corresponds to the element's current condition. Using default condition settings, red means CRITICAL, orange means MAJOR, and so on. Arrows between elements show the relationships between parent and children. This is illustrated in Figure 1-1:

Figure 1-1 *Default Layout View Drawing*



1.2 Custom Drawings

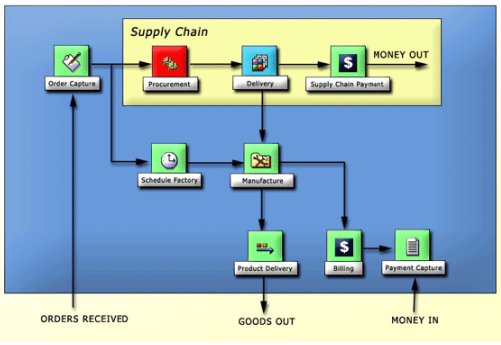
In some cases, you may wish to customize an element's layout by stylizing the node style; or, adding graphics to create a custom drawing -- in order to help users better understand the organization of elements in an infrastructure.

The following are examples of how the default Layout view can be modified to create attractive, useful and informational displays:



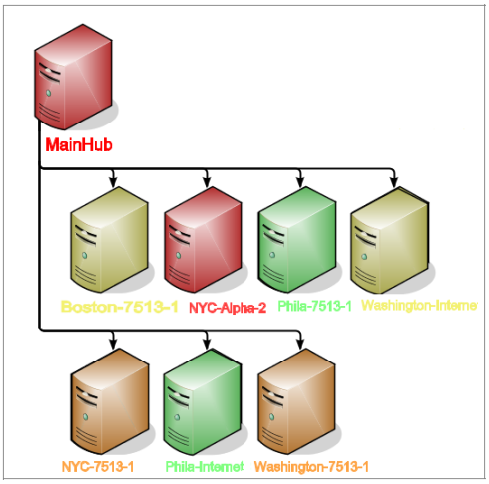
Geographic Maps

An imported map can be overlaid with circles or other shapes to represent the current network performance in each country. Location shapes are linked to element condition color. Or maps can be used in many creative ways such as linking and updating the relevant portion of the map to show system status.



Business Processes

Imported graphics represent various elements in a business process. The shapes beneath the graphics are linked to element condition colors. For example, the *Procurement* element is red, representing the CRITICAL condition. The Layout view drawing tools were used to add text labels, arrows, and basic background shapes using gradient colors.



Custom Node Styles

Customized node styles can enhance the automatic layout and illustrate the status of a network configuration management application. The computer graphic node style uses fill color to show current server condition and identify sites experiencing critical problems. Server names use bound text to also show server condition. The drawing shows mapping and visualization of configuration items that enables peer-to-peer and hierarchical views. It also avoids duplicates and enables matching of configuration items from different sources.The SVG graphics help users visualize critical relationships among elements in different areas of the element hierarchy.

1.3 Where Do I Start?

Read this guide to plan a Layout view and learn to use features that make the Layout drawing dynamic and how to customize the drawing with logos, clipart, and more. Examples are provided in Chapter 8, “Layout View Examples,” on page 91 to help you get started.

Of note:

- New users of the Layout view should first read Chapter 2, “Getting Started with the Layout View,” on page 13 for an introduction to commonly used features.
- Basic editing information, such as how to import clip art or a use the drawing tools, is found in:
  - Chapter 3, “Using Basic Drawing Tools and Concepts,” on page 19
  - Chapter 4, “Working with Clip Art and Graphics,” on page 33
  - Chapter 5, “Adding Elements and Updating Drawing Components Using Element Properties,” on page 39
- Advanced features, such as binding elements to a drawing are explained in Chapter 5, “Adding Elements and Updating Drawing Components Using Element Properties,” on page 39.
- Experienced users wanting to understand how to apply templates and node styles at different levels should read:
  - Chapter 6, “Working with Node Styles and Custom Graphics,” on page 65
  - Chapter 7, “Using Drawing Templates,” on page 83
- For step-by-step instructions in creating example applications using various Layout view features, see Chapter 8, “Layout View Examples,” on page 91.

- For more advanced topics including Bind language reference materials:
  - Appendix A, “Rendering Rule Precedence for Node Styles and Graphics,” on page 105
  - Appendix B, “Bind Language API Reference,” on page 131
  - Appendix C, “Editing the Source SVG Code,” on page 155

2 Getting Started with the Layout View

When a user switches to the Layout view for any element, the view updates to show a drawing if available. If no drawing has been created, the default automatic layout displays.

Typically, system users do not have the authority to modify Layout view drawings. In some cases, the Layout view can be disabled and not be available to users who are not members of the administrators group. For more information, see *Access Control* in the *Operations Center 5.5 Security Management Guide*.

The Layout view is read-only when indicated by a lock icon in the bottom left of the view. In read-only mode, you can navigate the view, zoom into different areas, and drill down to view linked elements.

The following sections cover basic functionality to start building custom drawings for element Layout views:

- Section 2.1, “Navigating Inside the Layout View,” on page 13
- Section 2.2, “Creating and Editing Custom Drawings,” on page 15
- Section 2.3, “Understanding Unplaced Elements,” on page 16
- Section 2.4, “Moving Drawing Elements,” on page 17
- Section 2.5, “Clearing a Drawing or Reverting to a Previous Drawing or Default Automatic Layout,” on page 17
- Section 2.6, “Setting Security on Graphic Elements,” on page 18
- Section 2.7, “Troubleshooting Tips,” on page 18

2.1 Navigating Inside the Layout View

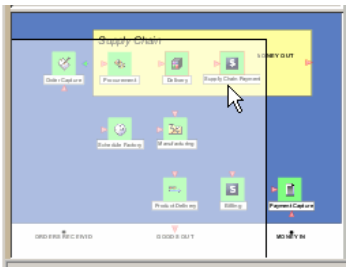
When you first open a Layout view, the drawing is in View mode, which means you can view, but cannot modify the drawing. In View mode, you can:

- **Use the basic navigation tools to move around the Layout view.**

View Menu Options	Button	Shortcut	Description
			Zooms into to a selected area. Click the *Selection Zoom* button, then click & drag inside the Layout view to draw a selection rectangle. All elements within the selection rectangle magnify to fill the viewing area.
Zoom 1:1		1	Resets the drawing to 100%, but does not center the drawing.

View Menu Options	Button	Shortcut	Description
Zoom +		+	Incrementally magnifies the drawing.
Zoom –		–	Incrementally reduces the size of the drawing.
Zoom to...			Selects the size of a drawing by a percentage.
Zoom to World		Home	Zooms to display the entire drawing. In most cases, it centers and resets the drawing to 100%. If the drawing is originally larger than the Layout view, it reduces the drawing until it fits inside the view.
			Use Pan to move the drawing position inside the view. Click *Pan* to change to pan mode, then click and drag the drawing to move it to the desired position.

- **Use the *Navigator* pane to navigate large or complex drawings**. Found in the lower left panel, it presents a "bird's eye" view of the drawing that is helpful when zoomed in on a large or complex drawing. Click and drag the transparent overlay pane to display different portions of a drawing in the Layout view.



To show/hide the navigator, select or deselect *Show Navigator* in the *View* menu.

- **Drill down to element hierarchies.** Move from the current Layout drawing to other drawings by clicking elements in the current drawing. Drawing objects can be linked to specific elements in the hierarchy. The system administrator who created a custom drawing for an element should explain the general purpose of the drawing, and identify the linked elements.

  If there is no drawing associated with the element, the default layout is shown. If the view is empty, a new drawing was created without any graphics.

- **Use the *Unplaced Elements* pane to identify and work with elements that have not been placed in the drawing**. See Section 2.3, "Understanding Unplaced Elements," on page 16).

2.2 Creating and Editing Custom Drawings

When considering changing the default Layout drawing, consider the changes needed to make it useful for purposes of analysis and decision making. Consider the people using the drawing and the information that is important and should surface immediately. Design a layout of icons and graphics that best convey element, service, or process information.

Some important general facts about working with the Layout view:

- Each Layout view drawing is associated with the element selected in the *Explorer* pane.
- When a Layout view is first displayed, it is read only. Making any changes to a Layout drawing creates a custom drawing. To modify a drawing requires switching to Edit mode, which locks the element and prevent other users from editing the same Layout view.
- A Layout drawing can only be modified by one user at a time. When a Layout drawing is being edited, it is locked until the user saves or cancels the drawing.
- Changes in the Layout view are not automatically saved. Use *Save Drawing* or `Ctrl+S` to save your work.

To create a drawing:

1. In the *Explorer* pane, select the element and open the *Layout* tab.
2. Right-click the *Layout* view background and select *New Drawing*, or use keyboard shortcut *(Ctrl + N)*. All element nodes and any existing drawing objects are cleared from the view.

   Child elements are listed in the *Unplaced Elements* pane. For more information about Unplaced Elements, see Section 2.3, "Understanding Unplaced Elements," on page 16.
3. To build a custom drawing do any of the following:
   - Select elements that are important to visualize on screen in a single drawing.

     Drag and drop existing child elements from the *Unplaced Elements* pane.

     For more information about unplaced elements, see Section 2.3, "Understanding Unplaced Elements," on page 16.

     Drag and drop elements from any branch of the elements hierarchy to a Layout drawing. For example, mix elements from an adapter hierarchy and the *Service Models* hierarchy in a single drawing.
   - Click and drag elements inside the drawing to arrange them in an optimal layout. Decide the best way to graphically represent these elements using node styles.

     Select a node style from the node styles that are available. You can select a simple graphic that displays only the element name or a descriptive style that displays the number of children and the date/time the element was last updated.
   - Add color, text, shapes, and clipart using the drawing tools. Use arrows to show relationships.

     For more information on drawing tools and shapes, see Chapter 3, "Using Basic Drawing Tools and Concepts," on page 19.
   - Dynamically update status and information by binding element conditions or attributes to shapes or text in the drawing.

     The binding feature automatically updates graphic colors when element conditions change. Similarly, if the element name or other linked attribute changes, bound text in a drawing also updates.

     For more information about binding elements to a drawing, see Chapter 5, "Adding Elements and Updating Drawing Components Using Element Properties," on page 39

- Import maps, logos, and other images to enhance the drawing.

  For more information on applying nodestyles and using custom graphics, see Chapter 6, "Working with Node Styles and Custom Graphics," on page 65.
- Apply a drawing template to apply a standard set of drawing objects or attributes.

  For more information about applying drawing templates, see Section 7.1, "Applying Templates to Drawings," on page 83.

To edit a Layout view drawing:

1. In the *Explorer* pane, select the desired element and open the *Layout* tab.
2. By default, the Layout view is read-only. Click *Switch to Edit Mode* in the Layout view status bar.

   In the *Explorer* pane, the word (Locked) displays next to the selected element to show the Layout view is locked for this element. The view updates to display the drawing tools toolbar.

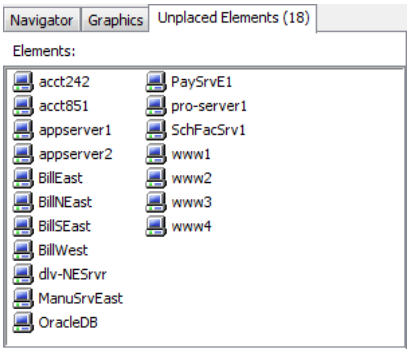
   TIP: You can also switch to Edit mode by pressing Ctrl+D or by clicking *Edit* > *Edit Drawing*.
3. Make changes to the drawing.
4. Save the drawing changes by pressing Ctrl+S or by right-clicking the background and selecting *Save Drawing*.

2.3 Understanding Unplaced Elements

The *Unplaced Elements* pane, found in the lower left panel, presents a listing of any elements that are not included in a custom drawing. When the Layout view is in *Edit* mode, these elements can be dragged and dropped into the drawing.

When you create a new drawing using the `New Drawing` option (or use the `Clear Drawing` option to start over), the Layout view is cleared and all children—native and linked elements—as well as any drawing objects are removed from the drawing and placed in the *Unplaced Elements* pane. They remain there until you drag and drop them into the drawing. In some cases, you might have some elements to remain "hidden" from the Layout drawing. For these elements, you would leave them in *Unplaced Elements*.

Figure 2-1 *Unplaced Elements Pane*



By default, the *Unplaced Elements* pane is available only for administrators—therefore, any unplaced elements are unknown by users who do not have permissions to view it. The *Unplaced Elements* pane can be exposed to users by setting the `SVG.ShowUnplaced` property to true in the `launch.jnlp` file.

To show the *Unplaced Elements* pane for all users:

1. Modify the */OperationsCenter_install_path*/html/client/template/launch.jnlp file to set the SVG.ShowUnplaced property to true:

   ```
   <property name="SVG.ShowUnplaced" 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*.

2.4 Moving Drawing Elements

A common action is to move elements in a drawing, for the purposes of reorganization or to make the content easier to read.

To move a drawing element:

1. Click *Switch to Edit Mode* in the Layout view status bar.

   In the *Explorer* pane, the word (Locked) displays next to the selected element. No one else can modify the Layout view for this element.

   The view updates to display the drawing tools toolbar.

2. Drag the elements to the appropriate location.
3. Save the changes to the drawing by pressing Ctrl+S or by right-clicking the background and selecting *Save Drawing*.

2.5 Clearing a Drawing or Reverting to a Previous Drawing or Default Automatic Layout

At any time, you can abandon unsaved changes to revert to the last saved drawing or the original default drawing.

Table 2-1 *Undoing a custom drawing*

To...	Do the following:
Revert to last saved drawing	Click *Edit* > *Revert to Last Saved*.
Remove a drawing and revert to the original automatic layout	Click *Edit* > *Revert to Automatic Layout*. This reverts to the default Gradient Bubble node style and a traditional tree layout style. Everything in the drawing is removed, except for the element associated with the drawing and its first-level children. The drawing must be saved at least once for the *Revert to Automatic Layout* option to be available.
Clear everything and start from scratch	Right-click the background of the drawing, then select *Clear Drawing*. Clearing a drawing removes everything from the Layout view, leaving it in Edit mode. However, nothing changes in the *Explorer* pane element hierarchy. The elements there remain intact.

2.6 Setting Security on Graphic Elements

Security permissions can be set for graphic elements and drawing objects, such as templates, node styles, and clip art. Follow the same rules as those used for establishing permissions for other elements, as described in the *Operations Center 5.5 Security Management Guide*. These permissions apply when editing Layout view drawings. However, the permissions are ignored when a user views a saved drawing.

For example, a user does not need *View* privileges for a clipart image in order to see the clipart in a Layout drawing. However, in *Edit* mode, a user needs *View* permissions to add the clipart to a new drawing or to an existing drawing.

2.7 Troubleshooting Tips

The following sections provide help for the following:

- Section 2.7.1, "Empty Drawings," on page 18
- Section 2.7.2, "Identifying if a Shape is Associated to an Element," on page 18

2.7.1 Empty Drawings

If the Layout view is empty with no elements or custom drawing, the following situations may be occurring:

- A new drawing was created and saved without any graphics. For instructions to re-create the default layout, see Section 2.5, "Clearing a Drawing or Reverting to a Previous Drawing or Default Automatic Layout," on page 17
- The drawing channel is set to *Element Graphic* when none has been created. For more information, see Section 6.6, "Associating Custom Graphics with Classes and Elements," on page 78.

2.7.2 Identifying if a Shape is Associated to an Element

By default, nothing in the drawing indicates a shape is bound to an element. Sometimes it is useful to identify the drawing shapes that are bound to elements.

To toggle between displaying and hiding the bind symbol, which is a lightning bolt, click *View* > *Show Bindings*:



The lightning bolt symbol indicates the circle shape is bound to an element.

Elements for which a user does not have View permission do not display in the Layout view drawing. Thus, if an element is bound to an icon in the Layout view, the icon is not visible to users who do not have View permission for the element.

3 Using Basic Drawing Tools and Concepts

This section covers the basic editing features and the primary drawing tools that are available to customize your Layout view drawing.

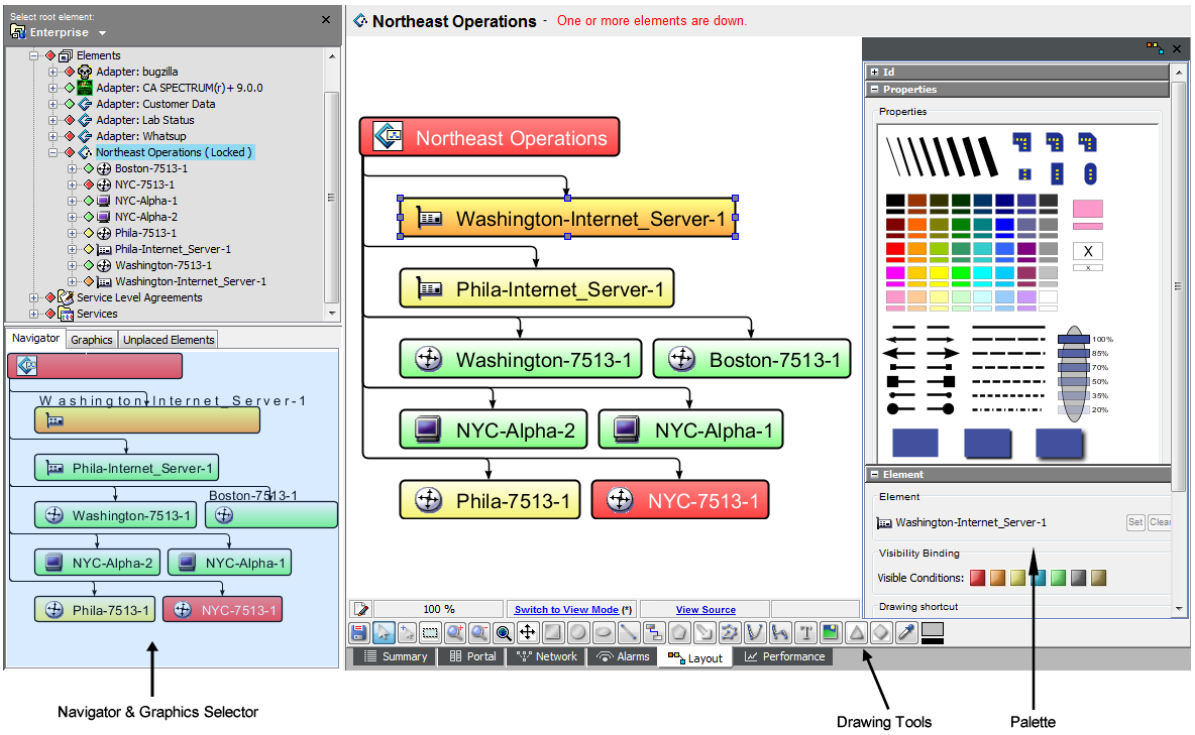
The following sections cover basic editing and drawing features for creating a custom drawing:

- Section 3.1, “Understanding the Layout View Panels and Drawing Palette,” on page 20
- Section 3.2, “Selecting Drawing Objects,” on page 21
- Section 3.3, “Adding Shapes and Lines,” on page 22
- Section 3.4, “Changing Fill Colors,” on page 24
- Section 3.5, “Changing the Background Color,” on page 24
- Section 3.6, “Using Undo,” on page 25
- Section 3.7, “Manipulating Shapes and Objects,” on page 25
- Section 3.8, “Using the Grid to Align Objects,” on page 26
- Section 3.9, “Grouping and Ungrouping Objects,” on page 27
- Section 3.10, “Using Containers to Control Objects,” on page 28

3.1 Understanding the Layout View Panels and Drawing Palette

In Edit mode, the Layout view displays drawing tools and drawing properties. Figure 3-1 shows the main components of the Layout view that are referenced throughout this section:

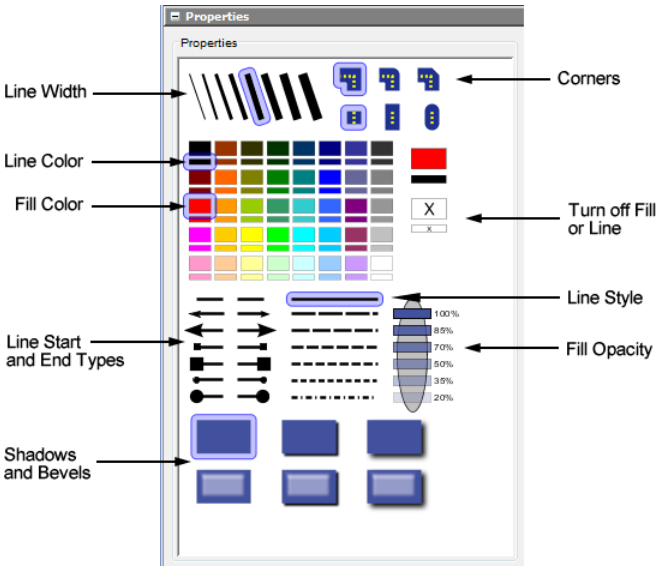
Figure 3-1 *Layout View's Drawing Tools and Properties*



Some tips on using the layout view:

- Use the Navigator to focus on a specific area of the drawing.
- Use the Drawing Tools to add shapes, lines, and text to the drawing. Also import graphics and zoom in or out of the drawing.
- The Palette displays various properties panes with options for drawing objects, such as colors, line widths, and auto update with element name or condition color. The property panes available depend on the type of object that is selected in the drawing. To facilitate viewing, collapse panes that are not of interest and expand the panes you are using most.

The Palette location in the browser can vary, depending on your selection.



The Palette can display as a floating dialog box, a tab, or a docked element. The Palette location depends on your selection. Click *View* > *Palette Location*, then select a location:

Controller: Adds *Palette* as a tab where the *Navigator* and *Graphics* tabs display.

Floating: The Palette can be dragged as a single component to any location on the screen. You can move the Palette at any time. Click (*Attach/Pushpin*) to dock the Palette to the right side of the screen.

Docked: Anchors the Palette to the right side of the screen. Click (*Detach*) to change to a floating Palette.

Hidden: Removes the Palette from the Layout view. To display the Palette, select one of the other menu options.

3.2 Selecting Drawing Objects

There are two selection modes in the Layout view:

- **Normal:** Grouped objects respect the layering of a group of objects. A group has to be ungrouped in order for a single object within the group to be selected and altered.
- **Flattened:** The layers created by object groupings are "flattened" and single objects in the group can easily be selected and altered without ungrouping all.

To select objects:

1. To select complex items (grouped items) as a single item, click (*Normal*).
2. To select and move a single item that is part of a group, click (*Flattened*).

3.3 Adding Shapes and Lines

The Layout view provides a full set of drawing tools. These tools function much the same way as features found in other drawing products. Add text, colored shapes and imported clipart to provide additional information and enhance the appearance of a drawing.

In Edit mode, drawing tools display along the bottom of the Layout view:

Figure 3-2 *Drawing Toolbar*



Each tool button has a roll-over tooltip. Place the mouse over a button and a tooltip displays the button name.

A few buttons on the toolbar enable freehand drawing of lines or shapes, depending on whether you select a fill color.

Table 3-1

Adding Text	
(*Add Text*)	Click on the drawing and enter your text. Use options in the *Properties* and *Text* panes to change color or font. To automatically update the text with element name or attribute, see Section 5.2.1, “Creating an Element Name Heading,” on page 41.
Adding Shapes	
(*Add Rectangle*) (*Add Circle*) (*Add Ellipse*) (*Add Polygon*) (*Add Triangle*) (*Add Diamond*)	Click and drag the outline of the shape anyplace in the drawing. Later, click and drag any of the shape’s endpoints to modify the size or dimensions of the shape.Change almost any feature of a shape using the Properties palette: the line and fill colors, line shapes and width, shape shadows, and so on. To automatically update the fill color based on element condition color, see Section 5.2.3, “Dynamically Updating a Shape’s Color based on Element Condition,” on page 44.
Drawing Lines and Arrows	
(*Add Line*)	Click and drag to draw the line.Change line color and thickness using the Properties palette. To modify the line angle, simply click and drag one of the end points.To snap the line in 15 degree increments, press *Shift* while drawing the line end points.Move the line by clicking and dragging its midpoint.

(*Add Connection*)	Connect two drawing objects.Position the mouse over an element to display blue “x” s at the connection points. Click one blue “x” to start the line, and drag the line over to click the blue “x” on a different object.
(*Add Block Arrow*)	Click and drag to draw the arrow.

Drawing Freehand Shapes and Lines

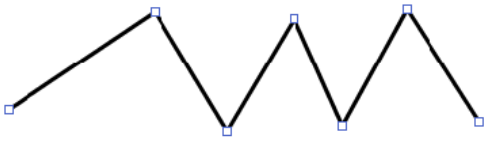
(*Add Polyline*)

(*Add Simple Curve*)

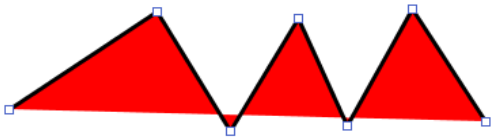
(*Add Complex Curve*)

1. Click to start the line, then drag to draw the first line segment. Click once to mark the end of the line segment.
2. Drag to draw the next segment. Click once to mark the end the line segment.

   TIP: To snap a polyline in 15 degree increments to create curves, press Shift while drawing.
3. When finished drawing all segments, double-click to finish the line.



4. To fill the shape drawn by the line, select a fill color.



3.4 Changing Fill Colors

There are several ways to change the line or fill color of a shape. Use the color palette, see Section 3.1, "Understanding the Layout View Panels and Drawing Palette," on page 20, or click (*Color*) on the Layout drawing toolbar.

It is possible to create custom colors and apply them to new or existing drawing components.

To apply colors:

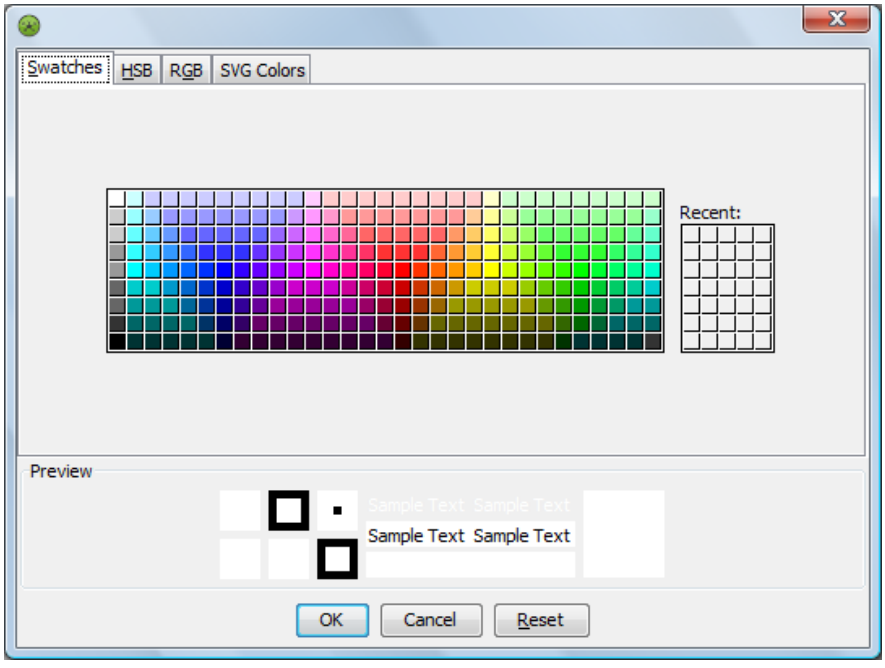
1. To change colors, do one of the following:
   - **1a** In the toolbar, click (*Color*) to open the Change Color dialog box.
     Select a new color using the *Swatches*, *HSB*, and *RGB* tabs.
   - **1b** In the toolbar, click (*Color Picker*), then click any other color in the drawing.
2. Drag the *Color* button to an existing shape.The shape color updates.

3.5 Changing the Background Color

A quick way to enhance a drawing is to add background color.

To specify the background color for the drawing:

1. Click *Edit* > *Set Background Color* to open the color selector:



2. To change the background color to create a dramatic effect, select a color, then click *OK*.
   The background changes to the selected color.

3.6 Using Undo

If you make a mistake while drawing, use the *Undo* feature. It works for most actions that are performed using the drawing tools and drawing options, such as *Group*, *Cut*, *Paste*, and so on.

To use the Undo and Redo features:

To do this	Press	Or Click
Undo	Ctrl+Z	*Edit* > *Undo*
Redo	Ctrl+Y	*Edit* > *Redo*

3.7 Manipulating Shapes and Objects

While working in the Layout view, you can use standard element options for the element currently selected in the *Explorer* pane. These options include viewing element properties, and viewing Impacted and Root Cause.

In Edit mode, right-clicking the background of the drawing displays drawing options in addition to the standard element options, which display under the *Element* menu option. *Edit* and *Format* menu options are available for editing drawing shapes, which are added using the drawing tools, and for graphics that represent elements in the *Elements* hierarchy (displayed in the *Explorer* pane).

- To select a drawing shape or element for editing, click the shape in the Layout view.

  The selection "handles" display around its perimeter.
- To select multiple elements, press and hold the `Shift` key, then select multiple elements.
- To change the color of shapes or lines in a drawing, use the *Properties* pane. To view a tooltip for any item on this pane, position the mouse over the item.

  Select a drawing component, such as a shape or line. The *Properties* pane updates to display the current properties for the drawing object.

  Keep in mind that the properties apply only to the Layout view items that can use them. For example, if you use the Palette to set a group's line, color, and fill properties, the drawing attempts to pass these property values to the children nested inside that group. This inheritance works only if a different property value has not been specified for the nested children. Specified property values for children take precedence over their parents' properties.
- To edit shapes, right-click a drawing shape, or click the *Edit* menu, then select an option:

  Copy: Copies the shape and all its properties to the clipboard.

  Paste: Pastes the shape in the drawing.

  Cut: Removes the shape and places it on the clipboard.

  Delete: Permanently removes the shape from the drawing.

- To edit editing shapes or elements that were manually added to the drawing (not elements that are part of the Automatic layout), right-click a shape or element, or click the *Format* menu and select an option:

  Align: Arranges multiple elements using the selected starting point, such as *Right* or *Top*. Available only when multiple elements are selected.

  Set to the same: Changes the elements to the same width, height, or size. Available only when multiple elements are selected.

  Spacing: Place the elements in the horizontal or vertical center of the view. Available only when multiple elements are selected.

  Order: When elements overlap, specify the order for the current shape in relation to the other element: on *Top* or *Bottom*, or one layer *Down* or *Up*.

  Rotate: Rotates the elements using the specified degree.

  Flip: Turns the elements 180 degrees along its vertical or horizontal axis.

3.8 Using the Grid to Align Objects

Use the grid feature to align shapes and other drawing components. Aligning components with the grid is optional.

Do the following tasks, as necessary:

1. To display the grid, click *View* > *Grid* > *Show Grid*.

   The grid displays in the Layout view.
2. To align elements with the closest grid line, click *View* > *Grid* > *Enable Align with Grid*.

   All new shapes and lines align with the grid.

   Existing elements do not align with the grid, unless you try to move them.
3. To hide the grid, click *View* > *Grid* > *Show Grid*.

   The grid disappears.
4. To change the grid spacing, do the following:
   - **4a** Click *View* > *Grid* > *Grid Settings* to open the Modify Grid Settings dialog box.
   - **4b** Edit the zoom level and intervals.
   - **4c** Click *OK*.

     The grid updates.

3.9 Grouping and Ungrouping Objects

Grouping drawing components combines multiple components into a single component. Grouping is useful when you should rearrange or manage many components on the screen. Instead of having to drag multiple components, you can click once and select multiple items and move them at one time.

Ungrouping is particularly important when editing imported images, such as maps. In order to edit one area of the image, you might need to ungroup the image first.

The *Group* options described in the following subsections are available under the *Edit* menu, and also by right-clicking a drawing components.

To group drawing components:

1. Use the selection methods listed above to select components. For example, this might include elements, shapes, and associated text labels.

   Elements generated by an Automatic layout cannot be grouped with other items.

2. Right-click the group, then select *Group*.

   The selected components become a single component.

To ungroup drawing components:

1. Select the group.
2. Right-click the group, then select *Ungroup*.

3.9.1 Working with Grouped Components

Actions performed on a group affect the group as a single entity. They do not affect individual components within the group. For example, if you drag and drop an element from the *Explorer* pane to a group, it binds to the entire group, not to any single drawing component in the group.

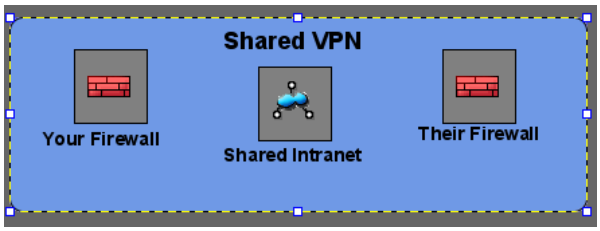
However, it is possible to change individual components while keeping the group intact. These changes include modifying shapes, colors, text, and binding elements to shapes or lines.

To change individual objects within a group:

1. Select the group.

   A dashed box outlines the group.

   In the following, the group includes the rectangle and all items within it:



   Another hint for identifying a group is in the Palette ID label. The node type displays in the label, and in the case of a group, it displays a `<G> ID` label instead of a `<PATH> ID`.

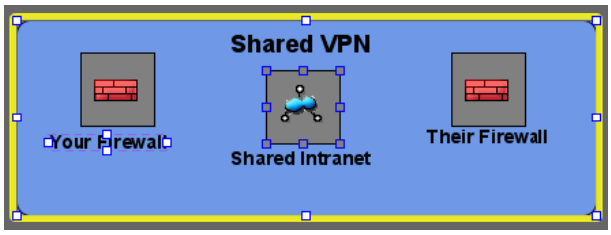
2. Right-click, then select *Change Group*.

   A yellow band outlines the group (see the figure in Step 3).

3 Select individual components within the group and make changes.

You can edit only the components that have white selection "handles," which were added using the drawing tools. Components that were linked or copied to the drawing from the *Element* hierarchy have grey selection "handles" and cannot be edited.

In the following, the rectangle and text can be edited:



The hierarchy elements, such as Your Firewall, cannot be edited.

4 When finished, right-click the container, then select *Save Group*.

3.10 Using Containers to Control Objects

Container can help you group objects with more options and control over objects than the standard grouping feature.

There are three types of containers that can be used with the Layout view:

- **Simple Container:** groups one or more objects in an outer container.
- **Stack Layout Container:** layers multiple objects. Stack Layout Container are useful to create node styles that show a different graphic based on the element state.

  For information about using stack containers to create a node style, see Section C.4, "Looking at the Source Code with a Toggling Graphic Node Style Example," on page 158.
- **Children Container:** shows one or more levels of child elements.

  For more information about using children containers, see Section 5.4, "Displaying an Element's Children using Child Containers," on page 48.

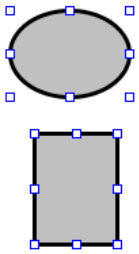
This section covers how to group multiple objects with a simple container:

- Section 3.10.1, "Grouping Components with a Container," on page 29
- Section 3.10.2, "Adjusting the Wrap Settings," on page 29
- Section 3.10.3, "Adjusting Container Layout," on page 30

3.10.1 Grouping Components with a Container

To group drawing components with a container:

1 Use the selection tool to select the desired components.

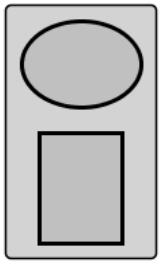


For example, this might include elements, shapes, and associated text labels.

Elements generated by an Automatic layout cannot be grouped with other items.

2 Right-click anywhere on a selected component and select *Customize* > *Container* > *Wrap in Container*.

The selected components are wrapped in an outer container.

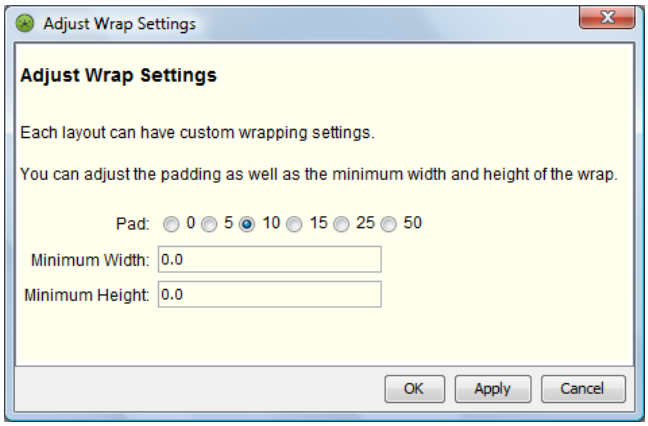


3.10.2 Adjusting the Wrap Settings

While outer containers can be modified just like any other rectangle or drawing shape, there are some options that are unique to containers.

To adjust the outer container and wrapping attributes:

1 To adjust the wrap settings, right-click the container and select *Customize* > *Container* > *Wrapping* > *Adjust Wrap Settings*. The Adjust Wrap Settings dialog box opens.



Do any of the following:

- Select any of the *Pad* radio buttons to adjust the spacing between the components.
- Specify the minimum width of the outer container in the *Minimum Width* field.
- Specify the minimum height of the outer container in the *Minimum Height* field.

2 To hide or show the outer container do one of the following:

- To hide the outer container, right-click the container and select *Customize* > *Container* > *Hide Wrapping Rectangle*.

  The outer rectangle is invisible to the view.
- To restore the outer container, right-click the container and select *Customize* > *Container* > *Show Wrapping Rectangle*.

3.10.3 Adjusting Container Layout

There are many types of layouts available for a container that set how components are displayed in the container. Grid and Flow layouts have additional settings for configuration.

Table 3-2 *Container Layout Types*

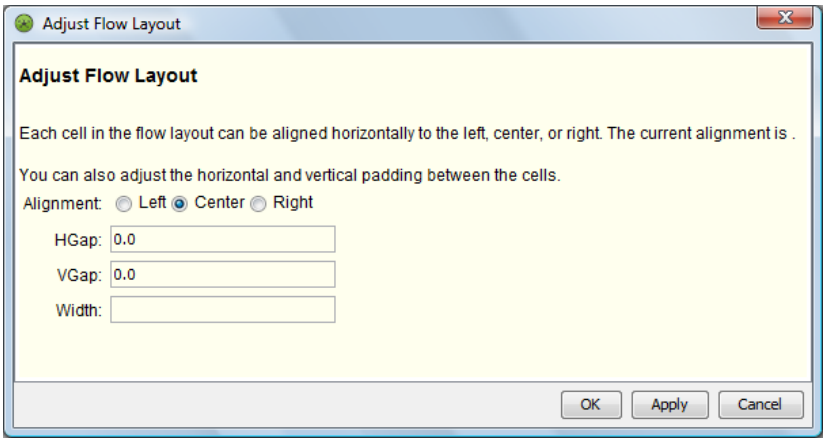
Layout Type	Use to display objects and elements...
Cluster	In a circular grouping of the objects, packing them together, with the largest ones in the center.
Cluster Box	In a rectangular grouping of the objects, packing them together, with the largest ones in the center.
Flow	In a single column, based on the front to back arrangement of the objects and elements. Items in the front display at the top.
Grid	Organized in rows and columns. Configure up to 6 columns. Elements automatically wrap to form as many rows as needed.
Horizontal	In a single horizontal row, based on the front to back arrangement of objects and elements. Items in the back display (left), followed by items further to the front.

Layout Type	Use to display objects and elements...
HTML	Similar to how you would use html to format text content. The components flow, usually from left to right, as they are added to a container, but control strings like "br", "tab" and "p" can be used to better adjust the location.
Stack	One at a time based on conditional settings that determine how they are toggled in and out. For an example, see Section 6.7, “Stacked Container Example: Toggling Graphics Based on Condition State,” on page 79.
Wrap	in a rectangular shape. Generally used to surround a group of child objects with a rectangle after the child objects have been laid out.

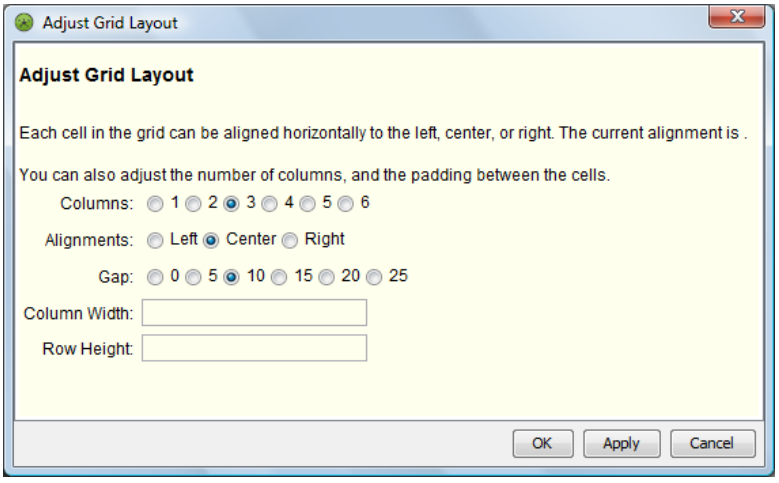
To adjust the outer container and wrapping attributes:

1. In the drawing, right-click the container, then select *Customize* > *Container* > *Layout* > *Set Layout*. The *Set Layout* dialog box opens.
2. Select the layout type in the drop-down list, and click *OK*.

   The view updates to the new layout type.
3. To adjust settings, right-click the container and do one of the following:
   - **Flow Layout:** select *Customize* > *Child Container* > *Layout* > *Adjust Flow Layout*.



   - **Grid Layout:** select *Customize* > *Container* > *Layout* > *Adjust Grid Layout*.



4 Working with Clip Art and Graphics

Graphics can be imported to enhance a drawing. For example, add company logos or custom graphics that represent an organization. These graphics can be resized and moved after they are in the Layout view.

- Section 4.1, “Importing Graphics,” on page 34
- Section 4.2, “Using Graphics from the Clip Art Library,” on page 35
- Section 4.3, “Managing Clipart Files,” on page 37
- Section 4.4, “Creating Graphics for the Clipart Library,” on page 38

4.1 Importing Graphics

Import graphics can be in the following formats: `.png`, `.gif`, `.bmp`, or `.jpg`.

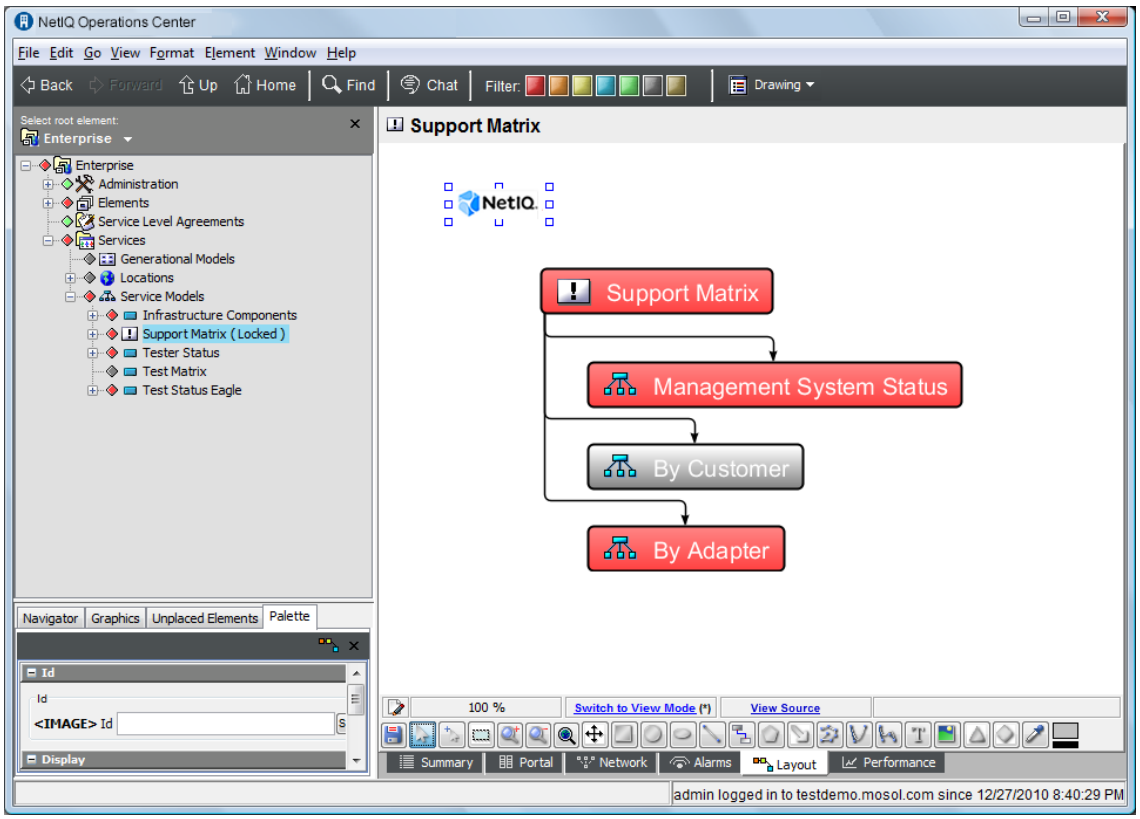
To import a graphic:

1. In the toolbar, click  (*Image*).
2. In the Layout view, click the background to place the image in its native size, or else draw an outline to scale the image to fit within the outlined rectangle.

   The Open dialog box opens.
3. Navigate to and select the image file.

   The image displays in the Layout view.

   In the following, the NetIQ logo is imported as a `.gif` file:



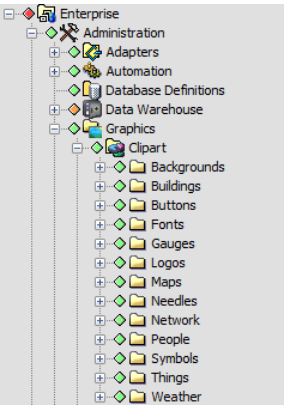
4. Move or resize the image as you would standard shapes that are drawn using the toolbar.

4.2 Using Graphics from the Clip Art Library

Sample clipart ships with Operations Center and is organized into folders representing different content categories. Use clip art for backgrounds or images within the Layout view. You can also import SVG clipart from other sources.

Clipart displays under *Administration* > *Graphics* > *Clipart*. You should import and organize clipart using this hierarchy:

Figure 4-1 *Clipart Hierarchy*



The *Graphics* tab (at the bottom of the Explorer pane) allows you access to the same clipart library under *Administration* > *Graphics* and is used to add clipart to the Layout drawing.

Drag and drop clipart onto the Layout view to use as a background or an additional image. In this way, clipart images are added as a linked objects from the Graphics Library. However, they can also be added as embedded directly into the Layout drawing.

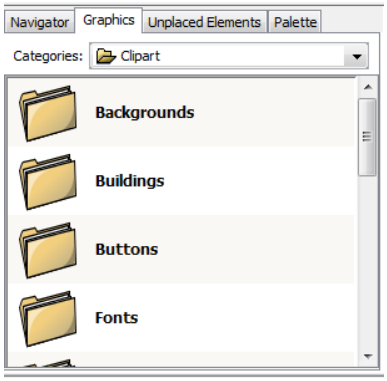
- Section 4.2.1, "Linking Clipart from the Graphics Library," on page 35
- Section 4.2.2, "Embedding Clipart Images," on page 37

4.2.1 Linking Clipart from the Graphics Library

Generally when adding clipart to a drawing, you are simply linking an object from the Graphics Library. This can be accomplished from the *Administration* > *Graphics* > *Clipart* hierarchy or from the *Graphics* tab at the bottom of the Explorer pane.

To add clipart to a drawing:

1. Click the *Graphics* tab.
2. Select *Clipart* from the *Categories* drop-down list to display the clipart folders:



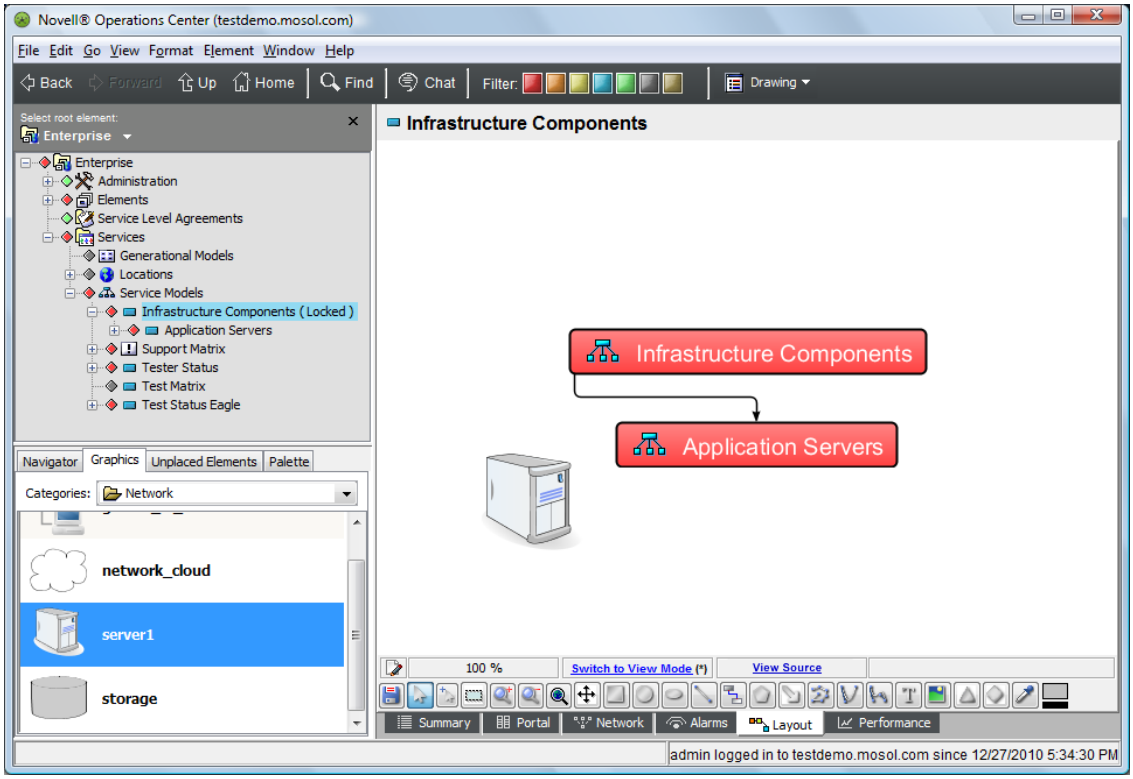
3 Double-click a folder.

The images in the folder display. If necessary, scroll down to view additional images.

4 To change the size of the icons displayed in the *Graphics* tab, right-click anyplace in the *Graphics* tab, then select the icon size (small, medium, and so on) from the pop-up menu.

5 Select a clipart image.

6 Drag and drop the image to the Layout view.



The server1 clipart graphic is added to the drawing.

Alternatively, you can drag and drop clipart from the *Administration* > *Graphics* > *Clipart* folder from the *Explorer* pane to the Layout view.

The drag and drop action automatically changes the Layout view from View to Edit mode.

7 If necessary, click a "handle" around the clipart and drag to enlarge or shrink the image.

You can also drag the image to reposition it.

8 If you are placing the clipart as background, right-click the image, then select *Order* > *Send to Bottom*.

The image displays beneath other drawing components in the Layout view.

4.2.2 Embedding Clipart Images

Embedding an image from the Graphics Library is only accomplished via the *Administration* > *Graphics* > *Clipart* hierarchy. An embedded image is saved as an object in the Layout drawing and is no longer linked from the Graphics Library.

To embed clipart into a drawing:

1 From the Explorer pane, click to expand *Administration* > *Graphics* > *Clipart*.

2 While holding the `Ctrl` key, drag and drop the clipart object into the Layout view.

The drag and drop action automatically changes the Layout view from View to Edit mode. The clipart image is added as an embedded image.

3 If necessary, click a "handle" around the clipart and drag to enlarge or shrink the image.

You can also drag the image to reposition it.

4 If you are placing the clipart as background, right-click the image, then select *Order* > *Send to Bottom*.

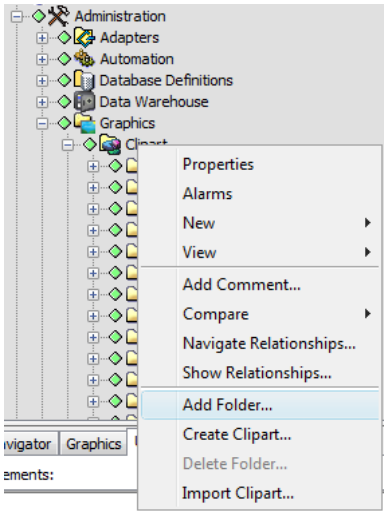
The image displays beneath other drawing components in the Layout view.

4.3 Managing Clipart Files

Clipart is stored under the *Administration* > *Graphics*. Any SVG graphical files can be imported into the clip art library. You can also create folders to organize your clip art. Files can be deleted or renamed using right-click menu options.

To add a clipart folder:

1 In the *Explorer* pane, under *Administration* > *Graphics*,



2 To create a new folder, right-click *Clipart* (or any folder beneath it), then select *Add Folder*.

3 To import a graphic to the clip art library, do the following:

3a Right-click the *Clipart* folder (or any folder beneath it), then select *Import Clipart* to open the Import Clipart dialog box.

3b Navigate to and select the SVG graphic file, then click *Open*. When prompted, specify a name for the new clipart image.

4.4 Creating Graphics for the Clipart Library

You can create clipart using the Layout view drawing tools.

To create clipart:

1 In the *Explorer* pane, right-click *Clipart* (or any folder beneath it), then select *Create Clipart*.

2 Specify a name for the new graphic, then click *OK*.

The name of the new graphic displays in the *Explorer* pane.

3 Click the *Layout View* tab.

The Layout view updates to display an icon representing the clipart element.

4 Verify that the Editing Channel is set to Drawing.

5 Press Ctrl+D to enter Edit mode.

6 Delete the icon representing the clipart element.

7 Create the content using the drawing tools.

8 Save the drawing by pressing Ctrl+S.

5 Adding Elements and Updating Drawing Components Using Element Properties

Although a Layout drawing is associated with a specific element, it is possible to add elements from other element hierarchies to the drawing. Elements can be linked (added to the current element hierarchy in the *Explorer* pane), or just displayed in the drawing.

The Layout view binding feature allows you to:

- Include drawing shapes, lines and arrows that display element condition color; and update automatically as the element's condition changes.
- Use children containers to display child elements for any number of levels of children that can be filtered and sorted as desired.
- Insert text showing element name, DName or attribute values.
- Show charts or gauges of element performance metrics.

This section focuses on the steps to bind drawing shapes or text to elements as well as create children containers. It is assumed that you know how to use the drawing toolbar to add shapes, arrows, lines, and text. For more information, see Section 3.3, "Adding Shapes and Lines," on page 22.

- Section 5.1, "Associating Elements to a Drawing," on page 39
- Section 5.2, "Dynamically Transforming Text, Shapes and Lines with Element Properties," on page 40
- Section 5.3, "Changing Default Drill-Down Behavior Using Drawing Shortcuts," on page 48
- Section 5.4, "Displaying an Element's Children using Child Containers," on page 48
- Section 5.5, "Integrating Performance Metric Charts and Gauges into a Drawing," on page 54

5.1 Associating Elements to a Drawing

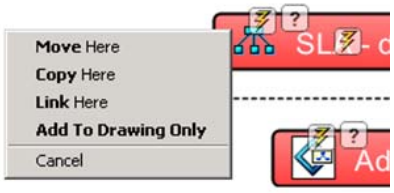
The term "binding" is used to describe the process of linking an element to a component of the Layout drawing. The element can originate from any element branch. It can be copied, moved, or linked to be a child of the element associated with the drawing, or it can simply be added to the drawing and not added to the current element's hierarchy.

To bind an element to a drawing:

1. In the *Explorer* pane, select the element whose Layout drawing should be modified and open the *Layout* tab.

   Any element can be selected in any hierarchy.
2. Click *Switch to Edit Mode*.

3 In the *Explorer* pane, drag an element from any part of the hierarchy and drop it on the background of the Layout view to display a menu for linking, moving or copying the element:



To expand or collapse the element hierarchy in the *Explorer* pane so you can locate an element for dragging and dropping, click the plus (+) or minus (–) symbols. This action does not exit the drawing for the current element.

IMPORTANT: Avoid a common error of inadvertently changing the Layout view drawing from one element to another. While editing the drawing for one element, do not left-click other elements in the *Explorer* pane. This switches the Layout view from the current element to the clicked element.

---

4 Select one of the following:

Move Here: Cuts and pastes the element (and its children) as a child of the target element, both in the *Explorer* pane and in the Layout view.

Copy Here: Pastes a copy of the element (and its children) as a child of the target element, both in the *Explorer* pane and in the Layout view.

Link Here: Creates a link between the added element (and its children) and the element associated with the Layout view drawing. The added element and its children display in the Layout view beneath the element associated with the drawing and also display beneath the element in the *Explorer* pane.

Add to Drawing Only: Adds the dropped element to the Layout view drawing. No link is established between the element in the *Explorer* pane and the Layout view drawing element.

5 Save the drawing changes by pressing `Ctrl+S`, or by right-clicking the background and selecting *Save Drawing*.

The added element displays in the drawing using the default node style graphic. The element's current condition is represented by the graphic fill color.

5.2 Dynamically Transforming Text, Shapes and Lines with Element Properties

When any drawing component (such as a shape or line) has been linked to an element, it can be dynamically updated based on condition, an element attribute, or by using NOC Script.

For example, bind the element condition color to the shape fill color or line fill color. A red square representing a network helps quickly identify a CRITICAL network problem in a drawing.

You can easily configure the following:

- headings or text that display element name, dName or attribute values.

  See Section 5.2.1, "Creating an Element Name Heading," on page 41 and Section 5.2.2, "Changing the Associated Element," on page 42

- shapes and lines that change colors based on element condition.

See Section 5.2.3, “Dynamically Updating a Shape’s Color based on Element Condition,” on page 44

- objects that display or hide based on condition levels.

  See Section 5.2.4, “Hiding and Showing Elements Based on Condition,” on page 47

- affect the drawings with custom behavior using NOC script.

  See Section 5.2.5, “Coding Custom Behavior,” on page 47

Typically, the bind is performed by creating the shape, then dragging and dropping an element to the shape. By default, the element’s condition is bound to the shape’s fill color.

IMPORTANT: Binding an adapter element (listed under *Administration* > *Adapters*) has unexpected results and is unsupported. The result of dragging and dropping an adapter element to a Layout view shape is that a duplicate adapter element with UNKNOWN status displays in the *Explorer* pane.

Keep in mind that the drawing itself remains bound to the element currently selected in the hierarchy (Explorer pane)—the drawing “owner.” Initially, element binding is inherited throughout the drawing. All children inherit the drawing owner’s DName information. This inheritance is changed when you use the Bind feature described in the following subsections to bind a different element to a child in the drawing.

5.2.1 Creating an Element Name Heading

You can add a text label that is bound to element name, DName, or element property attribute. Whenever the element changes, the content automatically updates based the newly associated element.

This feature is especially useful when creating node styles or templates that may be applied to multiple drawings.

IMPORTANT: Deleting elements that are bound to a graphic in the Layout view will remove the ability to update with the element’s state, property, and so on. However, the graphic remains in the drawing and if the element is restored, the graphic reanimates and the binding is restored. Otherwise, drop another element on the graphic to bind it.

To add a dynamically updated text label or heading:

1 Create placeholder text:

1a Click  (*Add Text*) and click inside the drawing, then enter placeholder text.



1b Edit the *Properties* and *Text* palette panes to change the color or font attributes.

2 Bind the text to the name, DName, or attribute of the current element:

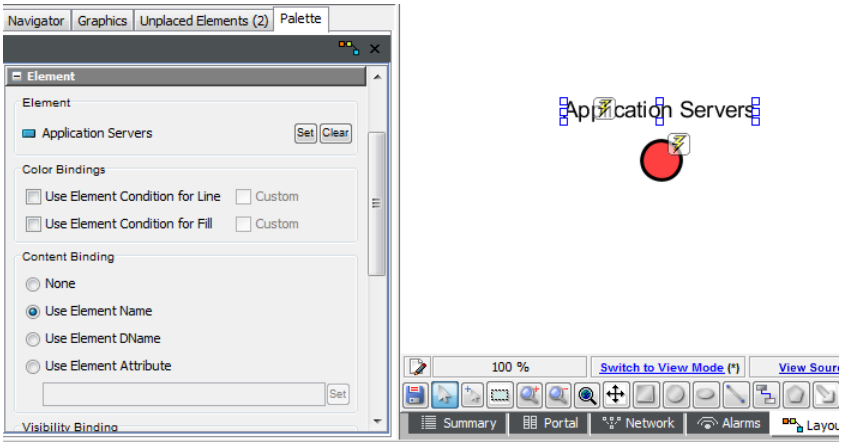
2a Select the text object.

2b In the *Palette*, select one of the options under *Content Binding*:

- **Use Element Name:** Displays the element's name.
- **Use Element DName:** Display the element's distinguished name.
- **Use Element Attribute:** Enter the attribute to display, then click *Set*.

  The list of valid attributes displays on the element's Properties page: *Element*, *Last Reported*, *Condition*, and so on.

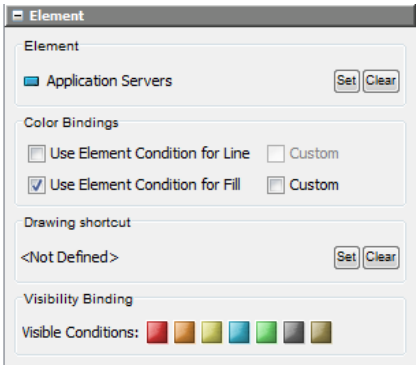
The following illustrates binding the text with Element Name. The text automatically updates with the element's name.



5.2.2 Changing the Associated Element

To bind a drawing component to a different Operations Center element:

1 In the *Element* pane, click *Set* in the *Element* section:



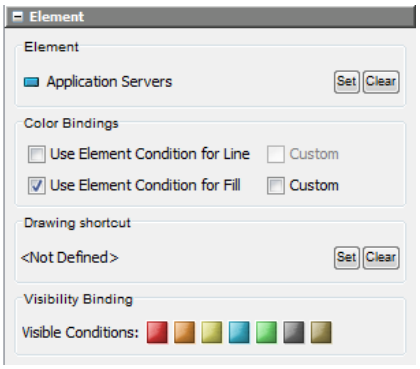
This opens the Browse for Element dialog box.

2 Select the element, then click *OK*.

The *Element* pane updates to display the selected element in the *Element* section, and the drawing component is linked to the newly selected element.

To remove the binding from the drawing component:

1 In the *Element* section of the *Element* pane, click *Clear*:



A dialog box displays asking you to confirm removing the binding.

2 Click *Yes*.

The drawing component is no longer bound to the element. However, keep in mind the drawing remains linked to the current element in the *Explorer* pane.

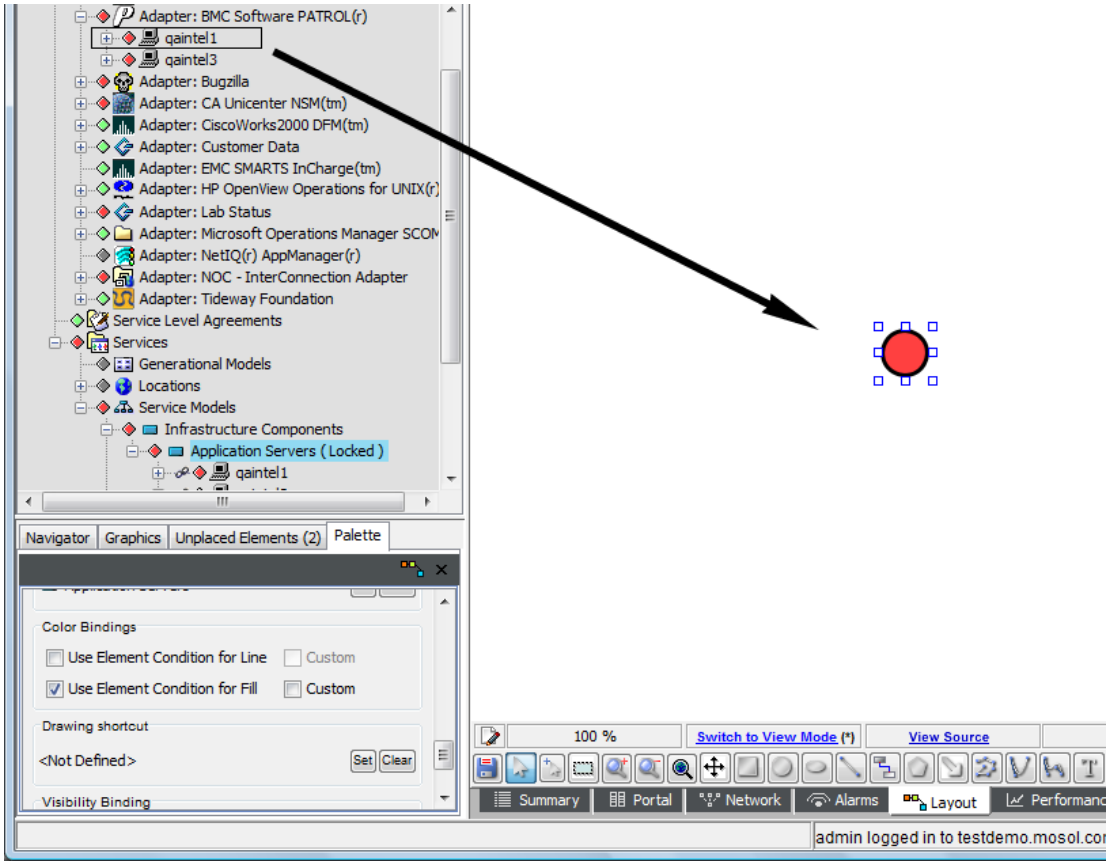
5.2.3 Dynamically Updating a Shape's Color based on Element Condition

After binding an element to a drawing component, decide whether to use the element's condition color to shade the drawing shape. Use this feature to have the drawing shape's color change automatically whenever the element condition changes.

There could be situations when you want to define the element condition color used to shade bound drawing components. For example, if an element condition is CRITICAL, MAJOR, or MINOR, the drawing component (such as a square) should be red. If the element is in any other condition, the square should be green, as if it were in the OK state.

Another application of this feature is to define a custom color for the default or set condition. For example, use a custom magenta color when the condition is MAJOR or MINOR.

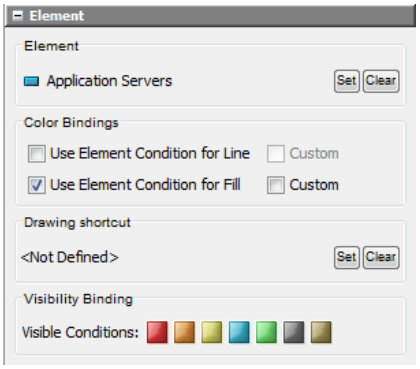
1 Associate the shape or line to the desired element by dragging any element from the *Explorer* pane and dropping it on the shape. By default, the drawing shape's fill color is linked to the element condition.



In the above illustrations, dragging and dropping the *Billing* element to the circle binds the element condition, which currently is CRITICAL (red), to the circle fill color. In the *Element* property pane, *Use Element Condition for Fill* is selected by default.

2 In the *Element* pane, select one or both of the following check boxes:

- *Use Element Condition for Line*
- *Use Element Condition for Fill*



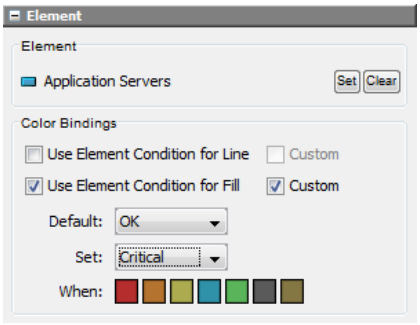
The drawing component colors update to match the element condition.

Note: If an adapter shuts down and any of its element conditions are bound as fill colors to drawing shapes, these shapes change to the UNKNOWN state. No tooltips are available while the adapter is down.

3 To customize the condition color:

3a Click the *Custom* check box next to *Use Element Condition for Line or for Fill*.

3b Select *CRITICAL* from the *Set* drop-down list, then click the red, orange, and yellow squares next to *When*:



If the linked element is in the CRITICAL, MAJOR, or MINOR state, the drawing component is shaded red, as though it were *CRITICAL*.

3c Select *OK* from the *Default* drop-down list.

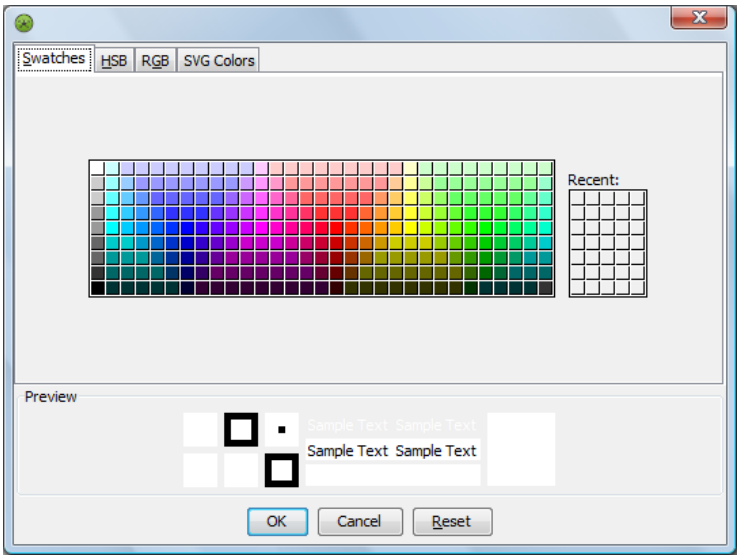
If the linked element is in any other state, the drawing component is shaded green, as though it were *OK*.

4 To define a custom condition color:

4a Select the *Custom* check box next to *Use Element Condition for Line or for Fill*.

4b Select <*custom*> from the *Set* drop-down list.

4c Click the color block to the right to display the custom color palette:



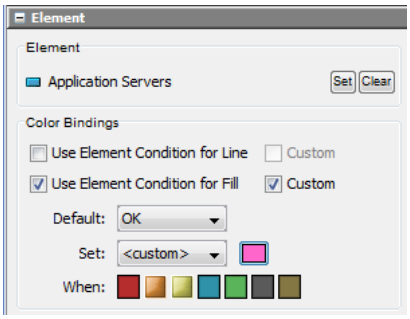
4d Define the custom color, then select it.

4e Click the orange (MAJOR) and yellow (MINOR) squares.

If the element condition is either MAJOR or MINOR, the bound drawing component is shaded using the custom color (in this example, magenta).

4f In the *Default* drop-down list, select a condition to use when the element is in any other state except MAJOR or MINOR.

In this example, the drawing component is shaded green (OK) if it is in any state except MAJOR or MINOR:



5.2.4 Hiding and Showing Elements Based on Condition

The Visibility Binding feature specifies the element conditions that determine whether elements are displayed or hidden in the Layout view. By default, all conditions are displayed.

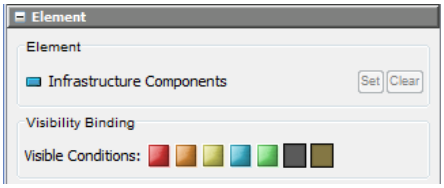
Elements can be hidden only in drawings that are in the View mode (read only).

To hide elements with one or more conditions:

1 In the *Visibility Bindings* section of the *Element* pane, click one or more condition *Color* buttons to hide all elements that have the conditions when the drawing is in View mode (read only).

The *Color* button appearance changes when it is hidden.

In Edit mode, all hidden elements display:



5.2.5 Coding Custom Behavior

NOC Script can be used to customize the behavior for drawing component colors or any additional behaviors as can be imagined. Use any of the options as outlined in the following table, then enter your custom NOC Script.

Shape Attribute	**Right-click menu option:**
Fill, Solid Color	*Customize* > *Add Binding* > *Fill*.
Fill, Gradient Color	*Customize* > *Add Binding* > *Condition Gradient* > *Fill*.
Stroke, Solid Color	*Customize* > *Add Binding* > *Stroke*.
Stroke, Gradient Color	*Customize* > *Add Binding* > *Condition Gradient* > *Stroke*.
Stoke Width	*Customize* > *Add Binding* > *Stroke-Width*.
Opacity	*Customize* > *Add Binding* > *Opacity*. Opacity configurations require the use of a custom NOCscript or a direct edit of the SVG source. For an example, see "Shape Transparency Based on Element Property Value" on page 140.
Custom Effect	*Customize* > *Add Binding* > *Script*.

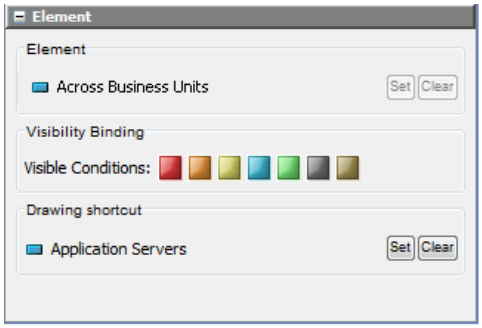
5.3 Changing Default Drill-Down Behavior Using Drawing Shortcuts

The drawing shortcut feature enables you to double-click an element in the Layout view and go to (select) a different element in the *Explorer* pane. This rule provides an exception to the default drill-down behavior for an element.

To set up the Drawing Shortcut:

1. In the *Drawing Shortcut* section of the *Element* pane, click *Set*.
2. Navigate to the element, then click *OK*.

   The selected element name displays in the *Drawing Shortcut* section:



When the *Across Business Units* object is double-clicked in the Layout view, the Layout view drawing drills in to show the *Application Servers* drawing, which is in a different branch of the element hierarchy, because of the defined Drawing Shortcut.

TIP: When an element is bound to a shape and you hover the mouse over the element in View mode, the status bar displays the name of the bound element as a link to the shape. However, when a drawing shortcut exists and you hover the mouse over the affected element, the status bar displays the drawing shortcut element name as the link to the shape.

5.4 Displaying an Element's Children using Child Containers

Child Containers make it easy to create a container of an element's children. After a Child Container is added to a drawing, you can decide to:

- Filter elements.
- Modify how elements are sorted.
- Change the layout of how element nodes are displayed.
- Change the number of levels of children shown.
- Specify the maximum number of children shown.
- Apply a new node style.

  For more information about Node Styles, see Chapter 6, "Working with Node Styles and Custom Graphics," on page 65.

NOTE: Stack containers are another type of container that can be added to a drawing. Stack containers are especially useful when creating new node styles as they can layer multiple objects which can be hidden or shown based on a condition that is met. For more information about Stack

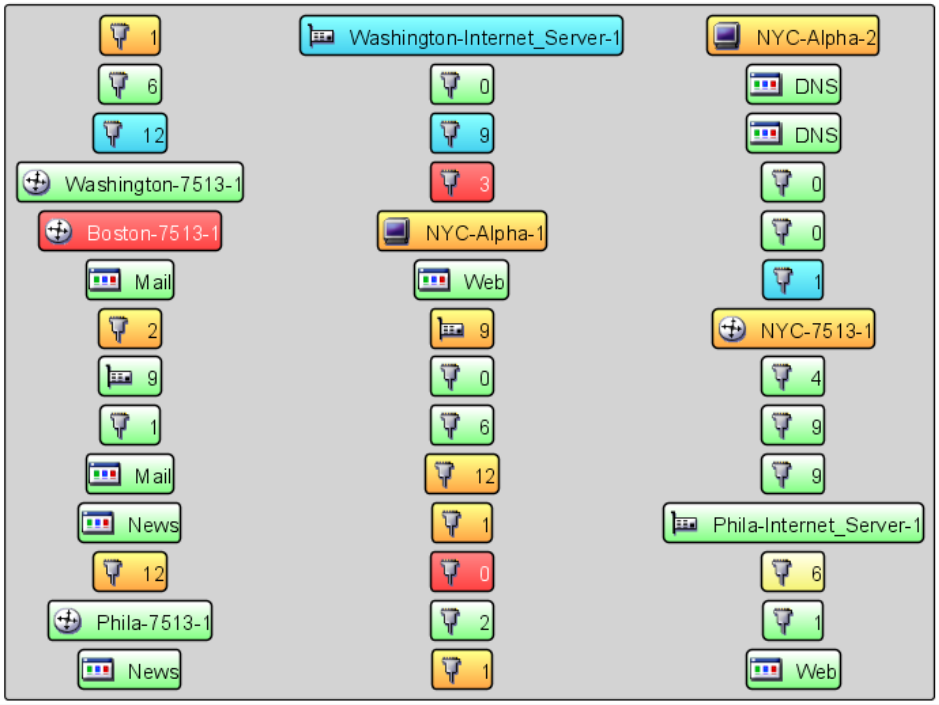
Containers, see Section 6.7, “Stacked Container Example: Toggling Graphics Based on Condition State,” on page 79 and Section C.4, “Looking at the Source Code with a Toggling Graphic Node Style Example,” on page 158.

To add a child container to a drawing:

1. Open the Layout view for the desired element.
2. Right-click the background of the Layout View, and select *New Drawing*.

   All elements are removed leaving the Layout view empty and in Edit Mode.
3. Right-click the Layout View, and select *Customize* > *Container* > *Add* > *Children Container*.

   A child container displaying all elements displays:



4. To filter the elements show, right-click the container, and select *Customize* > *Child Container* > *Filtering* > *Adjust Filtering*. Do any of the following:
   - Include the current element (“self”) in the drawing.
   - Look for matching elements under non-matching elements. This is useful when you are going more than 1 layer deep.
   - Use regular expressions to include elements based on name, dName, or class.
   - Exclude or include elements based on condition levels.
   - Or write a script to apply a custom filter.
5. To change the number of columns, right-click the container, then select *Customize* > *Child Container* > *Layout* > *Adjust Layout*. In this example, we selected 4 columns.
6. To sort the elements, right-click the container, then select *Customize* > *Child Container* > *Layout* > *Sorting*. You can sort by condition, name, dName, or class.

   In this example, we selected *Sort Direction* > *Ascending* and *Sort By* > *Condition* to show elements with the most severe condition first.



7 To adjust element depth or maximum number of elements shown, right-click the container, then select *Customize* > *Child Container* > *Depth* > *Adjust Depth*. In this example, we didn't make any changes to depth settings.

8 To change the style of the element nodes, do the following:

8a Right-click the container, then select *Change Group*.

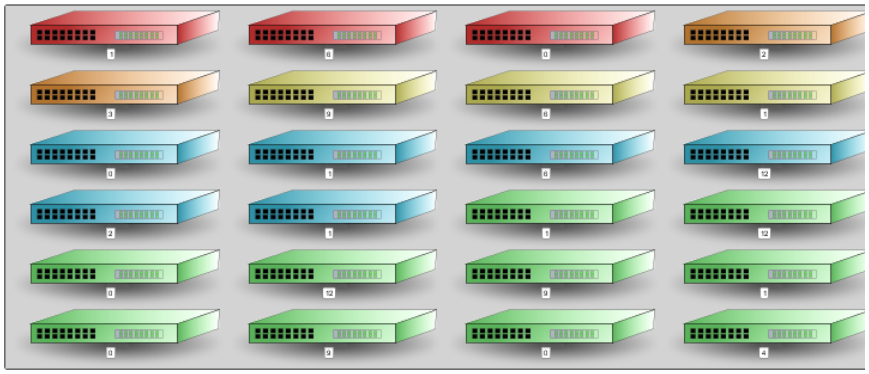
8b In the *Graphics* panel, select the *Nodes* category and locate the desired node style.

In this example, we selected *Technology* > *Networking* > *Router*.

8c Drag the node style image over onto an elements inside the container.

8d Select *Change Node Style for All Nodes* from the pop-up menu.

All element nodes update with the new node style:

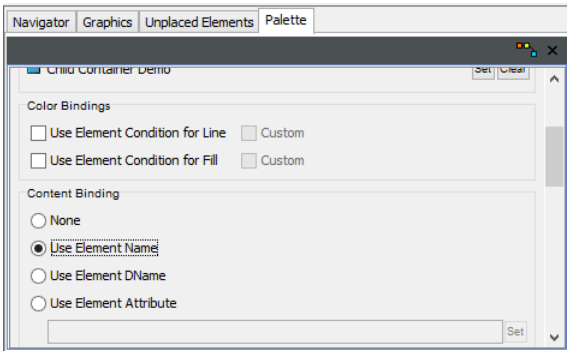


8e Right-click the container, then select *Save Group*.

9 To add an Element Name label, do the following:

9a Click the Text drawing tool, and click the Layout background.

9b Select the text object, and in the Palette, select *Content Binding* > *Use Element Name*.



9c Select both the child container and the text object, and select *Customize* > *Container* > *Wrap in Container*.

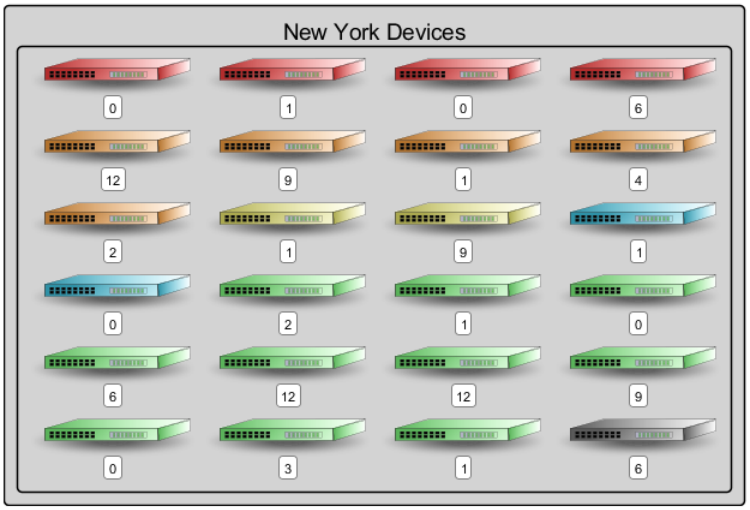
9d Right-click the container and select *Customize* > *Container* > *Layout* > *Set Layout*. Set the layout to *Flow*.

9e Do the following to order the objects and remove the background color on the container.

9e1 Right-click the container, and select *Change Group*.

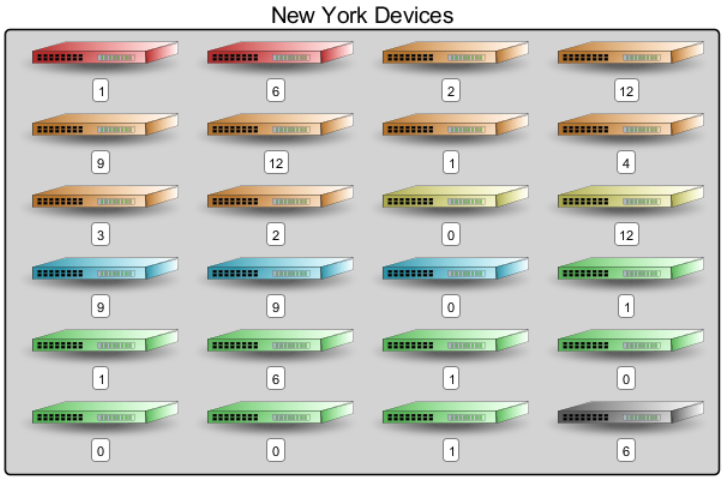
9e2 Right-click the text object, and select *Arrange* > *Send to Back*. The text moves to the top but disappears behind the container background.

9e3 Right-click the outer container, and select *Arrange* > *Send to Back*. The element name (text object) is now visible above the child container.



9e4 To make the outer container transparent, select the outer container, and in the *Palette* panel in the *Properties* section, and remove the color selections (*X* buttons in color selectors) for both fill and border.

9e5 Right-click the container, and select *Change Group*.



5.4.1 Customizing a Child Container

Child Containers can be customized to display every configuration of elements that you need. The following sections address the various options you can apply on Child Containers:

- "Quick Reference for Child Container Settings" on page 52
- "Adjusting the Layout Type and Layout Options" on page 53

Quick Reference for Child Container Settings

Table 5-1 provides a list of the various settings that allow you to customize the container in your Layout drawing.

Table 5-1 *Child Container Settings Quick Reference*

For options to set:	Right-Click the container object and select...
Current element	*Customize > Child Container > Filtering > Adjust Filtering*
Element depth	*Customize > Child Container > Depth > Adjust Depth*
Filter by name, Dname, or class. Filter using custom script.	*Customize > Child Container > Filtering > Adjust Filtering*
Filter on condition state	*Customize > Child Container > Filtering > Adjust Filtering*
Include matching children under non-matching elements	*Customize > Child Container > Filtering > Adjust Filtering*
Maximum elements	*Customize > Child Container > Depth > Adjust Depth*
Number of columns	*Customize > Child Container > Layout > Adjust Layout*
Sort by, Sort order	*Customize > Child Container > Layout > Sorting*

Adjusting the Layout Type and Layout Options

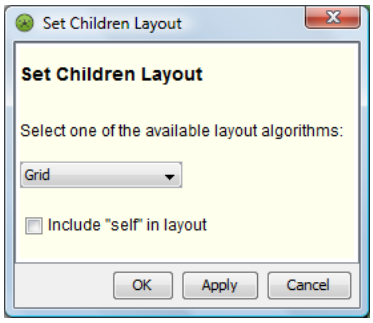
There are many types of layouts available for a container that set how components are displayed in the container. Grid and Flow layouts have additional settings for configuration.

Table 5-2

Layout Type	Great for...
Grid	Organizing many elements/objects using a series of columns and rows. Set how many items display before wrapping to the next row. Set filters based on element class or condition and adjust element depth settings.
Flow	Keeping a text label displaying with another object like a child container. This layout lets you set alignment of objects to left, center or right. The order in which objects display rely on the current order/arrangement of objects from front to back.

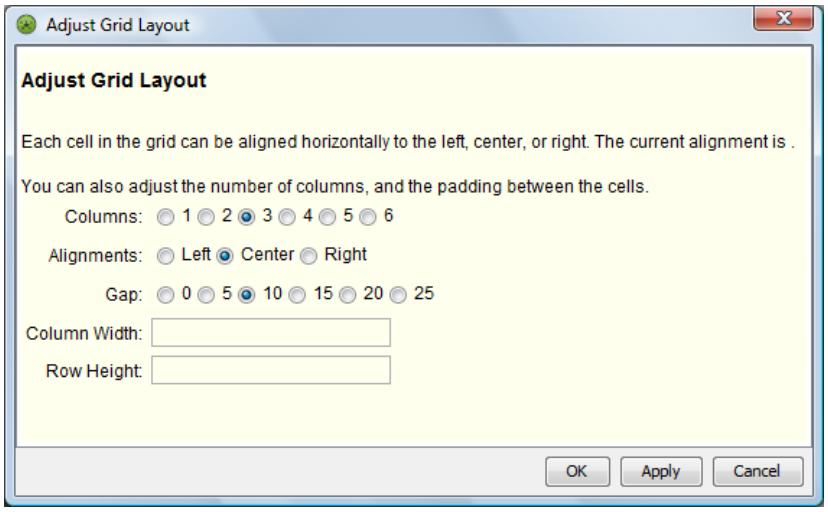
To change the layout type and set layout settings:

1. In the drawing, right-click the container, then select *Customize* > *Child Container* > *Layout* > *Set Children Layout*.



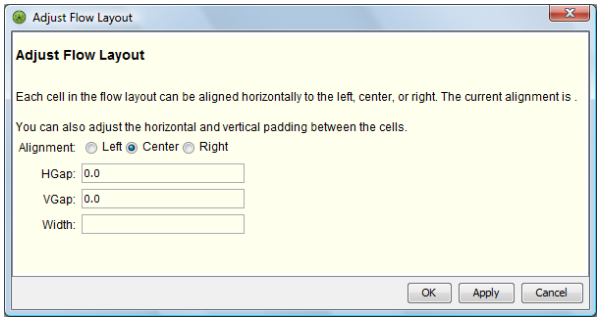
2. Select the layout type in the drop-down list.
3. To include the parent element in the container, select the *Include "self" in layout* radio button.
4. Click *OK*.

   The view updates to the new layout type.
5. To adjust settings for a Grid Layout, right-click the container and select *Customize* > *Child Container* > *Layout* > *Adjust Grid Layout*. The *Adjust Grid Layout* dialog box opens.



Specify options as desired for the number of columns, node alignment, spacing widths, column width and row height.

6 To adjust setting on a Flow Layout, right-click the container and select *Customize* > *Child Container* > *Layout* > *Adjust Flow Layout*. The *Adjust Flow Layout* dialog box opens.



Specify options as desired for the spacing and width.

5.5 Integrating Performance Metric Charts and Gauges into a Drawing

Charts and gauges for performance metrics can be easily added to any drawing. They can be added via the Layout *Customize* menu, or by applying them as a node style.

In order to use these features, Operations Center must be configured to capture historical performance data. For information on capturing performance metrics, see "Capturing Alarm and Performance History" in the *Operations Center 5.5 Server Configuration Guide*.

Layout drawing charts and gauges:

- Render performance metrics for the current element, another element, the parent or grandparent element, or a specified element or set of elements
- Compare one or more metrics (up to five) using a bar, stacked bar, line, or pie charts.
- Display information about a single metric series using various types of gauge displays.
- Can use regular expressions or scripts (FormulaScript) matchers to select metrics.

- (Charts) Must be configured for refresh interval, axis and grid values, and background color bands for acceptable condition (not available for Pie charts).
- (Gauges) Must be configured for refresh interval, minimum/maximum expected values, and condition threshold definitions.
- Provide various display options that allow you to toggle various information on and off inside the chart or gauge display.
- Can use one element for display name and algorithm setting context, and a another element for metric/source data.

Add charts or gauges to the Layout drawing via the Layout *Customize* menu, or by applying as a nodes style. For more information about node styles, see Chapter 6, "Working with Node Styles and Custom Graphics," on page 65.

The following sections describe how to add a chart to the Layout drawing and configure it:

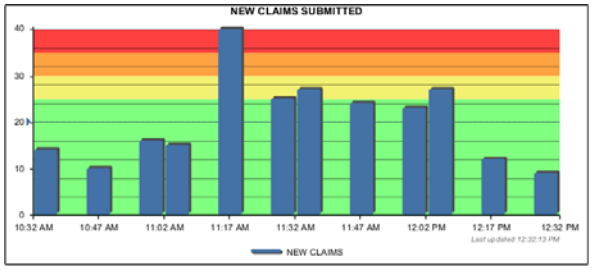
- Section 5.5.1, "Types of Charts," on page 55
- Section 5.5.2, "Types of Gauges," on page 56
- Section 5.5.3, "Adding Charts and Gauges," on page 58
- Section 5.5.4, "Configuring Chart and Gauge Properties," on page 58

5.5.1 Types of Charts

A series of charts are available to add to the Layout view drawing. All charts show information for one metric including current, average and range values; a trending line; and date/time of last update. Each chart is configurable with various options to hide or show any of this information. And, severity can be overlaid on the chart as background colors.

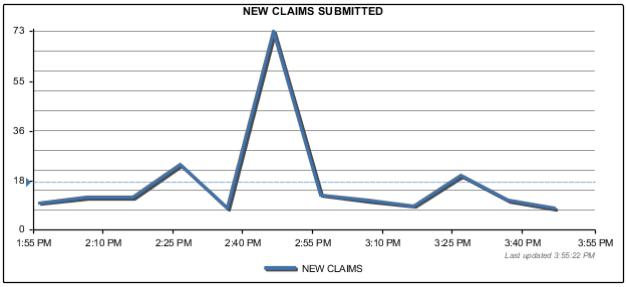
Bar

The bar chart uses vertical columns to chart the values of one or more metric series.



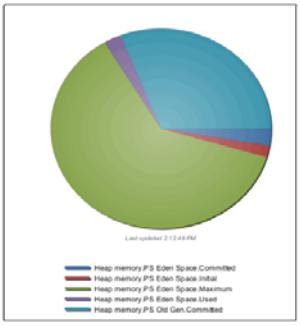
Line

The line chart plots data points in one or more metric series.



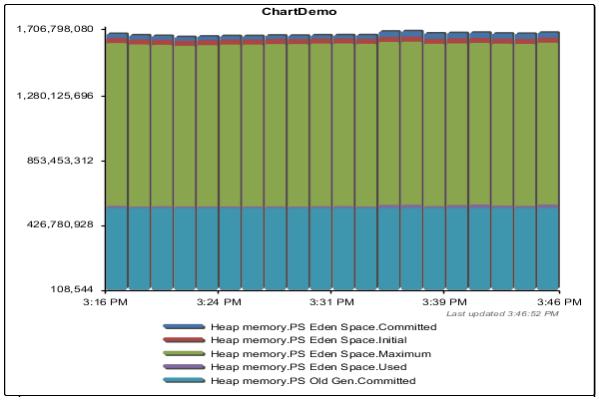
Pie

The pie chart is divided into proportional sections based on metric values. Use the pie chart when you want to compare the size of each metric as a slice with the whole of the pie.



Stacked Bar

The stacked bar chart compares the contribution of each metric to the total of all by using stacked columns. Use the stacked bar chart when you have multiple metrics and when you want to emphasize the total.

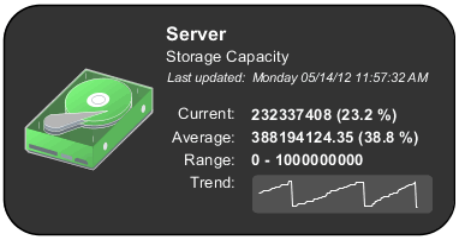


5.5.2 Types of Gauges

A series of gauges are available to add to the Layout view drawing. Gauges show information for one metric that can include current, average and range values; a sparkline; and date/time of last update. Each gauge is configurable with various options to hide or show any of this information. And, depending on the type of gauge, the current value is interpreted using a needle gauge, filled cylinder, progress bar, or lit LED bars.

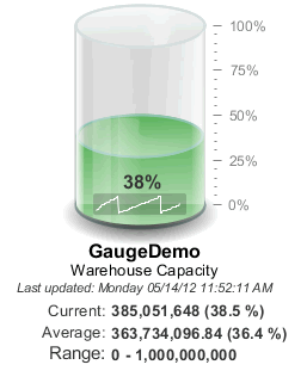
Basic Display

The basic display gauge shows various information about a metric's performance. Any graphic from the Operations Center *Graphics* > *Clipart* or *Graphics* > *Nodes* libraries can be included in the gauge display. The condition of the source element is used to updated the background color of attached graphic, when possible.



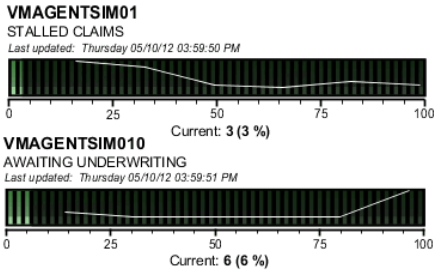
Cylinder

The cylinder gauge interprets the current value of a metric as a volume in a three-dimensional cylinder. Based on gauge configurations, the metric value is evaluated for severity which updates the background color of the volume fill inside the cylinder.



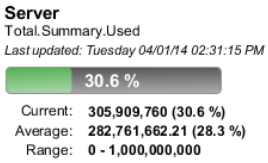
LED Bar

The LED gauge interprets the current value of a metric to light up bars on an LED display. Based on gauge configurations, the metric value is evaluated for severity which updates the background color of the bars.



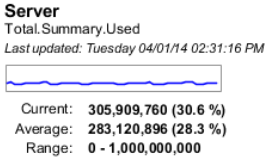
Percent Bar

The Percent Bar gauge interprets the current value of a metric to fill out the progress bar. Based on gauge configurations, the metric value is evaluated for severity which updates the background color of the progress bar.



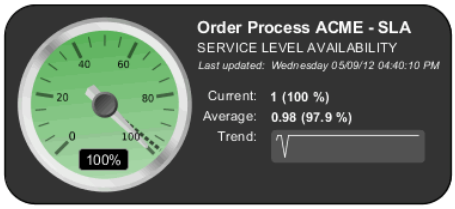
Sparkline

The Sparkline gauge shows a small trending line, called a sparkline, using the recent values of a specified metric.



Speedometer

The speedometer gauge interprets the current value of a metric and displays this “speed” on an analog-style needle. Based on gauge configurations, the metric value is evaluated for severity which updates the background color of the needle display.



5.5.3 Adding Charts and Gauges

To add a chart or gauge to the Layout drawing:

1 In the drawing (Edit Mode), right-click the background, then select *Customize*, then do one of the following:

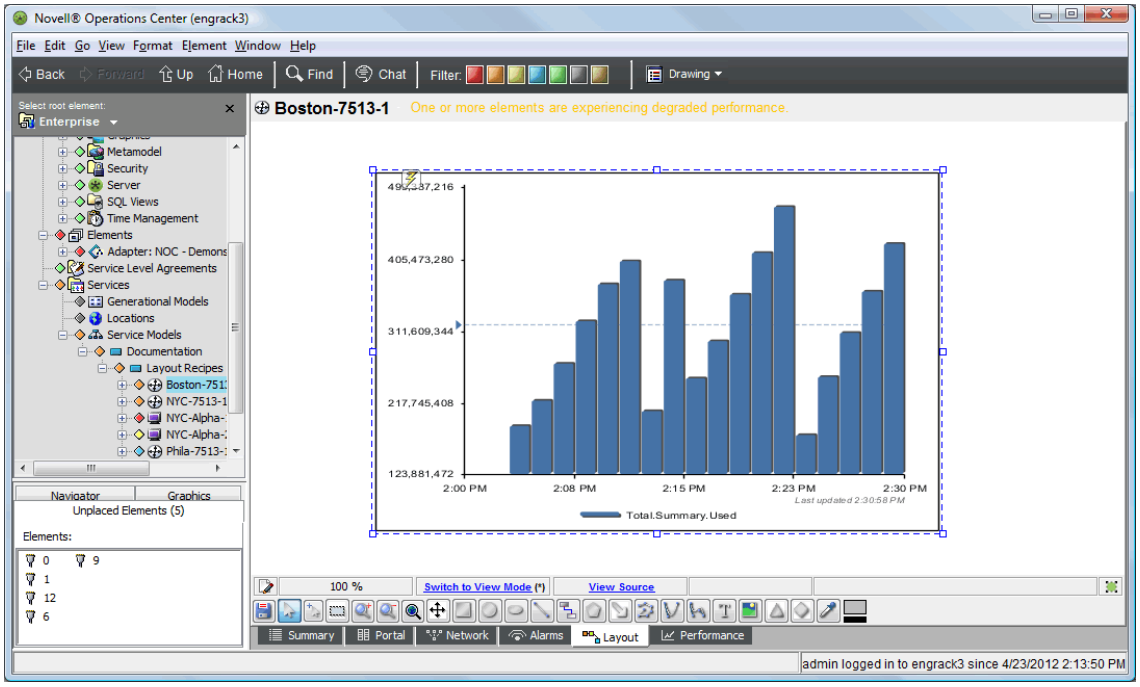
- To add a chart, go to *Customize* > *Charts* and select the chart type.

  Select from options to add a bar, line, pie or stacked bar chart.

- To add a gauge, go to *Customize* > *Gauges* and select the gauge type.

  Select from options to add a basic, cylinder, LED bar, percent bar, sparkline, or speedometer gauge.

The chart or gauge is added to the Layout drawing.



In the illustration above, we added a bar chart.

By default, charts are rendered showing the Operations Server metric, `Total.Summary.Used`, for a 30 minute interval. For these, as well as gauges, you must configure the source element and data properties (desired metric(s)) in Properties.

2 Continue Configuring Chart and Gauge Properties to customize the chart.

5.5.4 Configuring Chart and Gauge Properties

After a chart is added to the drawing, you can configure it. At a minimum, you must select the source element and define the metric(s) to chart.

- "Using Drag & Drop to Change Element Context" on page 59
- "Updating Chart and Gauge Properties" on page 59

Using Drag & Drop to Change Element Context

For some charts and gauges, you may want the context to be different from the natural element (the element currently selected in the Explorer pane). Setting a context association changes the defaults for chart/gauge titles as well as the context used when applying Source Element settings. For example, if the chart/gauge is looking for a metric from a parent element, the parent element of the context element is used to retrieve metric data. For information on Source Element settings, see Step 8 on page 62.

To change the source element using drag & drop:

1. While maintaining focus of the Layout drawing chart/gauge, locate the new element in the Explorer pane.
2. Click the new element and drag it over onto the chart or gauge. A message displays, "Are you sure you want to change this drawing item to be bound to element *ElementName*?"
3. Click *Yes*.

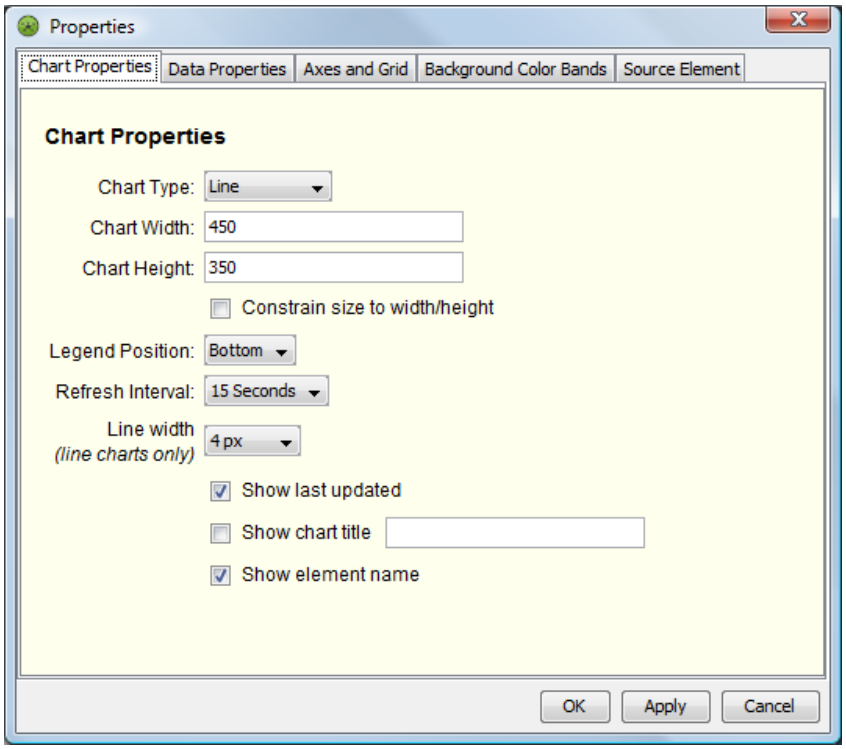
   The chart is now driven by the new element, it uses this element for the chart/gauge name as well as the context used for some Source Element settings.

Updating Chart and Gauge Properties

To configure a Layout drawing chart/gauge:

1. In the drawing (Edit Mode), right-click the chart or gauge, then select *Customize* > *Properties*.

   The Properties dialog opens.



   Properties tabs and options vary between the type of chart or gauge selected. The above illustration shows Chart Properties for Bar, Line and Stacked Bar charts.

By default, the chart or gauge is rendered using the Operations Center Server element, using the `Total.Summary.Used` metric, for last 30 minutes of data.

2 ***(Charts)*** In the *Chart Properties* tab, configure various display properties:

Chart Type: Select from *Bar*, *Line*, *Stacked Bar*, or *Pie*.

IMPORTANT: After changing the chart type, click *OK* to save the change and exit the Properties dialog. Then, reopen *Properties* to configure properties specific to the new chart type.

Chart Width: Specify the width of the chart, in pixels.

Chart Height: Specify the height of the chart, in pixels.

Select *Constrain size to width/height* to enforce chart dimensions in the Operations Center Dashboard. Use this option to keep charts uniform in size when there is more than one chart included in a view. Padding is adjusted from the inner chart, as needed, to keep the chart consistent with the size dimensions.

Legend Position: Specify the visibility and location of the legend. Select *Hidden* to hide the legend. Select from *Bottom*, *Right*, *Top* or *Left* to indicate where to show the legend.

Refresh Interval: Specify the amount of time between reload of chart data.

Line Width: (Line charts only) Specify the width of the chart line, in pixels.

As appropriate, select from the following check box options:

- Select *Show last updated* to show the date/time that the last update was received.
- Select *Show chart title* to show a title for the chart. Type the desired text in the field provided.
- Select *Show element name* to show the element's name in the chart title. If a title is specified in the option above, element name shows as a subtitle under the title.

3 ***(Pie Charts)*** In the *Pie Chart Properties* tab, configure various display properties:

Gradient: Select from *On* or *Off* to adjust the rendering of color in the pie chart slices.

Shadow Width: Use the drop-down to specify the thickness for the pie chart's shadow.

Sort Order: Specify the order in which to show slices inside the pie chart. Select from *Series Ascending*, *Series Descending*, *Values Ascending*, or *Values Descending*.

Legend Format: Specify possible additional information to display in the legend beside the metric name. Select from *Value Only*, *Percent Only*, *Both Value and Percent* or *Neither Value or Percent*.

4 ***(Gauges)*** In the *Gauge Properties* tab, configure various display properties:

Gauge Title: Specify a title to show in the top of the gauge display.

Last Update: Specify the syntax to display the date and time of last update. Use the Java SimpleDateFormat to specify date format, and wrap informational text in apostrophes.

For example, the default `'Last updated: ' EEEE MM/dd/yy hh:mm:ss a` outputs the following: `Last updated: Tuesday, 07/10/12 12:15:05 PM`.

Minimum (Expected): Specify the lowest valid value you expect the metric to be. This value is used when performing threshold evaluations.

Maximum (Expected): Specify the highest valid value you expect the metric to be. This value is used when performing threshold evaluations.

Refresh Interval: Specify, in seconds, the amount of time between reload of gauge data.

Condition Thresholds: ***(Non-Basic Gauges Only)*** Select from the following options to specify how the condition applies when evaluating the gauge value against threshold limits:

- **Condition applies when gauge is *equal or greater than* threshold:** Select this option to trigger the condition level when the gauge value is equal or greater than the threshold value.
- **Condition applies when gauge is *less than* threshold:** Select this option to trigger the condition level when the gauge is less than the threshold value.

Then, specify the percentage value for each threshold state. Threshold evaluations are calculated using the expected minimum and maximum values above. If the actual value is above the maximum expected value, the condition shows as the most severe condition level. If the condition is less than the minimum expected value, the condition shows as the least severe condition level.

Display Options: Select any of the following check box options to hide or show various information in the gauge display.

- **Show element name:** Deselect this option to hide the element name.
- **Show average value:** Deselect this option to hide the average metric value.
- **Show sparkline:** Deselect this option to hide the trending line.
- **Show range:** Deselect this option to hide the expected minimum and maximum values.
- **Show actual/current value:** Deselect this option to hide the current metric value.
- **Show metric name:** Deselect this option to hide the metric name, or Gauge Title value if specified.
- **Show last update:** Deselect this option to hide the last date and time the data was last refreshed.

5 In the *Data Properties* tab, define the performance metric to chart or drive the gauge and specify time range options:

Under *Specify Time Window*, specify the range of time to chart by entering a number and selecting from *Minutes*, *Hours* or *Days*,

If charting availability metrics, specify a calendar to apply in the *Use Calendar* drop-down list. Undefined time periods in the calendar are treated as blackout times and are not count against the element.

Under *Specify Metric*, specify the metric(s) to chart by selecting one of the following options:

- **Single Metric:** Charts the one metric as explicitly defined.

  Specify the name of the metric in the *Metric Name* field, then specify text to show for the metric's line in the *Label* field.

- **Specify Regular Expression Metric Matcher:** Charts up to five metrics using a regular expression.

  Specify a regular expression to match in the *Regular Expression* field.

  Even if the regular expression finds more than 5 matching metrics, the chart only shows the first 5 found.

- **Specify Script:** Charts up to five metrics by running the specified script.

  Edit the script example or enter valid NOC Script to capture the desired metrics.

  The script must be written using NOC Script. For more information, see the *Operations Center 5.5 Scripting Guide*. For information about available objects, see Section B.8, "Script Objects," on page 153

  For gauges, Even if the script finds more than 5 matching metrics, the chart only shows the first 5 found.

6 ***(Line, Bar and Stacked Bar Charts)*** In the *Axis and Grid* tab, configure properties related to the chart's axis and grid lines.

Chart Minimum Value: Enter the smallest value to show on the chart Y axis. Enter numeric values only without separators. If not specified, the chart's axis adjusts to show the full range of values. If specified and more data exists, the metric line will cap and shows as a flat line.

Chart Maximum Value: Enter the largest value to show on the chart Y axis. Enter numbers only without separators. If not specified, the chart's axis adjusts to show the full range of values. If specified and more data exists, the metric line will cap and shows as a flat line.

As appropriate, select from the following check box options:

- Select *Show series average lines* to chart a line for the average of each metric.
- Select *Show series line shadow* to show chart lines styled with a shadow.
- Select *Show X axis grid lines* to show grid lines running vertically.
- Select *Show Y axis grid lines* to show grid lines running horizontally.

7 ***(Line, Bar and Stacked Bar Charts)*** In the *Background Color Bands* tab, configure conditional banding based on severity ranges.

Select the check box for each severity to show banded in the chart background. For each selected, specify the *Floor* and *Ceiling* values, using numeric values only without separators.

8 In the *Source Element* tab, specify the relative element that supplies metric(s) data.

Self: Uses the natural element (current element selected in the Explorer pane), or the context element if one is applied. For more information on context elements, see "Using Drag & Drop to Change Element Context" on page 59.

Parent Element: Uses the parent of the natural or context element.

Grandparent Element: Uses the grandparent of the natural or context element.

Specify DName: Uses a specified element. Enter the full DName of the desired element.

Specify Custom DName With Script: Matches an element based on the script provided. Enter valid NOC Script code. For more information, see the *Operations Center 5.5 Scripting Guide*. For information about available objects, see Section B.8, "Script Objects," on page 153

9 (Basic Gauge) In the *Graphic* tab, specify an optional graphic to show in the gauge. The graphic defined must reside in the *Graphics* > *Clipart* or *Graphics* > *Nodes* libraries.

In the *Graphics Location* field, specify the path to the graphic using double-underscore "__" characters to separate folder and graphic names. Do not include the Graphics folder in the path.

For example, to reference `Graphics > Clipart > People > person`, specify the *Graphic Location* as: `Clipart__People__person`.

Special characters must be encoded. Use the steps below to capture the exact path location for the clipart or graphic file. If no graphic is specified or cannot be located, and the *Show Graphic* option is enabled, the gradient bubble nodestyle is displayed for the source element. To hide the graphic, unselect the *Show Graphic* check box.

To obtain the graphic location path, do the following:

9a Exit the gauge Properties dialog.

9b In the Explorer pane, located the desired graphic in *Administration* > *Graphics* > *Clipart* or *Administration* > *Graphics* > *Nodes*.

9c Right-click the graphic name, and select *Generate Graphic Location*.

The graphic location is copied to the clipboard.

9d Reopen the gauge Properties dialog, switch to the Graphic tab, and paste the location into the *Graphics Location* field.

10 Click *Apply* at any time to apply the changes. The chart updates.

11 Click *OK* to save changes and close the Properties dialog.

6 Working with Node Styles and Custom Graphics

One key feature of any drawing is the graphic used to represent the elements. This graphic is called a node style. The default node style is the Gradient Bubble, which displays rounded rectangles containing the element name with a fill color corresponding to the element's condition.

Operations Center ships with many node style examples that can be accessed from under *Administration* > *Graphics* > *Nodes*. A library of three dimensional technology-related node styles (with 2 condition fill options for most) can be found under *Administration* > *Graphics* > *Nodes* > *3D_Library*.

The following sections cover creating, applying, and managing node styles:

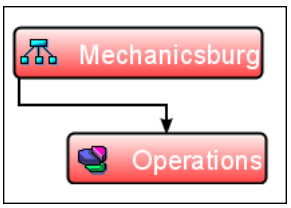
- Section 6.1, "Understanding the Node Style and the Node Style Library," on page 65
- Section 6.2, "Changing the Node Style," on page 67
- Section 6.3, "Applying the Ambient Node Style," on page 69
- Section 6.4, "Creating and Applying New Node Styles," on page 69
- Section 6.5, "Importing and Managing Node Styles," on page 76
- Section 6.6, "Associating Custom Graphics with Classes and Elements," on page 78
- Section 6.7, "Stacked Container Example: Toggling Graphics Based on Condition State," on page 79

6.1 Understanding the Node Style and the Node Style Library

Operations Center ships with many different node styles. It's easy to change the node style used in a drawing. Simply drag and drop a node style from the *Explorer* pane to an element in the drawing.

A dynamic node style is shaded using the current condition color of an element and displays text that represents the element name, as illustrated in Figure 6-1:

Figure 6-1 *The Default Node Style Updates Automatically with Current Condition Color*

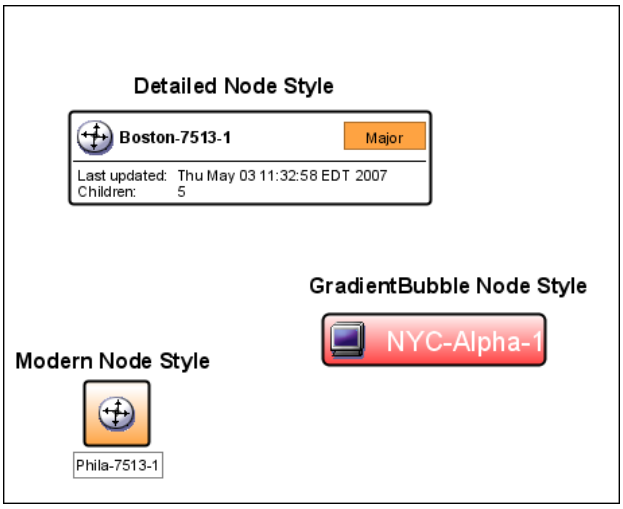


The fill color and text name change automatically when the linked element is updated. For example, in Figure 6-1, the Clean Bubble node style is bound to the *Mechanicsburg* element. If the element changes from CRITICAL to OK, the fill color changes from red to green.

All node styles shipped with Operations Center are dynamic. Just drag and drop a node style on an element and it uses the element name and condition color.

Figure 6-2 shows some of the standard node styles:

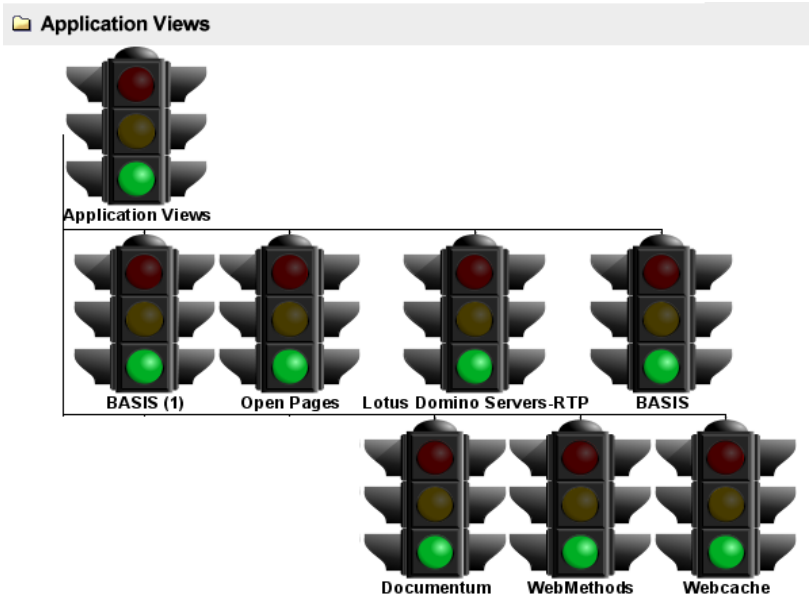
Figure 6-2 *Node Style Sample*



The node styles display under *Administration* > *Graphics* > *Nodes*. Click a node style and a sample drawing representing the node style displays in the Layout view.

A popular node style is the Stoplight, which uses a red, yellow, or green "light" to identify the status of a linked element. For example, a red light indicates an element has a CRITICAL or MAJOR severity, as illustrated in Figure 6-3:

Figure 6-3 *Layout View – Stoplight Node Style*



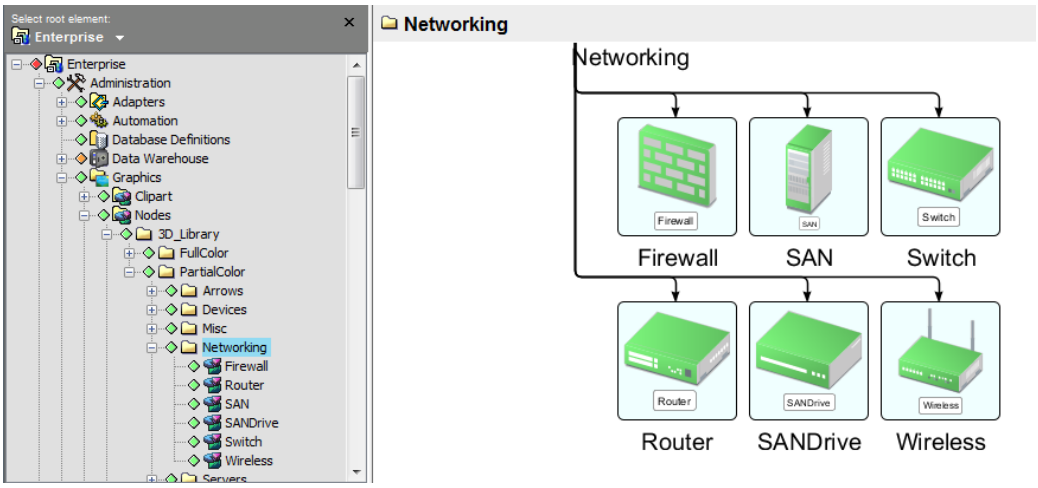
The seven severity levels used in Operations Center map to one of the three colored lights in the Stoplight node style, as defined in Table 6-1:

Table 6-1 *Stoplight Node Style Color Mapping*

Element Severity	Stoplight Color
CRITICAL, MAJOR	Red
MINOR, INFO	Yellow
OK, UNKNOWN, UNMANAGED	Green

Figure 6-1 illustrates some of the node styles that are available under the *3D_Library* folder.

Figure 6-4 *3D, Partial-Filled Node Styles*



You can change the node style for all elements in a drawing, for a specific element, or for all elements of a class.

6.2 Changing the Node Style

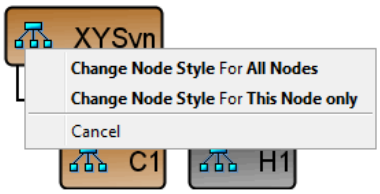
To create a standard view of all elements in a drawing, apply one node style to all the elements in a drawing.

- Section 6.2.1, "Applying a Node Style to All Elements in a Drawing," on page 68
- Section 6.2.2, "Changing the Node Style for One Element Only," on page 69

6.2.1 Applying a Node Style to All Elements in a Drawing

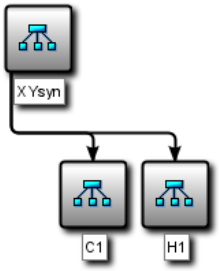
To apply a node style to all elements in a drawing:

1. In the *Explorer* pane, click the element whose Layout drawing is to be modified.
2. Click *Switch to Edit Mode*.
3. Drag a node style from the *Explorer* pane and drop it on an element in the drawing.
4. From the pop-up menu, select *Change Node Style for All Nodes*:



All elements in the drawing update to use the new node style. Additional elements added to the drawing also use the new node style.

The following illustrates all elements using the newly selected Modern Shadowed node style:



6.2.2 Changing the Node Style for One Element Only

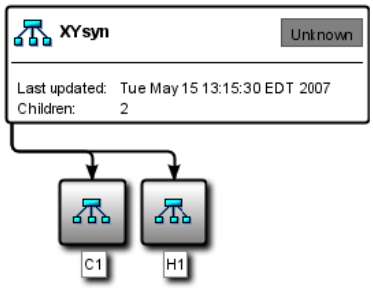
It is possible to use different node styles in a drawing. This can distinguish a set of elements or provide detailed information only for a few elements and save layout space for displaying additional elements using a more compact node style.

To change the node style:

1. Drag a node style from the *Explorer* pane and drop it on an element in the drawing.
2. From the pop-up menu, select *Change Node Style for This Node Only*.

   Only the selected element updates to use the new node style.

   The following illustrates that this drawing uses the Detailed node style for one element and the Modern node style for other elements:



6.3 Applying the Ambient Node Style

The Ambient node style changes all elements that currently use the system default node style. It is a way to quickly change multiple nodes at once.

To apply an Ambient node style:

1. Drag and drop a node style to the drawing background.
2. Click *Yes* when asked to confirm changing to the Ambient node style.

   All nodes using the system default node style change to the Ambient node style. Other nodes retain their current node styles.

6.4 Creating and Applying New Node Styles

You can create various types of node styles. They can be quite simple or more complex.

As an example, the following sections walk you through creating a dynamic node style that contains a small-sized circle and text that could represent a city on a map, and then applying the node style to a map.

For a different node style example using a Stack Layout Container to hide and show graphics based on state, see Section C.4, “Looking at the Source Code with a Toggling Graphic Node Style Example,” on page 158.

When defining a new node style, specify whether to make the node style dynamic by binding an element's name or condition color to the node style text or fill/line color.

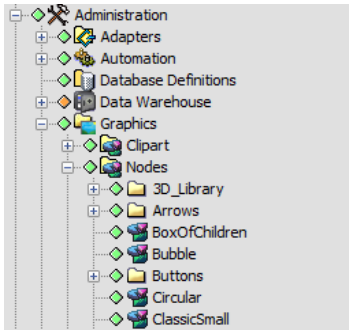
- Section 6.4.1, "Creating the Node Style," on page 70
- Section 6.4.2, "Testing the Node Style Using Preview Elements," on page 72
- Section 6.4.3, "Applying the Node Style," on page 75

6.4.1 Creating the Node Style

This sections shows how to create a small circle that might be used as a node style for a map.

To create a node style and add it to the drawing:

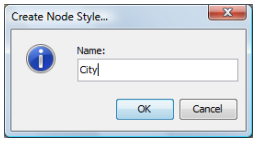
1 In the *Explorer* pane, expand *Administration* > *Graphics* > *Nodes* to display a list of existing node styles.



2 Right-click *Nodes*, then select *Create Node Style*.

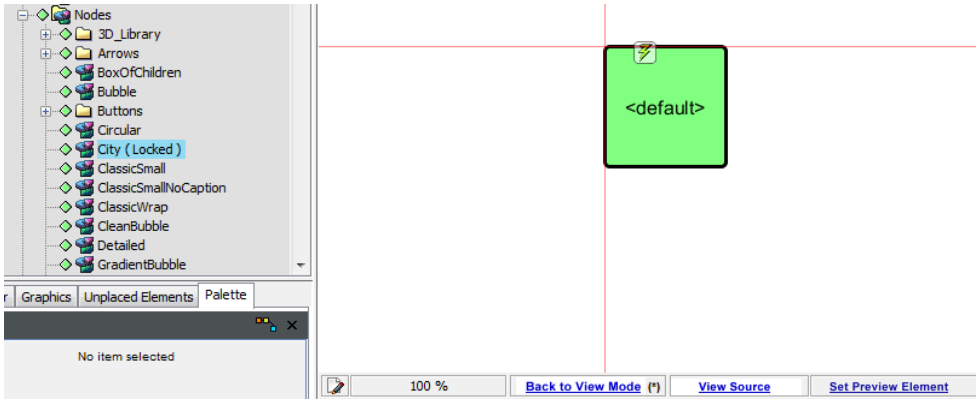
3 In the Create Node Style dialog box, specify a name.

In this example, it is City:



4 Click *OK*.

The node style is highlighted in the *Explorer* pane and the Layout view switches to Edit mode:



The node style uses the `default` template when creating a new node style. If the `default` template has been customized, the new node style might look different from the above illustration.

5 Click *Edit* > *New Drawing*.

6 In the confirmation dialog box, click *Yes*.

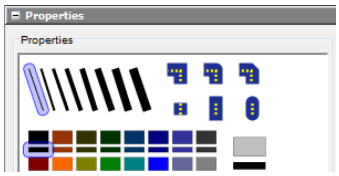
The default graphics are deleted, leaving a blank drawing area.

7 Verify the Editing Channel is set to Drawing.

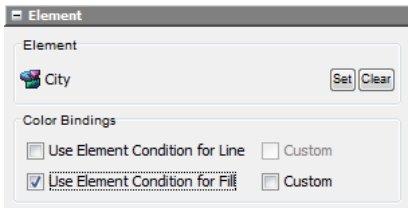
8 In the toolbar, click  (*Circle*).

9 Draw a small circle that represents a city.

10 In the *Palette*, in the *Properties* pane, select the thinnest line to outline the circle:



11 In the *Element* pane, select the check box next to *Use Element Condition for Fill*.



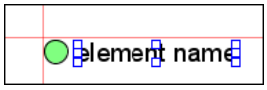
This colors each circle using a corresponding element's current severity code color (such as CRITICAL is red).

12 In the drawing toolbar, click  (*Text*).

13 Click anywhere on the drawing, then enter `element name` in the field.

14 Click *OK*.

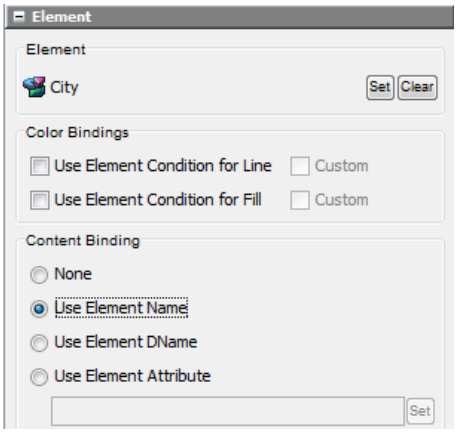
15 Drag the text to place it to the right of the circle, as shown:



Optionally, you can align the text and graphics with the red axes that represent the X, Y origin (0,0) for the node style.

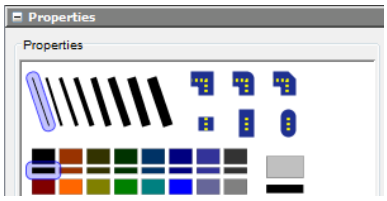
16 In the *Palette*, in the *Element* pane, select the *Use Element Name* radio button.

The text changes to the current element name:



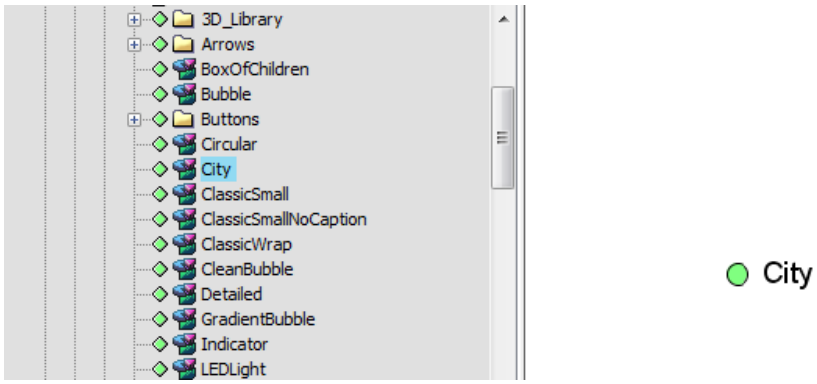
17 In the *Text* property pane, change the font size to 10.

18 In the *Properties* pane, change the text fill color to black:



19 Click *Edit* > *Save Drawing* (or use `Ctrl + s`).

The finished new node style is created and ready to be used.



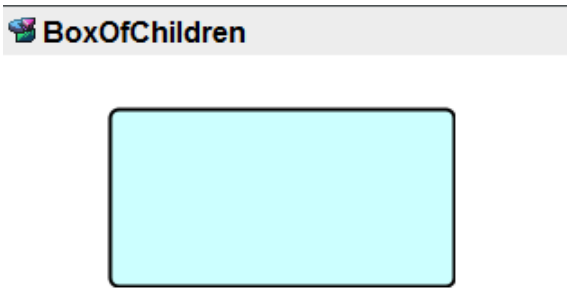
6.4.2 Testing the Node Style Using Preview Elements

Using a preview element allows you to test the node style without having to apply it to an actual element. You can set a preview element at point while you are creating a new node style.

Preview elements are especially helpful when creating or testing node styles that have element or child containers because they allow you to test the node style's behavior when more than one element is involved.

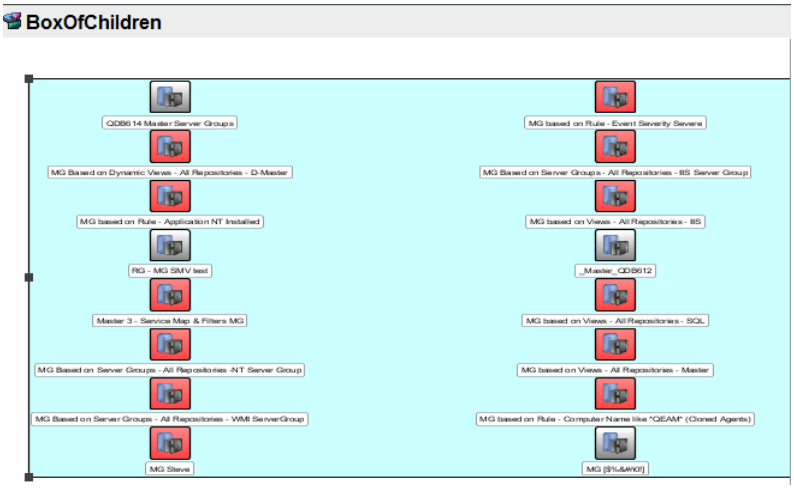
For example, in the illustration below, we have a node style that lists the elements children, but without setting a preview element, we just see an empty box.

Figure 6-5 *Box of Children Node Style No Preview Element Set*



After a preview element is set, as in Figure 6-6, you can see how the layout and behavior is, without having to test it by applying it to an existing element in your hierarchy.

Figure 6-6 *Box of Children Node Style With Preview Element*



Preview elements can be set for both node styles and drawing templates. For information on templates, see Chapter 7, "Using Drawing Templates," on page 83.

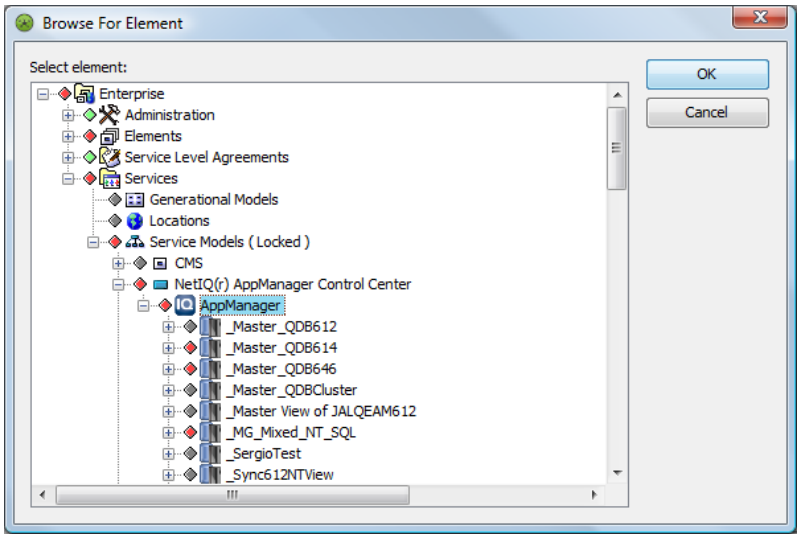
- "Setting a Preview Element" on page 73
- "Disabling/Disabling the Preview Element" on page 74
- "Clearing the Preview Element" on page 74

Setting a Preview Element

To set a preview element:

1. In the Layout view, click *Switch to Edit* to edit the node style (or template).
2. Do one of the following:
   - Click *Set Preview Element*.
   - Click *Change Preview Element*.

   A Browse For Element dialog opens.



3 Locate and select the desired element to be used to preview the node style or template.

4 Click *OK*.

The node style or template updates showing element information from the preview element setting.

Disabling/Disabling the Preview Element

To remove the preview element for a node style (or template):

1 In the Layout view, click *Switch to Edit* to edit the node style (or template).

2 Right-click the background of the Layout view, and select *Preview Element* > *Disable Element*.

The preview element settings are disabled to show the node style or template definition without a preview element.

3 To enable the preview element, right-click the background of the Layout view, and select *Preview Element* > *Enable Element*.

The preview element settings are re-applied to the node style or template.

Clearing the Preview Element

To clear preview element settings for a node style (or template):

1 In the Layout view, click *Switch to Edit* to edit the node style (or template).

2 Right-click the background of the Layout view, and select *Preview Element* > *Clear Element*.

The preview element setting is deleted. The node style or template definition displays without a preview element.

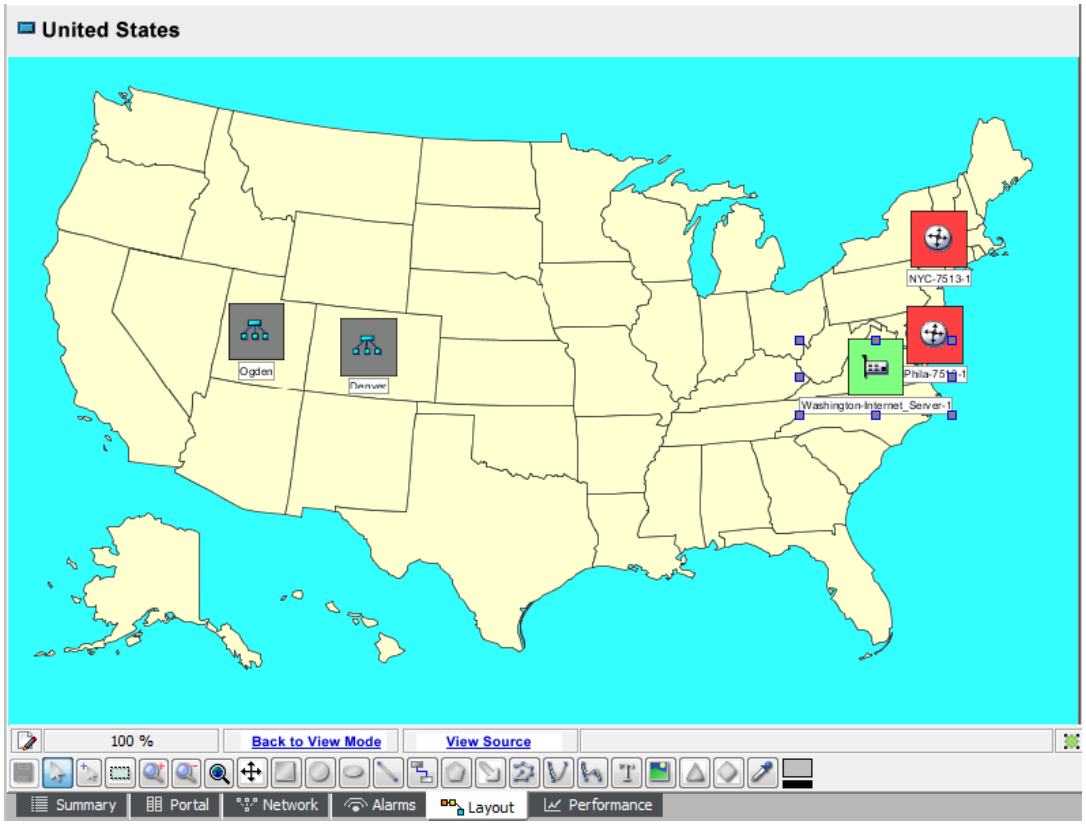
6.4.3 Applying the Node Style

The next step is applying a node style to one or more elements in a Layout drawing. This example uses the node style created in Section 6.4.1, "Creating the Node Style," on page 70.

To apply a new node style:

1. In the *Explorer* pane, select the element that contains child elements whose node styles should change.

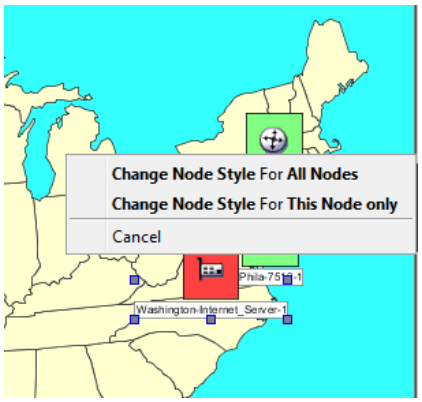
   In this example, a *Service Model* element named *United States* is selected:



2. In the bottom of the Layout view, click *Switch to Edit Mode*.
3. Expand *Administration* > *Graphics* > *Nodes*.

4 Locate the node style you want use (*City*) and drag it to an element (such as *NYC*).

A pop-up menu displays options for changing the node style:



5 Select *Change Node Style for All Nodes*.

All elements are updated to use the new node style.

Notice that the circle fill colors automatically change to match the city elements' current severities.

The following illustrates that the circles are bound to the element conditions and the fill colors automatically update when the element conditions change:



6.5 Importing and Managing Node Styles

In some cases, you might want to import a node style created on a different system.

Consider the impact and limitations of changing, renaming, or deleting node styles with regard to existing drawings. After renaming or deleting a node style, the next loading of a drawing that used the original node style fails. However, changing the content of a node style presents no problem because related drawings are updated upon the next viewing.

Do the following, as necessary:

- Section 6.5.1, “Importing a Node Style,” on page 77
- Section 6.5.2, “Deleting a Node Style,” on page 77
- Section 6.5.3, “Renaming a Node Style,” on page 77
- Section 6.5.4, “Adding a Node Style Folder,” on page 77
- Section 6.5.5, “Deleting a Folder,” on page 78

6.5.1 Importing a Node Style

To import a node style:

1. In the *Explorer* pane, expand *Administration* > *Graphics*.
2. Right-click *Nodes* (or any folder beneath it), then select *Import Node Style* to open the Import Node Style dialog box.
3. Navigate to and select the node style file, then click *Open*.

   The new node style displays in the *Explorer* pane.

6.5.2 Deleting a Node Style

Default node styles cannot be deleted. However, any custom node style you create can be.

To delete a node style:

1. Right-click a node style, then select *Delete Node Style* to open a confirmation dialog box.
2. Click *OK*.

   The node style disappears from the *Explorer* pane.

6.5.3 Renaming a Node Style

Default node styles cannot be renamed. However, any custom node style you create can be.

To rename a node style:

1. Right-click the element, then select *Rename* to open the Rename dialog box.
2. Specify the new name, then click *OK*.

   The new graphic name displays in the *Explorer* pane.

6.5.4 Adding a Node Style Folder

Create folders in the *Explorer* pane to organize node styles.

To add a node style folder:

1. In the *Explorer* pane, expand *Administration* > *Graphics*.
2. Right-click *Nodes* (or any folder beneath it), then select *Add Folder* to open the Create Node Style Folder dialog box.
3. Specify a name for the folder, then click *OK*.

   The new folder displays in the *Explorer* pane.

6.5.5 Deleting a Folder

Default node style folders cannot be deleted. However, any folder you create can be.

To delete a node style folder:

1. Right-click the node style folder, then select *Delete Folder* to open a confirmation dialog box.
2. Click *OK*.

   The folder name disappears from the *Explorer* pane.

6.6 Associating Custom Graphics with Classes and Elements

Use the Editing Channel to create custom graphics and drawings and associate them with specific classes and elements. You can also associate a node style or drawing template with an element or class.

A class association results in all elements created using the class inherit the associated graphic, drawing, node style, or template. It is possible to override the class association by selecting an element and changing the drawing, template, graphic, or node style associated with it.

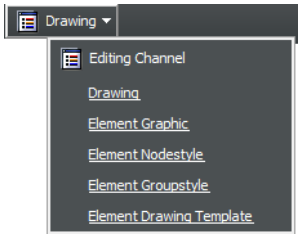
- Section 6.6.1, "Viewing a Drawing or Graphic," on page 78
- Section 6.6.2, "Creating and Associating a Custom Drawing or Graphic," on page 79
- Section 6.6.3, "Associating a Node Style or Drawing Template," on page 79

6.6.1 Viewing a Drawing or Graphic

To view a drawing or graphic associated with a class or element:

1. Do one of the following:
   - In the *Explorer* pane, expand *Elements* > parent elements (if necessary), then select an element.
   - In the *Explorer* pane, expand *Administration* > *Metamodel* > *Classes*, then select a class.
2. In the Layout view toolbar, click the *Editing Channel* drop-down list (which by default, is selected to and displays as *Drawing*), then select one of the following options:

Drawing
Element Graphic
Element Node Style
Element Drawing Template



The Layout view updates. The Layout view is empty if there is no graphic or drawing associated with the element or class, or if a new drawing was created without any graphics. If there is no associated node style or template, the Layout view displays "none" in the middle of the screen.

When a different element is selected in the *Explorer* pane, the Editing Channel returns to the default *Drawing* channel.

6.6.2 Creating and Associating a Custom Drawing or Graphic

To create and associate a custom drawing or graphic with an element or class:

1. Do one of the following:
   - In the *Explorer* pane, select an element.
   - In the *Explorer* pane, expand *Administration* > *Metamodel* > *Classes*, then select a class.
2. In the Layout view toolbar, click the *Editing Channel* drop-down list, then select *Drawing* or *Element Graphic*.

   The Layout view updates.
3. Click *Switch to Edit Mode*.
4. Create the drawing or graphic.
5. Press Ctrl+S to save the drawing or graphic.

6.6.3 Associating a Node Style or Drawing Template

Associating node styles and templates with an element or class requires dragging and dropping the node style or template on to the Layout view.

To associate a node style or drawing template with an element or class:

1. Do one of the following:
   - In the *Explorer* pane, select an element.
   - In the *Explorer* pane, expand *Administration* > *Metamodel* > *Classes*, then select a class.
2. In the Layout view toolbar, click the *Editing Channel* drop-down list, then select *Element Node Style* or *Element Drawing Template*.

   The Layout view updates.
3. In the *Explorer* pane, click the plus (+) symbols to expand *Administration* > *Graphics*, then expand either *Node Styles* or *Templates*.

   TIP: Take care not to click these elements. You do not want to change the focus from the class or element selected in Step 1.

   ---
4. Drag and drop a node style or template into the Layout view.

   The Layout view updates to display the selected node style or template.
5. To replace a node style or template in the Layout view, simply drag and drop a different node style or template into the Layout view.

   The view updates.
6. To delete the existing node style or template, click *Clear*.

6.7 Stacked Container Example: Toggling Graphics Based on Condition State

In this example, we'll create a node style that displays a different graphic matching the condition state of an element. When the element's condition updates, the node will show an appropriate weather graphic overlaid on a condition color-filled background. For example,

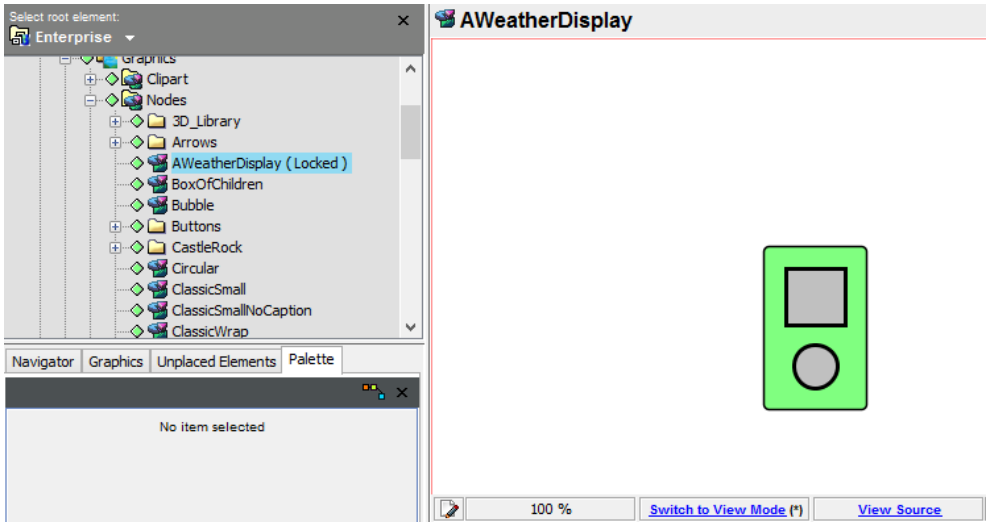


OK MAJOR CRITICAL

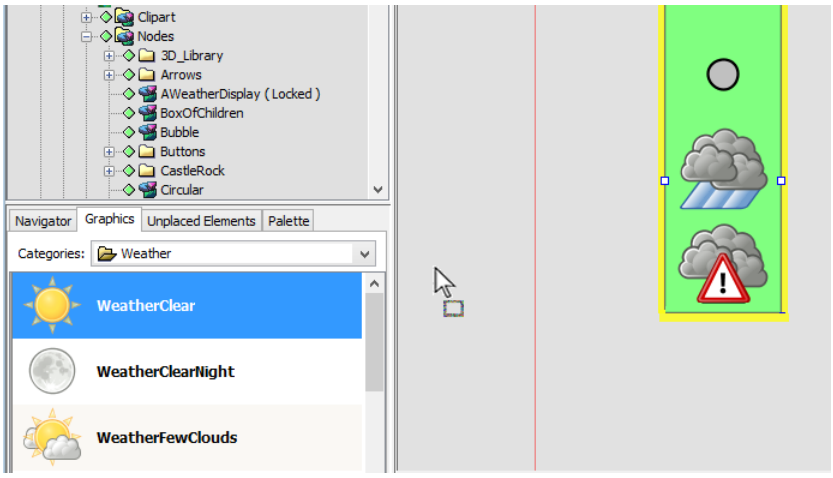
For more information on the clip art and graphics library, see Chapter 4, “Working with Clip Art and Graphics,” on page 33. For an additional example, that includes looking at the source code bind language, see and Section C.4, “Looking at the Source Code with a Toggling Graphic Node Style Example,” on page 158.

To create a stacked node style:

1. In the *Explorer* pane, expand *Administration* > *Graphics* > *Nodes* to display a list of existing node styles.
2. Right-click *Nodes*, and select *Create Node Style*.
3. Select your new node style and open the *Layout* view.
4. Right-click the background of the Layout View, and select *New Drawing*. All elements are removed leaving the Layout view empty and using Edit Mode.
5. Right-click the Layout View, and select *Customize* > *Container* > *Add* > *Stacked Layout Container*. A container displays showing two placeholder elements:



6. Right-click the container (green rectangle object), and select *Change Group*.
7. Add all the graphic files that you would like to display for condition changes.
   - **7a** Open the *Graphics* tab in the left pane and select the desired category.

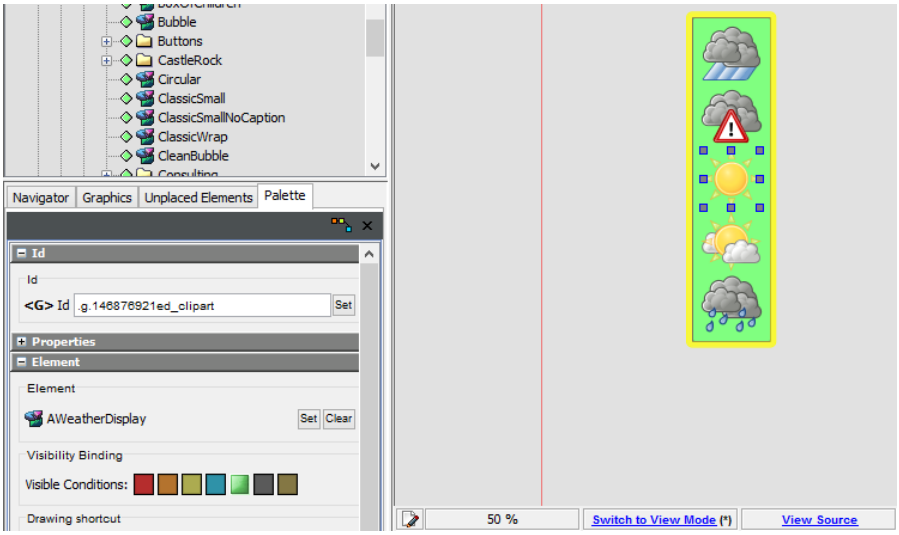


7b Click and drag any of the graphics to anywhere on the background of the Layout view. One by one they are automatically added to the container. The order that you add them is not important.

7c Click the default shapes or any unwanted objects in the container and delete them.

8 Associate the graphics to condition levels.

8a Click to select the clipart object you want to assign.



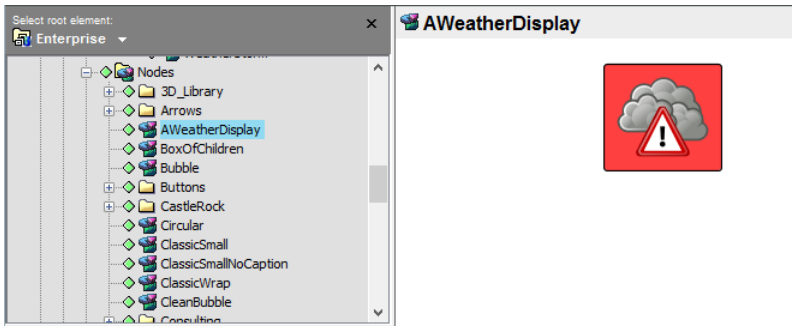
8b Open the *Palette* tab and select the condition states (*Visible Conditions* settings under *Element*) that must be present for this object to display. Click to enable or disable conditions. In this example, the sun displays only for OK condition values.

8c Repeat these steps for each graphic or object in the container.

9 Right-click the container and select *Save Group*.

10 To test the node style, click *Set Preview Element* and select an element in *Elements* or *Services*.

In this example, we selected an element with a CRITICAL condition. Only the *WeatherSevereAlert* graphic displays with the condition filled background. As expected, all other graphics are hidden for this state.



Be sure to test or double-check values for each condition state so that you know the behavior of the node style functions the way you'd like. One way to test is to apply the nodestyle to a branch of your *Services* hierarchy that has elements representing all condition states.

11 *Switch to View Mode* and save the node style.

12 Later, to make changes to the node style, do the following:

12a In Edit Mode, right-click the container, and select *Change Group*.

12b Make the desired changes.

12c When you are finished, right-click the container, and select *Save Group*.

12d Return to View Mode and save the changes.

7 Using Drawing Templates

Templates can save time when there is a need to create the same type of layout multiple times. For example, use the Radial template to lay out elements with the parent in the center and children as spokes. Operations Center ships with some default templates. An easy way to create custom templates is to copy a default template, rename it, then modify it.

The following sections cover all aspects of creating and applying drawing templates:

- Section 7.1, “Applying Templates to Drawings,” on page 83
- Section 7.2, “Unapplying Templates from Drawings,” on page 85
- Section 7.3, “Creating Templates,” on page 86
- Section 7.4, “Importing and Managing Templates,” on page 89

7.1 Applying Templates to Drawings

Applying a template to a drawing modifies the default, automatic drawing. If changes are made to the template, any drawings attached to the template are automatically updated with the changes.

When a drawing has a template applied to it, a icon displays under the lower right corner of the drawing. Hover the mouse over the template to view the name of the attached template.

IMPORTANT: A template applies to a drawing only for as long as you do not modify and save that drawing. If you make and save changes to a drawing after applying a template, the template is no longer applied to the drawing.

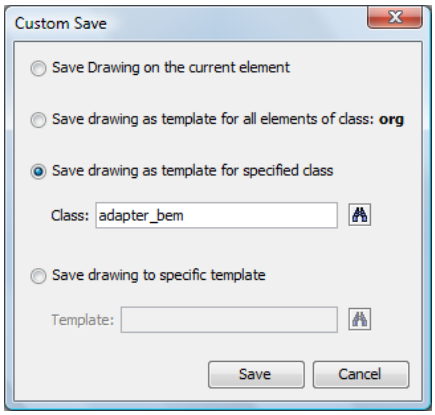
The following sections describe the various ways of applying templates to a drawing:

- Section 7.1.1, “Applying a Template to Elements of a Class,” on page 84
- Section 7.1.2, “Applying a Template to an Element,” on page 84
- Section 7.1.3, “Using a Template to Create a Custom Drawing,” on page 85

7.1.1 Applying a Template to Elements of a Class

To apply a template to elements of a specific class:

1. Do one of the following:
   - Open the *Layout* view containing the custom drawing you wish to apply as a class template.
   - In the *Explorer* pane, click to expand *Administration* > *Graphics* > *Templates* and open the desired template's *Layout* view.
2. Right-click in the background of the *Layout* view and select *Save as Custom*. The *Custom Save* dialog box opens.



3. Do one of the following:
   - If working from the layout view of an element with the same class as you wish to apply the template to, select the *Save drawing as template for all elements of class:* `className` radio button.

     The template is applied to any elements of the class and is saved as a template under *Administration* > *Graphics* > *Templates* > *_class* and named using the class name. Note that the template is not applied to the drawing you are saving from.
   - If working from the template directory or an element of a different class, select the *Save drawing as template for specified class*radio button then specify the class name.

     The template is applied to any elements of the selected class and is saved as a template under *Administration* > *Graphics* > *Templates* > *_class* and named using the class name. Note that the template is not applied to the drawing you are saving from.

     IMPORTANT: Saving a class template overwrites any existing template for that class.
4. Click *Save*.

7.1.2 Applying a Template to an Element

To apply a template to a single element:

1. Navigate to the desired element and open the element's *Layout* view.
2. On the console toolbar, change the *Editing Channel* to *Element Drawing Template*.
3. Drag and drop a template from *Administration* > *Graphics* > *Templates* to the element's Layout view background. The drawing is attached to the template.
4. Change the *Editing Channel* to *Drawing*.

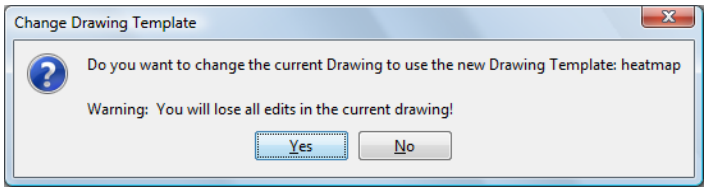
Existing elements are drawn using the shapes, arrows, colors, and so on inherited by the selected template. A template icon displays under the lower right corner of the drawing.

Future changes to the template are automatically inherited by the drawing.

7.1.3 Using a Template to Create a Custom Drawing

To apply a template as a custom drawing:

1. Drag and drop a template from *Administration* > *Graphics* > *Templates* to the element's Layout view background. A confirmation box opens.



2. Do one of the following:
   - Click *Yes* to update the drawing with template properties. A custom drawing saves for the element with properties of the template drawing.

     A custom drawing icon displays below the lower right corner of the drawing.

     The template is not dynamically linked to the element drawing and changes to the template do not affect the element's custom drawing.
   - Click *No* to cancel.

7.2 Unapplying Templates from Drawings

Depending on where you want to unapply a template, there are two ways:

- Section 7.2.1, "Unapplying a Template for an Element," on page 85
- Section 7.2.2, "Unapplying a Template for a Class," on page 85

7.2.1 Unapplying a Template for an Element

To unapply a template on a single element:

1. Open the element's *Layout* view.
2. Change the *Editing Channel* to *Element Drawing Template*.
3. Click *Clear*. The drawing reverts to the default automatic layout.

7.2.2 Unapplying a Template for a Class

The best way to unapply a template to a class of elements is to delete the template. When the template is deleted, all elements of that class revert to the default automatic layout.

Class templates are saved in *Administration* > *Graphics* > *Templates*> *_class* with the same name as the class they are applied to.

For information on deleting templates, see Section 7.4.3, "Deleting a Template," on page 89.

7.3 Creating Templates

There are two ways to create new templates:

- Create a new template and build the drawing from scratch.

  The disadvantage to working directly from the *Templates* directory is that there will only be one node available to work with.

- Save an existing custom drawing as a template.

  This method is best if you are working with containers. It is also easiest to visualize as you can work with in an environment where there are element children.

The following sections describe how to create templates:

- Section 7.3.1, "Creating a New Template," on page 86
- Section 7.3.2, "Creating a Template from a Custom Drawing," on page 87
- Section 7.3.3, "Editing Templates," on page 88

7.3.1 Creating a New Template

While it is easy to create new templates working with the *Templates* node, the disadvantage is that there is only one node visible while creating the drawing.

For more information about creating templates working with containers or child elements present, see Section 7.3.2, "Creating a Template from a Custom Drawing," on page 87.

To create a new template:

1. In the *Explorer* pane, click to expand *Administration* > *Graphics* > *Templates*.
2. Right-click *Templates*, then select *Create Template*. The *Create Template* dialog box opens.
3. Specify a template name.
4. Click *OK*.

   The new template displays in the *Explorer* pane and Layout view updates to display the drawing for the new template.
5. Verify the *Editing Channel* is set to *Drawing*.
6. Use the *Layout* view drawing tools to define a drawing that will become a new template.

   You can also drag and drop clip art, node styles, and other templates to define the new template.

   For information about editing the drawing and creating custom drawings, see Section 2.2, "Creating and Editing Custom Drawings," on page 15.

   For information on using the drawing tools, see Chapter 3, "Using Basic Drawing Tools and Concepts," on page 19.

   For information on creating custom node styles and adding graphics, see Chapter 6, "Working with Node Styles and Custom Graphics," on page 65.

   For information on testing the template by setting a preview element, see Section 6.4.2, "Testing the Node Style Using Preview Elements," on page 72.
7. Save the drawing for the new template by pressing *Ctrl*+*S*.

7.3.2 Creating a Template from a Custom Drawing

The advantage of creating a template from a custom drawing is that you can more easily work with containers and see how the template would be applied on a larger scale, where a hierarchy exists under an element.

While working with a custom drawing, you can also make it a template to apply to other elements or a class of elements.

To create a template from a custom drawing:

1 Open the *Layout* view containing the custom drawing you wish to save as a template.

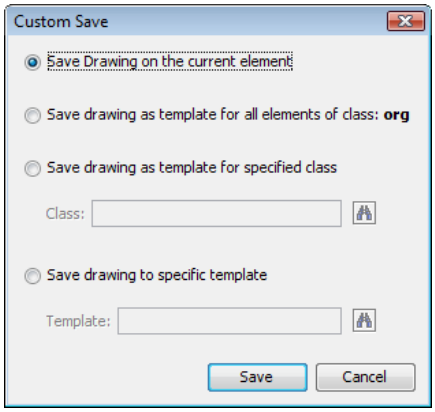
If you do not already have the custom drawing created, click *Switch to Edit Mode* and create a custom drawing that you wish to use as a template from any element.

For information about editing the drawing and creating custom drawings, see Section 2.2, "Creating and Editing Custom Drawings," on page 15.

For information on using the drawing tools, see Chapter 3, "Using Basic Drawing Tools and Concepts," on page 19.

For information on creating custom node styles and adding graphics, see Chapter 6, "Working with Node Styles and Custom Graphics," on page 65.

2 Right-click in the background of the *Layout* view and select *Save as Custom*. A *Custom Save* dialog box opens.



3 Do one of the following:

- If you've just made changes to the current drawing and wish to save them for that element, select the *Save drawing on the current element* radio button.

  IMPORTANT: Selecting this option cancels the link between the element and a template if one was previously applied.

- To save the drawing as a template for elements of the same class, select the *Save drawing as template for all elements of class:* `className` radio button.

  The template is applied to any elements of the class and is saved as a template under *Administration* > *Graphics* > *Templates* > *_class* and named using the class name. Note that the template is not applied to the drawing you are saving from.

  IMPORTANT: Saving a template for a class overwrites any existing template for that class.

- To save the drawing as a template for another class of elements, select the *Save drawing as template for specified class* radio button then specify the class name.

The template is applied to any elements of the selected class and is saved as a template under *Administration* > *Graphics* > *Templates* > *_class* and named using the class name. Note that the template is not applied to the drawing you are saving from.

IMPORTANT: Saving a template for a class overwrites any existing template for that class.

- To save the drawing as a template, select the *Save drawing to a specific template* radio button.

  Specify a new template name in the field or browse to select an existing template to overwrite.

  The drawing is saved as a template under *Administration* > *Graphics* > *Templates*.

4. To test the new template, locate the new template under *Administration* > *Graphics* > *Templates* and set a preview element.

   For instructions, see Section 6.4.2, "Testing the Node Style Using Preview Elements," on page 72.

5. Click *Save*.

7.3.3 Editing Templates

Templates can be edited and saved from *Administration* > *Graphics* > *Templates* or from the *Layout* view of an element where the template is actively applied.

To edit a template:

1. Do one of the following:
   - Open the *Layout* view of an element where the template is applied.

     To verify the template applied, hover the mouse over the  icon displayed under the lower right corner of the element's *Layout* view drawing. The tool tip displays the name of the linked template.
   - In the *Explorer* pane, click to expand *Administration* > *Graphics* > *Templates* and open the desired template's *Layout* view.
2. Click *Switch to Edit Mode* and update the drawing as desired.

   For information about editing the drawing and creating custom drawings, see Section 2.2, "Creating and Editing Custom Drawings," on page 15.

   For information on using the drawing tools, see Chapter 3, "Using Basic Drawing Tools and Concepts," on page 19.

   For information on creating custom node styles and adding graphics, see Chapter 6, "Working with Node Styles and Custom Graphics," on page 65.

   For information on testing the template by setting a preview element, see Section 6.4.2, "Testing the Node Style Using Preview Elements," on page 72.
3. Save the changes by pressing by pressing *Ctrl+S*.
   - If editing from an element's drawing, a confirmation box opens. Click *Yes* to save the changes to the template.

     Or, click *No* to view other options. For more information about these options, see Section 7.3.2, "Creating a Template from a Custom Drawing," on page 87

   Changes are saved to the template and the drawings for all linked elements update.

7.4 Importing and Managing Templates

- Section 7.4.1, “Importing a Template,” on page 89
- Section 7.4.2, “Organizing Templates Using Folders,” on page 89
- Section 7.4.3, “Deleting a Template,” on page 89
- Section 7.4.4, “Renaming a Template or Folder,” on page 90

7.4.1 Importing a Template

To import a template:

1. In the *Explorer* pane, click to expand *Administration* > *Graphics* > *Templates*.
2. Right-click *Templates* or any folder beneath it, then select *Import Template*. The *Import Template* dialog box opens.
3. Navigate to and select the template file, then click *Open*.

   The new template displays as a *Templates* element.

7.4.2 Organizing Templates Using Folders

To organize templates using folders:

1. In the *Explorer* pane, click to expand *Administration* > *Graphics* > *Templates*.
2. Right-click *Templates* or any folder beneath it, then select *Add Folder* to open the *Create Template Folder* dialog box.
3. In the *Explorer* pane, click to expand *Administration* > *Graphics* > *Templates*.
4. Do one of the following:
   - To create a new folder, right-click *Templates* or any folder beneath it, then select *Add Folder*. The Create Template Folder dialog box opens.

     Specify a name for the folder, then click *OK*.

     The new folder displays in the *Explorer* pane.
   - To delete a folder, right-click the folder, then select *Delete Folder*. A confirmation dialog box opens.

     Click *OK*.

7.4.3 Deleting a Template

To delete a template:

1. In the *Explorer* pane, click to expand *Administration* > *Graphics* > *Templates*.

   To delete a template for a class an return to using the automatic layout for class elements, open the *_class* directory.
2. Right-click the template, then select *Delete Template*. A confirmation box opens.
3. Click *OK*.

   The template disappears from the *Explorer* pane and any associated drawings return to the default automatic layout.

7.4.4 Renaming a Template or Folder

To rename a template or folder:

1. In the *Explorer* pane, click to expand *Administration* > *Graphics* > *Templates*.
2. Right-click the template or folder, then select *Rename*. The *Rename* dialog opens.
3. Specify the new name, then click *OK*.
   The new name displays in the *Explorer* pane.

8 Layout View Examples

The following sections give step-by-step instructions for creating various types of customized layout drawings:

- Section 8.1, “Geographical View Example,” on page 91

  This first example is a geographical Layout view that uses maps and dynamic elements that update automatically to show element conditions.

- Section 8.2, “Tabular Data View,” on page 98

  This second example shows a tabular view application, of a service catalog summary view. The colored rectangles identify the current status of individual services and the corresponding Service Level Agreement (SLA) health.

- Section 8.3, “Business Process View Example,” on page 101

  This third example shows the key steps in a business process, which is graphically represented in the Layout view. Each step is linked to an element condition to show critical events and relationships.

8.1 Geographical View Example

Businesses whose internal resources and services span multiple geographic areas find it useful to analyze performance and target troubleshooting methods by location. For example, mergers and other types of vendor or partner relationships create interrelated business processes, such as supply chain or post-sales support. Businesses that deliver products and services across geographic areas of varying size find it useful to view service delivery levels, outages, and other problems by location. This view can help prioritize problem resolution, organize delivery or service routes, and site visits, and identify and proactively protect at-risk areas from technology outages.

Use the Layout view to create a high-level view of the status of services and availability of resources by geographic area, without having to change existing element hierarchies or create new ones. It is easy to add a map to the Layout view, then organize relevant elements on the map.

Rather than sort through events based only on criticality or date, your staff can pinpoint “hot spots” that experience critical technology outages frequently, resulting in substandard delivery of services to customers. This helps in selecting sites for redeploying or purchasing new resources. It also keeps the IT staff focused on maintaining service availability to troubled regions, rather than just responding to alarm after alarm.

Understanding how resources impact service by location also helps IT staff alert the business to potential service disruptions. For instance, if IT cannot avoid problems with routers in the Boston area, the staff can tell business managers which customers might experience service disruptions. Together, the business and IT staff can take proactive steps to mitigate or even avoid trouble in the first place, as well as to accurately assist impacted customers.

Some business-critical applications rely on an IT infrastructure spread across multiple continents. The sheer scope of these resources makes finding the root cause of availability problems a difficult process. However, you can create Layout drawings that identify the state of applications and which particular technology components are impacting that state — all in real time. The team can quickly identify a business issue's source and location, as well as prioritize its support workload. For example, retail bank executives want to know the state of their business in real time, such as the state of ATM operations and telephone service in different regions and branches.

Layout views can show the state of key services by city, region or country. The views have the power to drill down from, for instance, regional ATM operations to the ATMs in a specific bank. Bank executives can hold IT accountable for their services. They have immediate information on the services that drive profits and customer satisfaction. If an outage is unavoidable on the technical side, the business can take action to mitigate the impact, such as by making alternate arrangements for customers and branches.

Assume a wireless company has recently completed a merger with a smaller company. IT managers must eliminate redundant hardware and software. The challenge is realizing cost-savings while guaranteeing the reliability of critical business services, such as point-of-sale resources and credit checking.

By integrating and interpreting data from across the provider's infrastructure and viewing it by geographic region, IT staff can identify technology issues to focus on first. For instance, IT executives can measure the results of a resource consolidation on service availability in key cities.

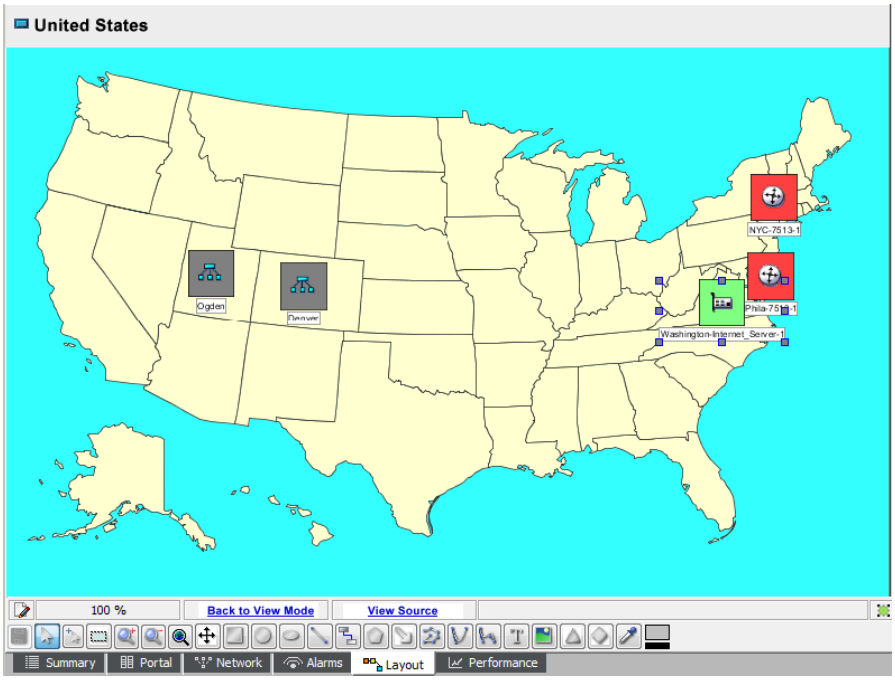
Another important goal after a merger is consolidating the large number of servers the company relied on. Managers can use Layout views to see high-level information, such as whether a key company application is available in certain regions. It's a way for IT to communicate value, as well to keep management aware of their ability to do business. Operations Center integrates data from the manufacturer's existing management tools. With this information, Operations Center continuously determines — in real time and over time — how IT resources are impacting overall services, services in a particular region, or key technology groupings in various data centers. The staff becomes focused on responding and even preventing service problems — not just reacting to alarms.

The company has organized its Operations Center business views by key applications, such as credit checking, point-of-sale data collection, and customer problem reports. These are the top-level elements in Operations Center. Within each application, server availability and performance data are rolled up into major sales markets, such as Boston, New York, Philadelphia, and Washington, DC.

Now assume the IT manager wants to view availability status for each application by city. The completed Layout view drawing in Figure 8-1 shows the current credit checking system availability in Boston, New York, Philadelphia, and Washington, DC. Red corresponds to CRITICAL status, orange corresponds to MAJOR status, and so on. CRITICAL cities require more detailed inspection, because it is likely multiple servers are down and the credit checking system might not be functioning in the affected sales centers.

Figure 8-1 illustrates that the cities are linked to existing elements and their current condition. The red CRITICAL squares indicates problems exist with the credit checking system in NYC and Washington DC.

Figure 8-1 *Map Example*



In this example, each city is represented by a small circle whose fill color matches the city availability condition and by text displaying the city name. Each city is bound to an element in the *Elements* hierarchy.

Creating this type of view consists of:

- Adding specific elements to the drawing
- Adding a map image to the Layout background
- Adding a title, change the background color
- Changing the background color
- Saving the drawing

To create conditions across a map:

1. In the *Explorer* pane, select an element.

   The Layout drawing will be associated with this element. In this example, we created a new *Service Model* element named *United States*.

2. In the bottom of the Layout view, click *New Drawing*. The drawing clears and switches to Edit mode.

3. Place the elements from the *Element* hierarchy that will be used in this drawing.

   The display format used for each element is called a node style.

   Operations Center ships with many different node styles. You can simply select one, then drag and drop it on a drawing.

   Select a node style to format the cities displayed in the drawing.

   3a In the *Explorer* pane, expand and locate the elements that should be surfaced in this drawing.

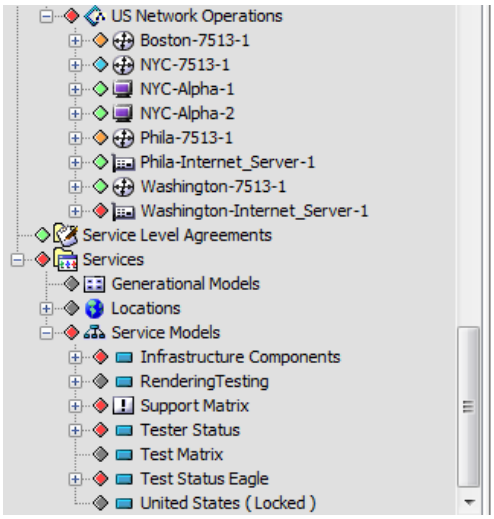
   In this example, we expand *Elements* > *US Network Operations*.

   You can click the plus (+) and minus (–) symbols to expand and collapse element hierarchies. However, do not click an element name in the *Explorer* pane or the Layout view shifts to that element and away from the *United States* element.

   3b From the *Explorer* pane, drag and drop Boston to the Layout view.

   3c From the pop-up menu, select *Add to Drawing Only*.

   The default node style displays to represent Boston:





   3d Expand *Administration* > *Graphics* > *Nodes*.

   3e Locate the node style you want use (ClassicSmall is used in this example) and drag it to Boston.

   3f From the menu, select *Change Node Style for All Nodes*.

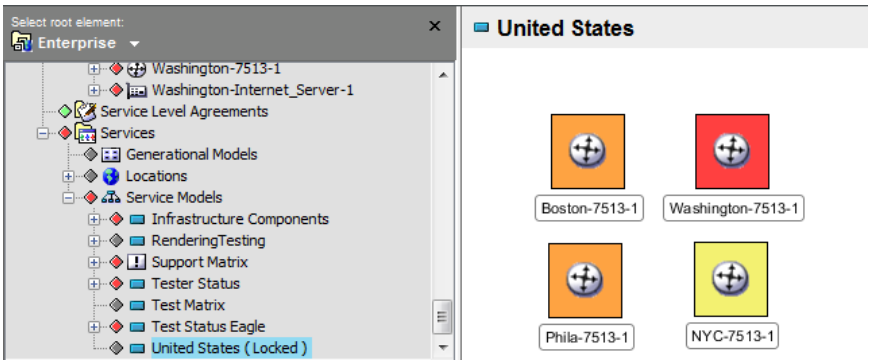
   Boston updates to use the new node style.

   3g Drag the remaining cities from the *Explorer* pane to the Layout view.

   Select *Add to Drawing Only* for all of them.

   Notice that the circle fill colors automatically change to match the city elements' current conditions.

The fill colors for the squares are bound to the element conditions:



4 Add a background map and arrange geographical locations:

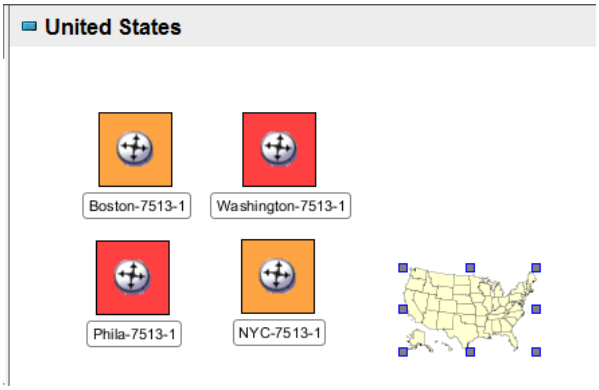
The final steps to complete the drawing consist of

A number of maps ship with Operations Center. You can also import maps. In this example. we are adding a background map of the United States and arranging the cities in the correct geographic locations.

4a Click the *Graphics* tab.

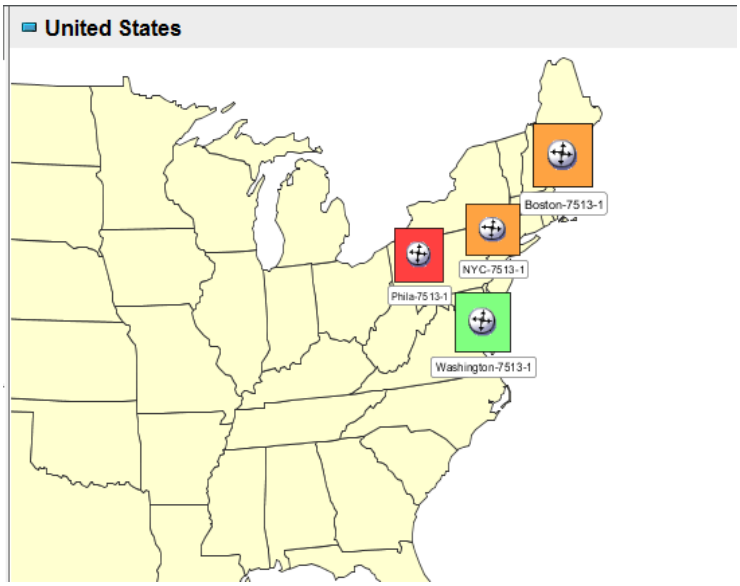
4b From the *Categories* drop-down list, select *Maps*.

4c Drag and drop the US Map to the view pane:



4d Right-click the map, then select *Arrange* > *Send to Back*.

4e Resize the map graphic by dragging its endpoints until it is approximately the same size as the one shown in the following:



4f Click and drag the cities to the appropriate geographic location on the map.

5 Add text to complete the drawing. We are adding a title and background color, before saving the drawing.

5a Create a title for the drawing by clicking  (*Text*) on the drawing tools toolbar.

5b Enter `Availability by City`.

5c Change the text size, font, or color using the property panes.

5d Drag the title to the appropriate location.

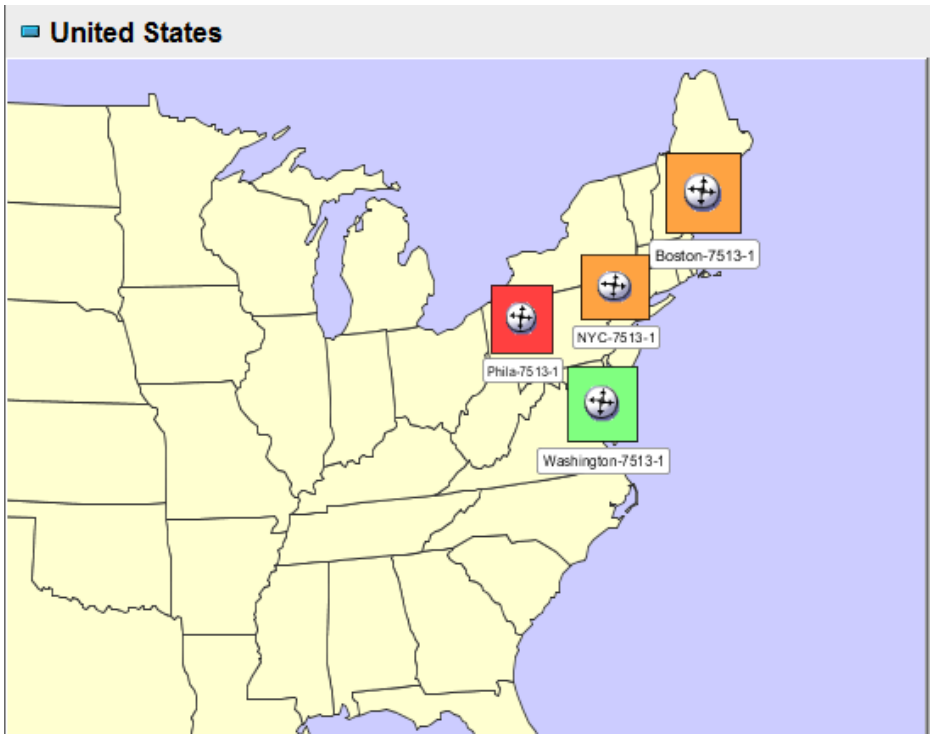
5e Right-click the Layout view, then select *Set Background Color*.

5f Select a color from the palette, then click *OK*.

The background color updates.

6 Click *Edit* > *Save Drawing*.

The finished drawing:

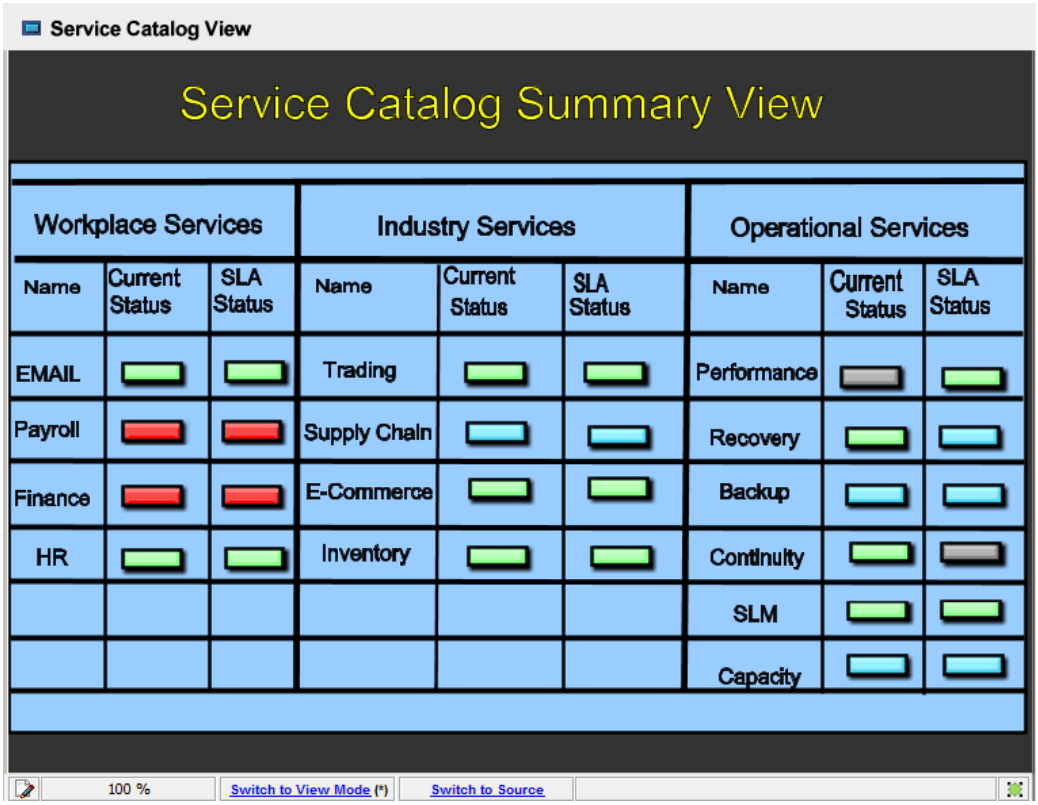


Notice that placing the mouse over any drawing object that is linked to an element displays a summary of the element condition. In this example, place the mouse over the city name or symbol.

8.2 Tabular Data View

The Layout view illustrated in Figure 8-2 provides a service catalog summary view. The colored rectangles help quickly identify the current status of individual services and the corresponding SLA health.

Figure 8-2 *Service Catalog Summary View*



Creating this type of view consists of:

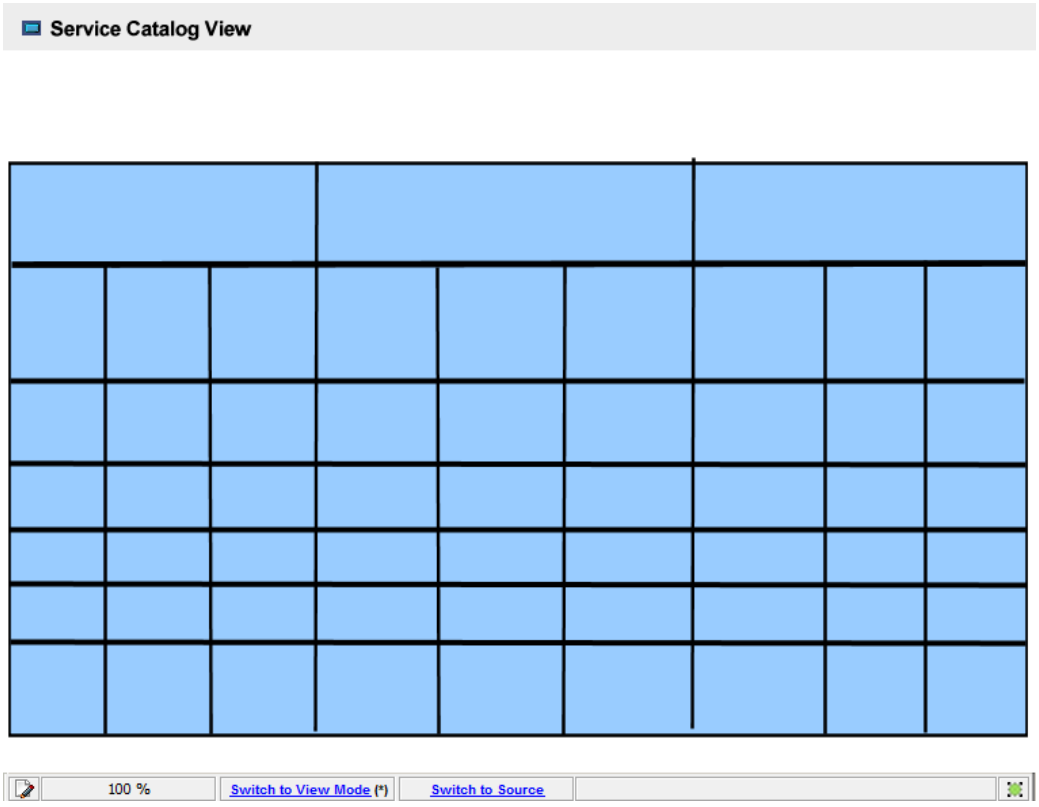
- Using drawing tools to create a grid for services and adding text
- Creating rectangles to represent the current conditions for individual services and their SLAs
- Dragging and dropping service and SLA elements to the appropriate rectangles, to bind the element conditions to the fill color of the rectangles

The following sections describe how to create the above view:

To create a grid diagram in the Layout view:

1 First we will draw the grid:

1a Draw a rectangle that has a blue fill as shown in the following:



1b Add lines to subdivide the grid into rows and columns.

2 Next we will finish the drawing components:

2a Add text to name the rows and columns.

2b Add a chart title at the top of the drawing.

2c Add rectangles to represent the element and SLA conditions.

2d Change the background color by right-clicking the background and selecting *Set Background Color*.

2e Drag and drop service elements and SLAs from the *Explorer* pane to the rectangles in the drawing.

By default, the element conditions bind to the rectangle fill colors.

3 The last step is associating the shapes in the grid to elements and updating color based on the element's condition state.

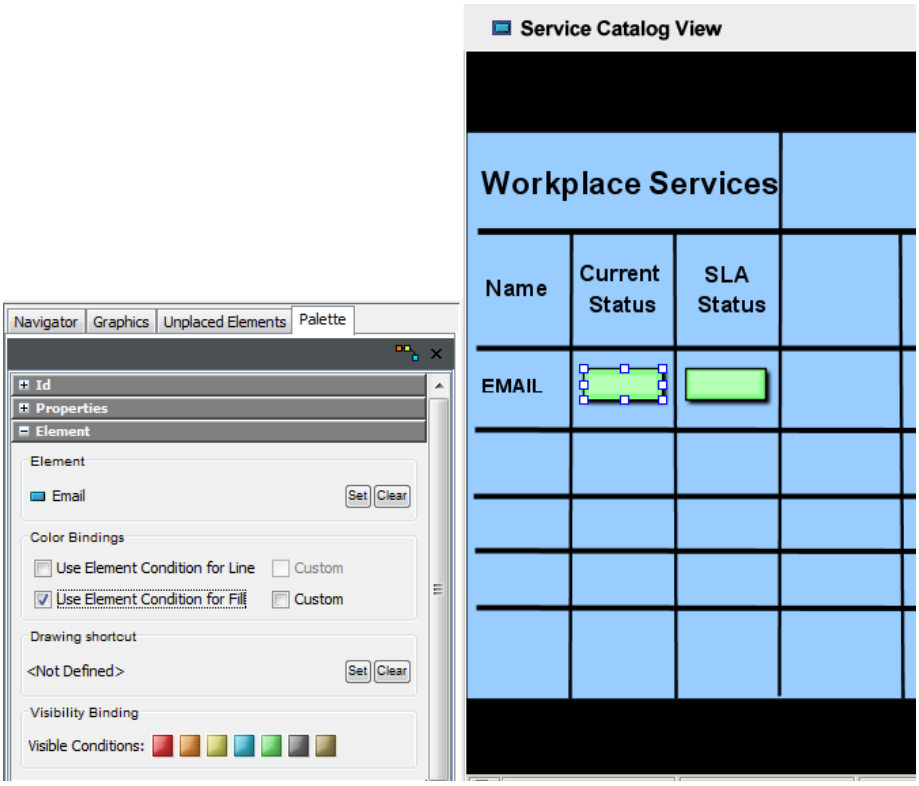
3a Drag an element from the *Explorer* pane to the appropriate rectangle in the drawing.

By default, the element condition color is used to fill the rectangle.

3b Drag elements from any branch of the *Explorer* pane.

For example, drag SLAs from the *SLA* branch and drag elements from the *Service Models* branch.

3c Confirm the correct element is used for binding by checking the Element name displayed in the *Element* property pane.

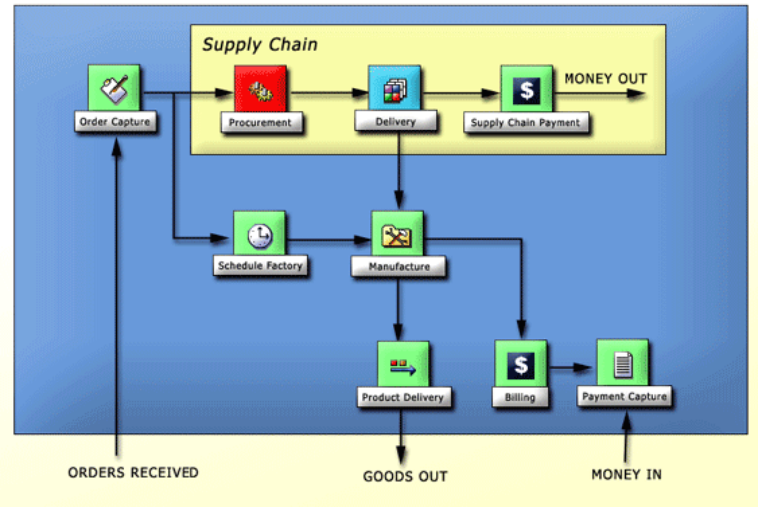


8.3 Business Process View Example

This Layout view illustrated in Figure 8-3 identifies key steps in a business process. Steps are linked to elements so that the fill color use to represent each step corresponds to the current element state. End-to-end service management enables monitoring and controlling service problems from a single console view. The drawing integrates all existing sources of infrastructure data into a single place, easing the problem of diagnosis where cause and effect are separate.

Figure 8-3 shows the business processes that support order processing. Color-coding reflects each process' state — red for CRITICAL, green for OK, and so on. IT staffers can right-click a process in trouble and immediately see what technology issue is causing the problem and what other technology and business areas that issue is impacting.

Figure 8-3 *Completed Supply Chain Business Process Layout View*



Creating this example requires:

- Changing the background color
- Dragging and dropping elements from different branches of the *Elements* hierarchy in the *Explorer* pane to the Layout view
- Changing the node style
- Adding a rectangle to highlight specific elements
- Adding heading/descriptive text
- Adding lines and arrows to illustrate connection between drawing elements

Start by changing the background color and adding a filled rectangle to draw attention to a specific set of steps:

1 Change the background color of the business process view:

1a Right-click the background, then select *Set Background Color*.

1b Select a color from the palette.

In this example, a blue background is used.

2 Add elements to the business process view:

2a Drag and drop elements from the *Explorer* pane to create the correct sequence of steps in the business process:



2b When prompted, select to a placement action for the element.

By default, the dropped elements use the Gradient Bubble node style.

For details on the menu items, see Chapter 5, "Adding Elements and Updating Drawing Components Using Element Properties," on page 39.

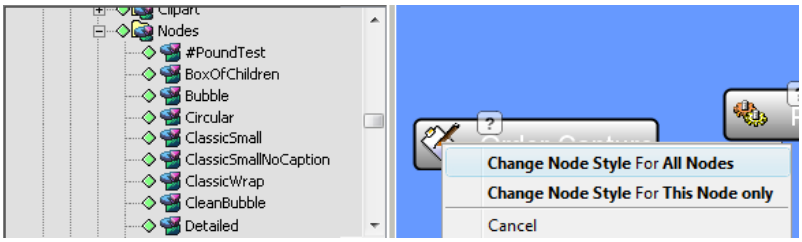
3 Change the node style for all elements in the drawing.

By default, the Bubble Gradient node style is used. In this application, a smaller node style is preferable, to enable displaying the maximum number of elements on the screen at one time. In this example, the Classic Small node style is used.

3a In the *Explorer* pane, expand *Administration* > *Graphics* > *Nodes*.

3b Drag the Classic Small node style from the *Explorer* pane and drop it on an element in the drawing.

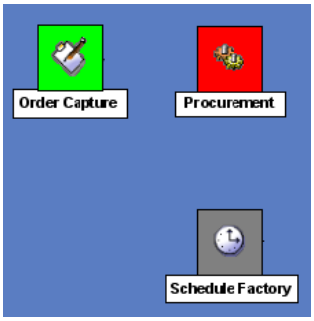
The following illustrates changing all node styles to ClassicSmall:



3c From the menu, select *Change Node Style for All Nodes*.

All elements in the drawing update to use the new node style.

The following illustrates elements using the *ClassicSmall* node style:



4 Add a rectangle to highlight specific elements in the Supply Chain process after completing all drag and drop actions. Otherwise, if you drop an element onto a shape, such as a rectangle, the element is bound to the shape.

4a Click  (*Rectangle*) on the drawing toolbar.

4b Drag to draw an outline of the shape anyplace in the view, then release the mouse button to display the finished shape.

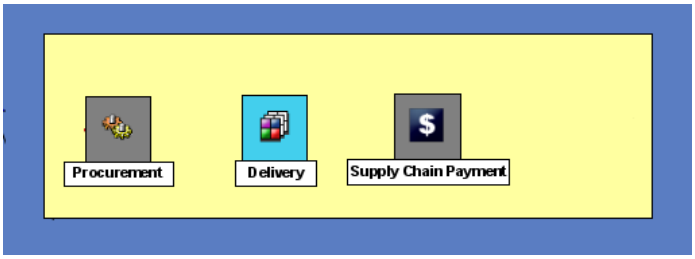
4c Drag the shape to the correct location in the drawing.

4d If it is necessary to modify the size, click and drag any of the shape's endpoints.

4e Select the rectangle, then click the appropriate fill color in the Palette.

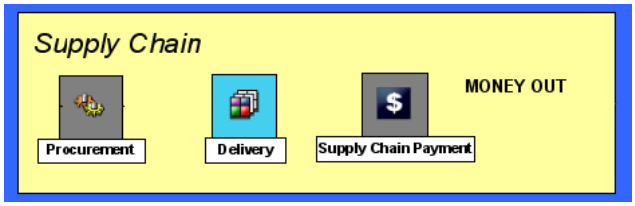
4f If the rectangle displays on top of the elements, thereby obscuring them, change the display's order by right-clicking the rectangle and selecting *Order* > *Send to Bottom*.

The rectangle moves beneath the elements:



5 Add descriptive text.

Use the  Text tool to add descriptive text to the drawing. This text is not bound to any elements.

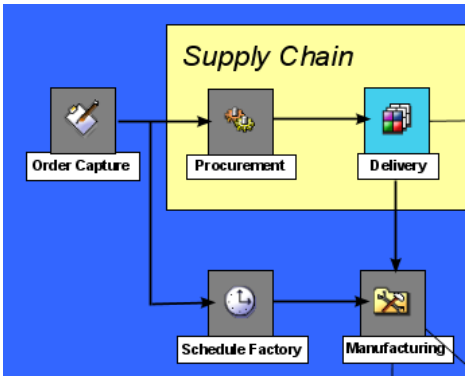


6 As a final step, use the drawing toolbar to add arrows showing the relationships among the steps in the business process.

6a Click  (*Line*).

6b Click to start and drag to draw the line.

6c Release the mouse to finish drawing:



The above illustrates that you can add arrows to show the process flow.

To change the line width or add arrow heads use the options in the drawing palette, see Section 3.1, “Understanding the Layout View Panels and Drawing Palette,” on page 20.

A Rendering Rule Precedence for Node Styles and Graphics

This section describes the lookup strategy used to determine the node style or custom graphic rendering used for an element in a Layout view drawing.

It also provides examples using a series of step-by-step instructions. Re-create the examples to understand how the node style/graphic representation works in the Layout view. Considering the numerous ways a node style can be applied to elements, it could be unclear which settings or overrides take precedence. Read through the section, then try the example steps to see first-hand how the node style settings and priorities work.

It is assumed the reader is a proficient user of Operations Center and the Layout view in particular. Many steps in this section are general in nature and it is assumed the reader knows how to navigate the Layout view and can create drawings and portals.

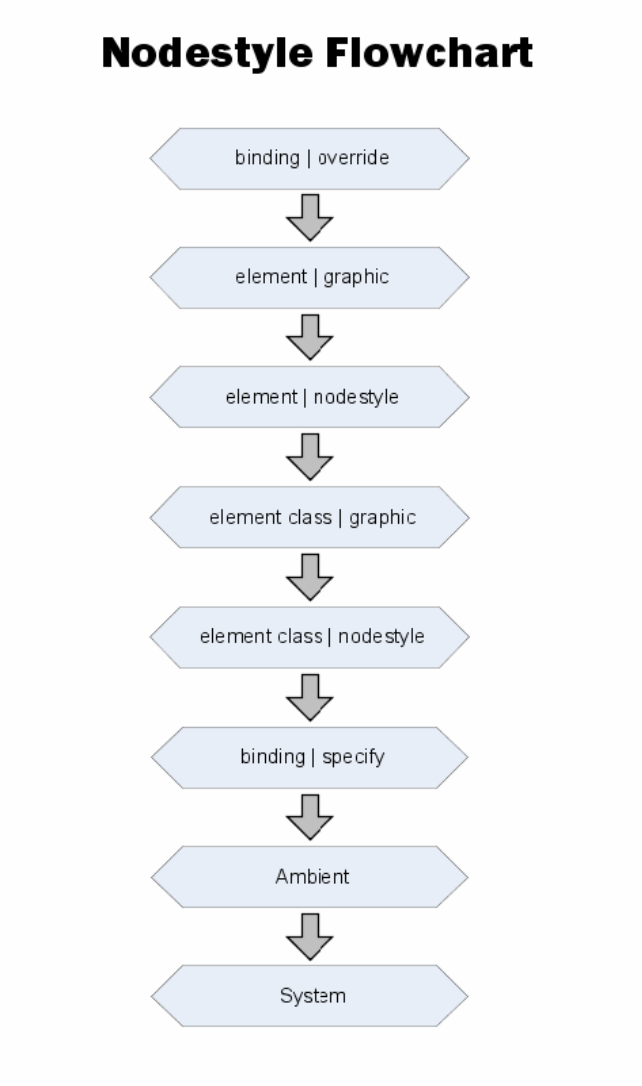
- Section A.1, “Element Rendering Look Up Strategy,” on page 106
- Section A.2, “Creating a Test Model,” on page 108
- Section A.3, “Viewing the Layout in the Dashboard,” on page 112
- Section A.4, “Creating Node Styles for Testing,” on page 112
- Section A.5, “Applying the Ambient Node Style to the Drawing,” on page 113
- Section A.6, “Changing the Node Style through Binding,” on page 114
- Section A.7, “Applying a Node Style to a Class,” on page 116
- Section A.8, “Applying a Graphic to a Class,” on page 117

A.1 Element Rendering Look Up Strategy

In the context of the Operations Center Layout view, element rendering is performed by locating the selected node style or graphic representation for a specific element in a specific drawing.

This property can be set in many places. A default lookup strategy determines which node style or graphic rendering to use for an element in a drawing. Referring to Figure A-1, the lookup works from top to bottom:

Figure A-1 *Nodestyle Flowchart*



The lookup strategy is performed in the following order:

1. `binding | override (* DRAWING SPECIFIC)`

   Use this setting to override all other node styles and graphics in a drawing and assign a specific node style for a specific element.

2. `element | graphic`

   An element in a drawing has a custom graphic defined for this instance of the element. This graphic is specified in the element's Element Graphic drawing channel. This graphic might have been created by a user in v4.0, or it could result from a conversion from v3.5.*x* for an *Organization* element that contained a custom graphic.

3. `element | node style`

   An element in a drawing has a custom node style defined for this instance of the element. This graphic is specified in the Element Node Style drawing channel. This channel is merely a reference to a custom node style listed under *Administration* > *Graphics* > *Node Style*.

4. `element class | graphic`

   An element in a drawing has a class definition with a custom graphic defined. This graphic is specified in the Element Graphic drawing channel for this instance of the class definition, which appears in the *Metamodel* branch of the *Administration* hierarchy.

   IMPORTANT: There is currently an open issue with these two `element class` options. Only `Org` type elements support these settings. This is a constraint put in place by the Metamodel implementation.

   ---

5. `element class | node style`

   An element in a drawing has class definition with a custom node style defined. This is specified in the Element Node Style drawing channel for this instance of the class definition, which appears in the *Metamodel* branch of the *Administration* hierarchy.

6. `binding | specify ( * DRAWING SPECIFIC)`

   An element was added to a drawing using the *Automatic Layout* option. The user chose to set a node style for the all elements generated by this layout option.

   TIP: This setting can be overridden by the higher priority settings that are found for each element in the binding.

   ---

7. `Ambient ( * DRAWING SPECIFIC)`

   The Ambient node style changes all elements that currently use the system default node style. It is a way to quickly change multiple nodes at once. Drag and drop a node style to the drawing background.

8. `System`

   There is no node style set on a drawing and no node style or graphic found for the target element by searching the list above.

The subsequent sections provide examples of how the element rendering priorities work. Start by creating custom classes and apply them to new elements, then create a corresponding portal. You can create several node styles and apply them.

A.2 Creating a Test Model

This example that walks you through creating a new Metamodel class and applying the class to a set of elements. It also shows you how to manually bind an element to the drawing.

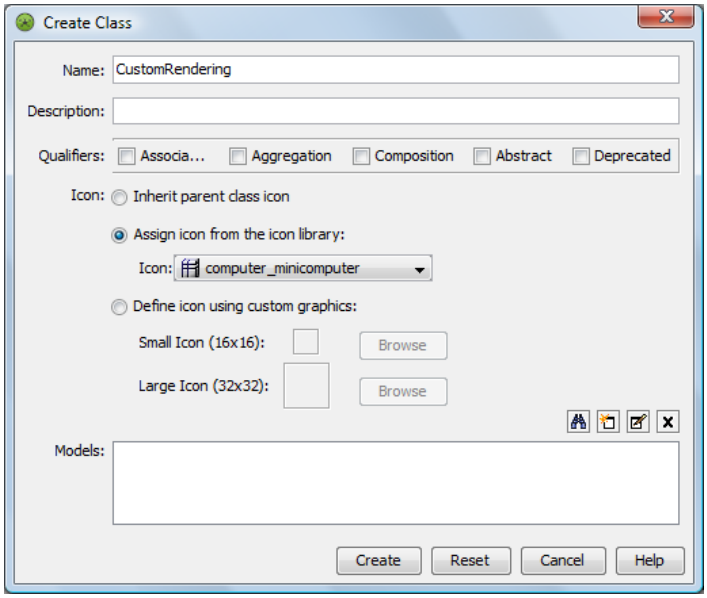
- Section A.2.1, “Creating a New Metamodel Class and Elements,” on page 108
- Section A.2.2, “Creating a Manually Bound Element in the Drawing,” on page 111

A.2.1 Creating a New Metamodel Class and Elements

To create a new Metamodel class and elements that use this class:

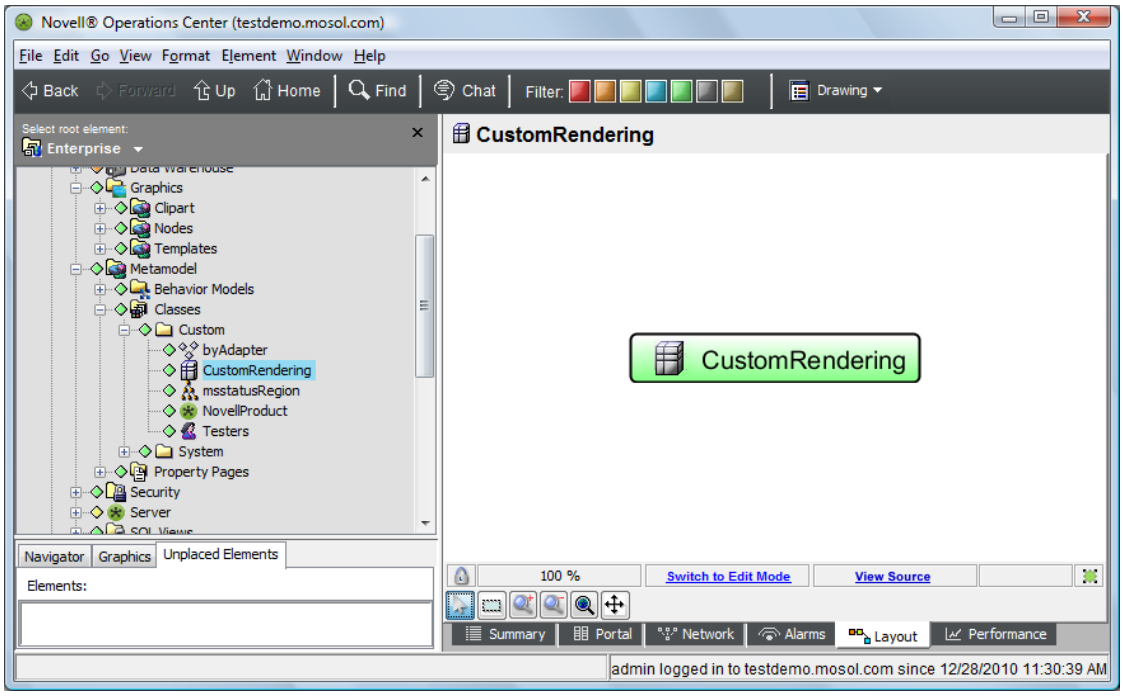
1 In the *Explorer* pane, under *Administration* > *Metamodel* > *Classes*, create a new class named *CustomRendering*.

This class does not require changing any Metamodel settings.



2 Select this new class in the *Explorer* pane, then switch to the Layout view.

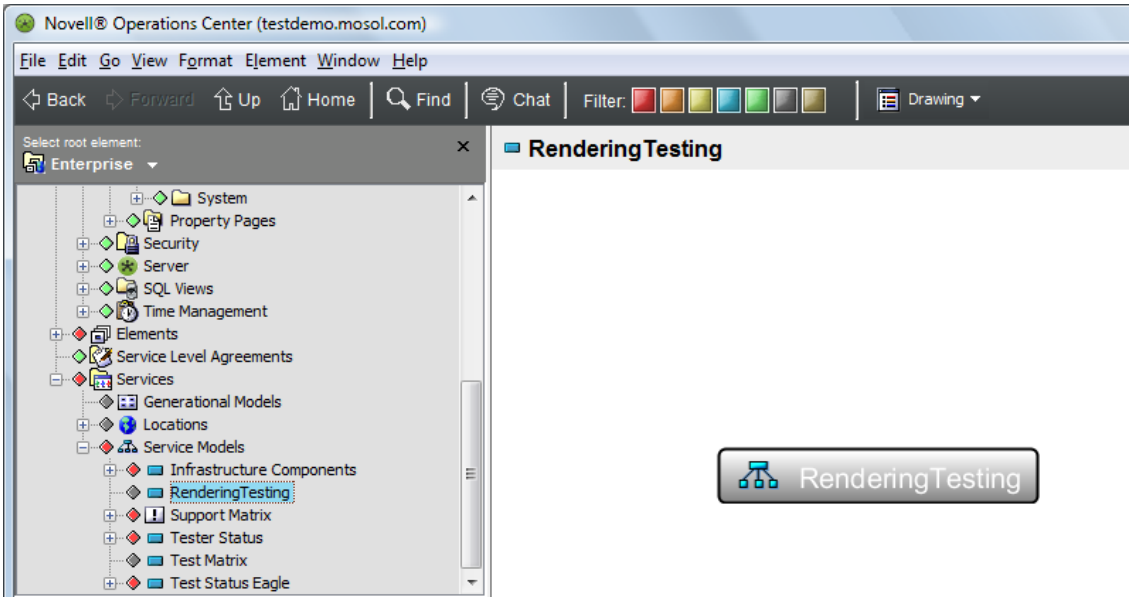
The view should resemble the following:



3 Create a new *Service Model* element named *RenderingTesting*.

4 In the *Explorer* pane, select *RenderingTesting*.

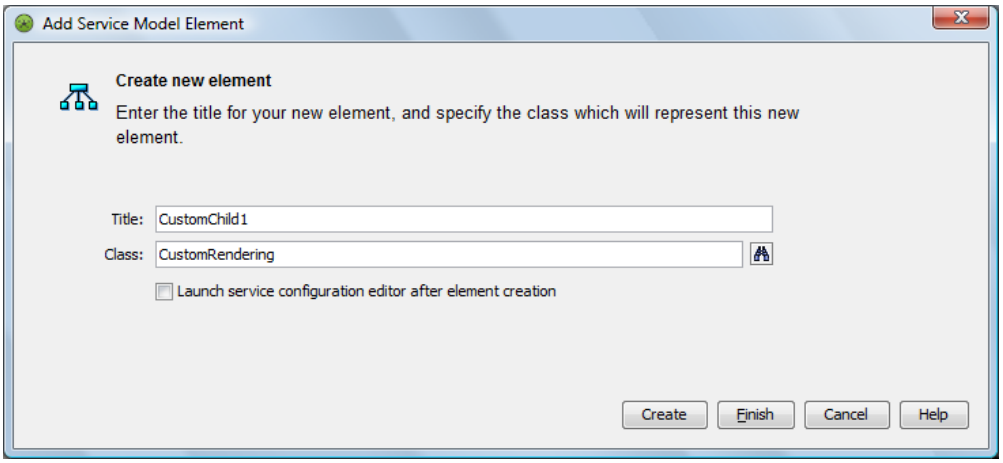
Notice that the default node style used to represent the element is *Gradient Bubble* in the Layout view:



5 Create three child elements under *RenderingTesting* named:

- *CustomChild1*

  In the *Class* field, select the custom class *CustomRendering* (defined earlier in Step 1 on page 108):



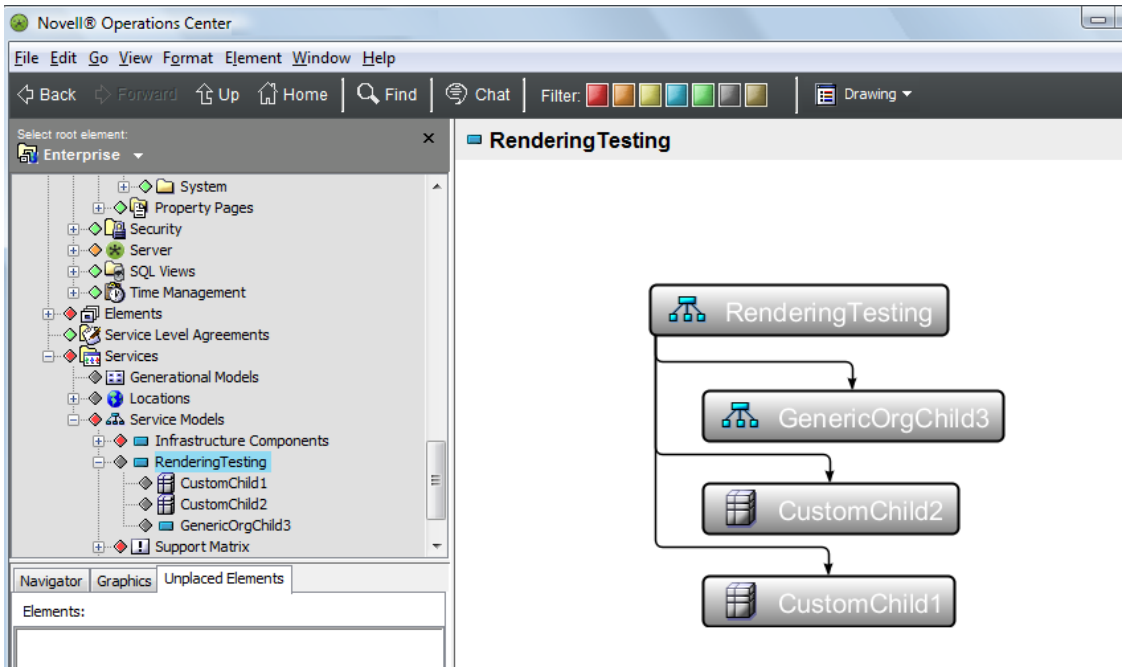
- *CustomChild2*

  In the *Class* field, select the custom class *CustomRendering*.

- *GenericOrgChild3*

  In the *Class* field, leave the default *Org* class.

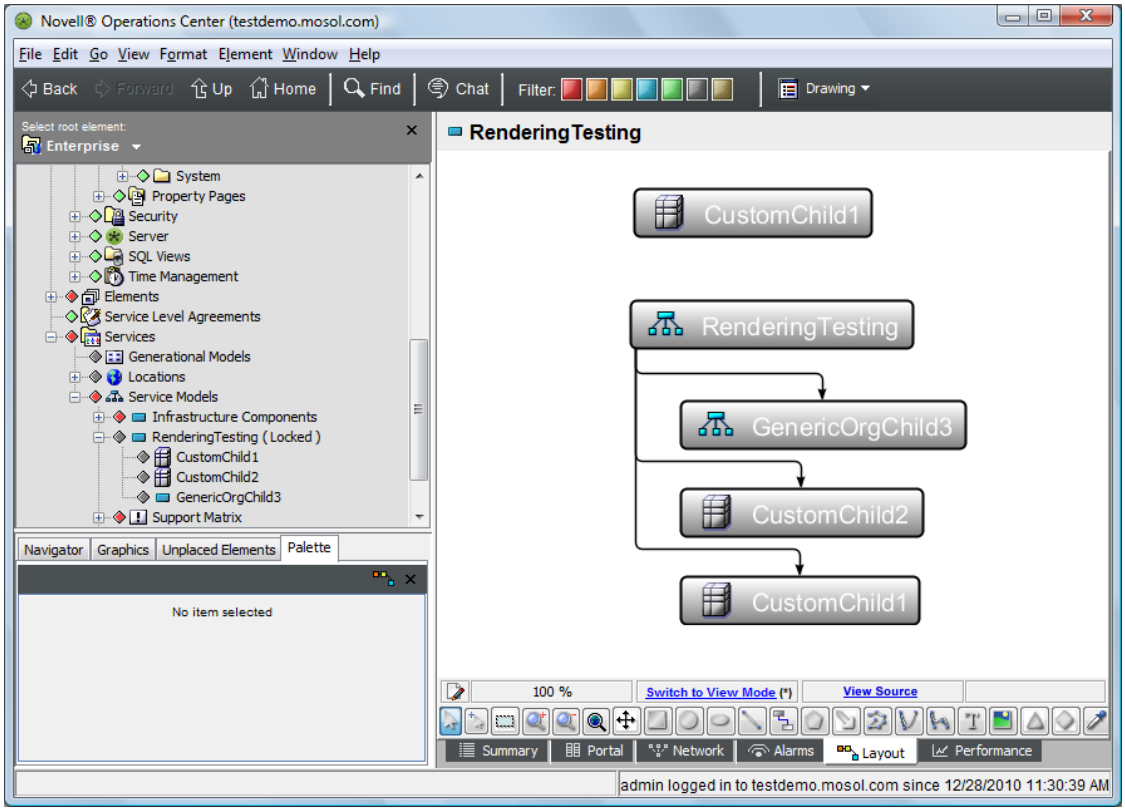
The resulting Layout view should resemble the following:



A.2.2 Creating a Manually Bound Element in the Drawing

To create a manually bound element inside a drawing:

1. In the Layout view, switch to Edit mode.
2. Drag and drop *CustomChild1* from the *Explorer* pane to the background of the drawing.
   Make sure you drop the element the background and not on any of the existing elements.
3. Select *Add to Drawing Only* from the pop-up menu.
   The resulting drawing resembles the following:

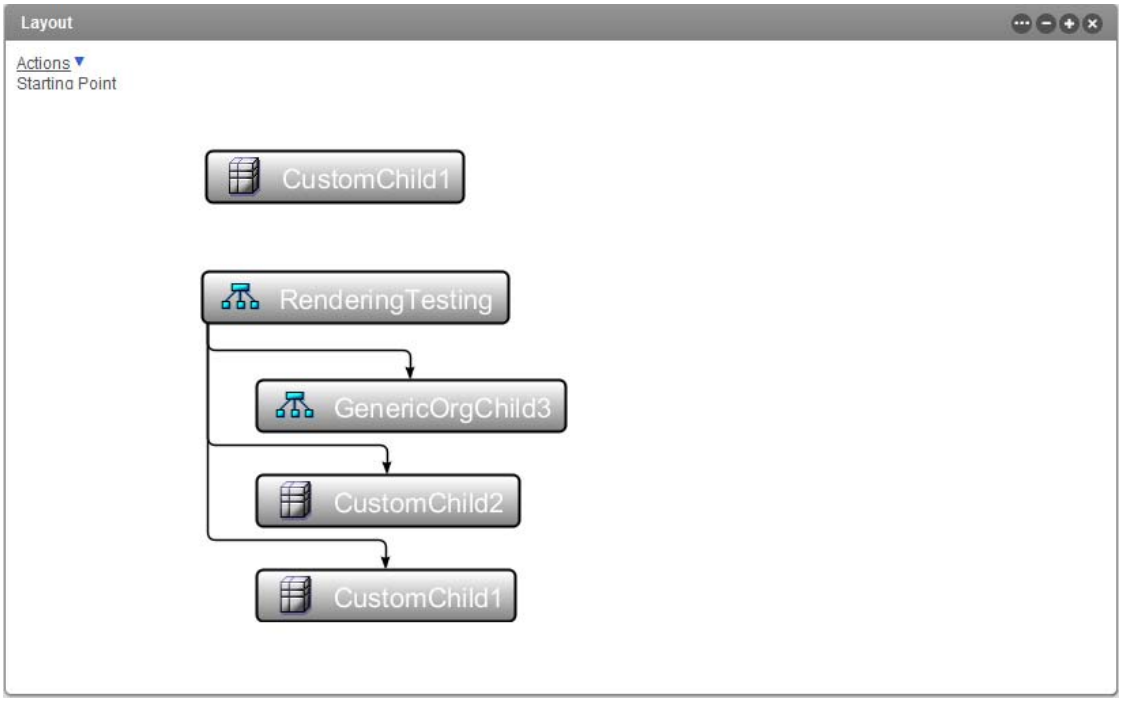


A.3 Viewing the Layout in the Dashboard

To view the layout in the Dashboard:

1. Create a default portal for the *RenderingTesting* element.

   The rendered portal should look identical to the Layout drawing shown here:



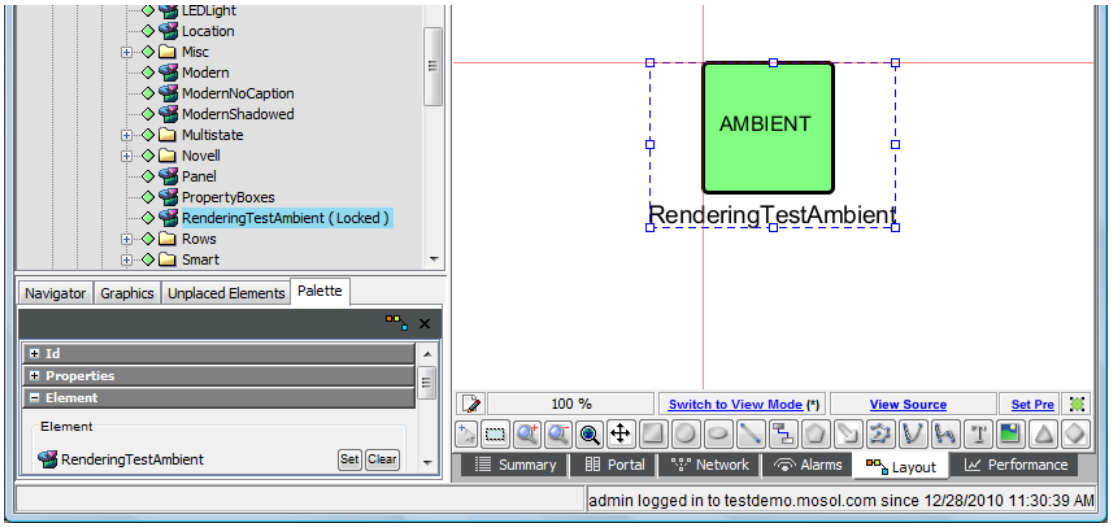
2. Deselect the *Use Relationships* check box in the Layout portlet so that the SVG layout is used.

A.4 Creating Node Styles for Testing

To create node styles for testing, do the following:

1. Under *Administration* > *Graphics* > *Nodes*, create a node style named *RenderingTestAmbient*.
2. Edit the node style as follows:
   - Right-click the node style graphic, then select *Ungroup*.
   - Change the `<default>` text to AMBIENT.
   - Add a text label beneath the square and bind the label to the *Element Name* option.
   - Select everything in the drawing, then right-click *Group*.
   - Save the drawing.

The following illustrates the resulting drawing:



3 Create a node style named *RenderingTestClass* and edit it as follows:

3a Change the `<default>` text to CLASS.

3b Add a text label beneath the square and bind that label to the *Element Name* option.

4 Create a node style named *RenderingTestElement* and edit it as follows:

4a Change the `<default>` text to ELEMENT.

4b Add a text label beneath the square and bind that label to the *Element Name* option.

5 Create a node style named *RenderingTestOverride* and edit it as follows:

5a Change the `<default>` text to OVERRIDE.

5b Add a text label beneath the square and bind that label to the *Element Name* option.

6 Create a node style named *RenderingTestSpecify* and edit it as follows:

6a Change the `<default>` text to SPECIFY.

6b Add a text label beneath the square and bind that label to the *Element Name* option.

A.5 Applying the Ambient Node Style to the Drawing

The Ambient node style changes all elements that currently use the system default node style. It is a way to quickly change multiple nodes at once. Drag and drop a node style to the drawing background.

To apply the Ambient node style to all elements in a drawing:

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

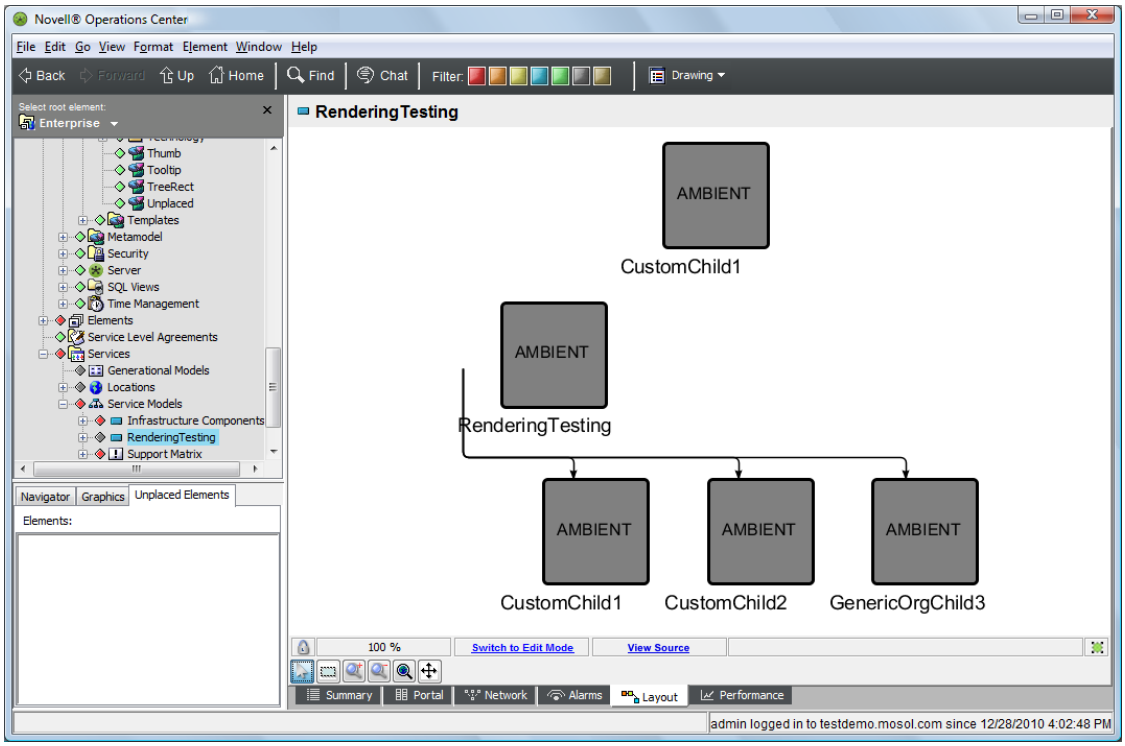
At this point, no node style has been set relative to the drawing so the system default of Gradient Bubble is used.

2 Under *Administration* > *Graphics* > *Node Styles*, drag and drop the *RenderingTestAmbient* node style to the background of the drawing.

Be sure to drop on the background and not on an element in the drawing.

3 When prompted, click *Yes* to set the Ambient node style.

4 Verify that the drawing resembles the following:



5 Save the drawing.

6 Refresh the associated Operations Center dashboard page and verify the same view displays.

A.6 Changing the Node Style through Binding

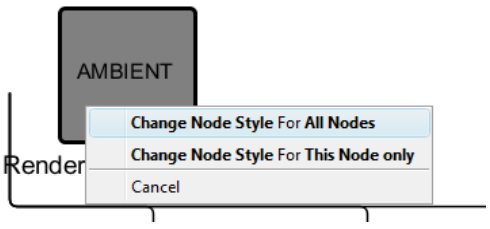
At this point, the Ambient node style has been set in the drawing. Now, you will change the node style for a specific element, then when prompted, change the node style for all nodes in the drawing.

The following example applies the `binding | specify` switch.

To update the node style for all drawing nodes:

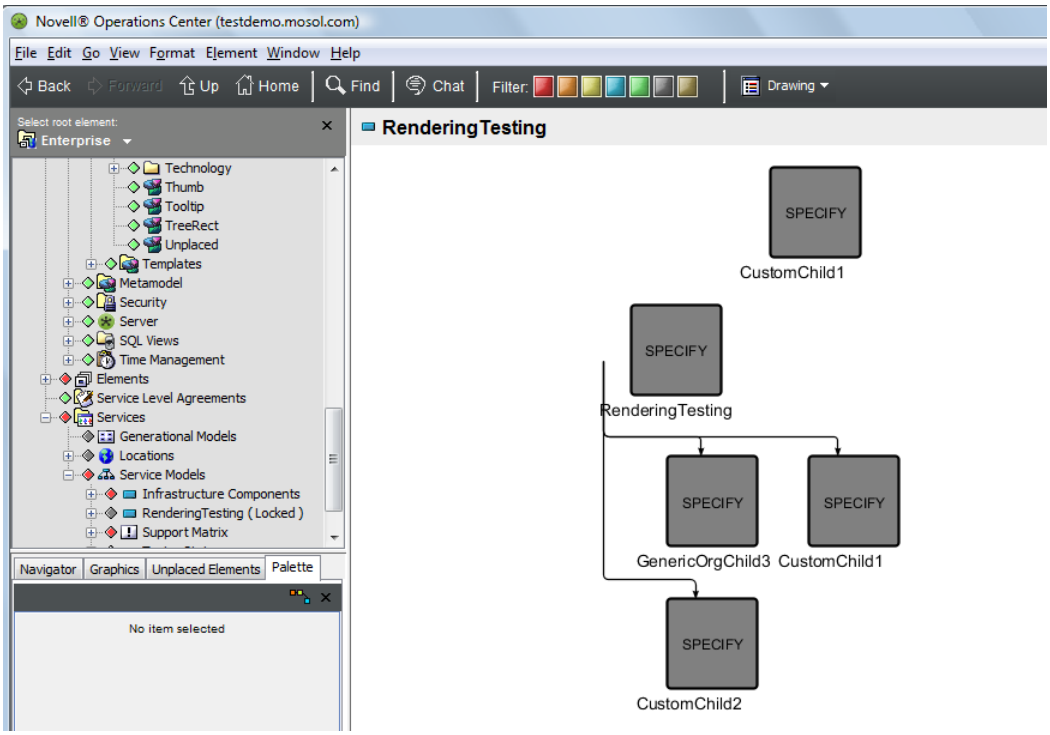
1 In the *Explorer* pane, under *Administration* > *Graphics* > *Node Styles*, drag and drop the *RenderingTestSpecify* node style to the *RenderingTesting* element in the drawing.

2 Select *All Nodes* from the menu.



The following results occur:

- The binding node style is set for the binding tree that starts with *RenderingTesting* as the root.
- The Ambient node style also changes, as *All Nodes* was selected.
- Verify that the drawing resembles the figure below. The manually-bound CustomChild1 is also rendered using the SPECIFY node style because the AMBIENT node style was changed when *All Nodes* was selected.



3 Save the drawing.

4 Refresh the Operations Center dashboard page and verify the view above displays.

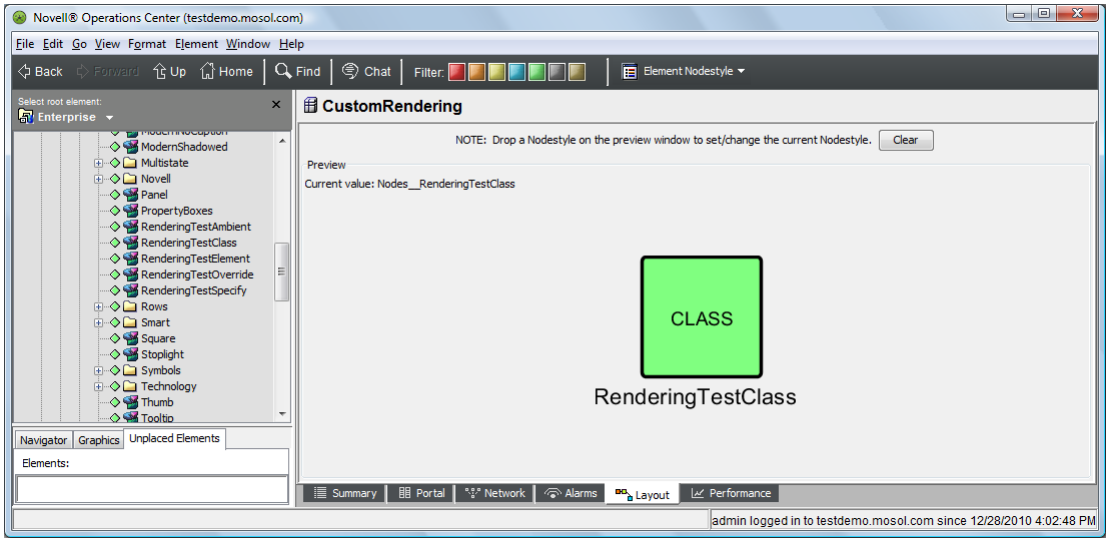
A.7 Applying a Node Style to a Class

The following section of the example applies a node style to a class and tests the `element class | node style` switch.

To apply a node style to a class:

1. In the *Explorer* pane, under the *Metamodel* hierarchy, select the *CustomRendering* custom class.
2. On the console toolbar, switch to the *Element Nodestyle* drawing channel.
3. In the *Explorer* pane, under *Administration* > *Graphics* > *Node Styles*, drag and drop the *RenderingTestClass* node style to the Layout view.

   This sets the node style to *RenderingTestClass* for the *CustomRendering* element class.

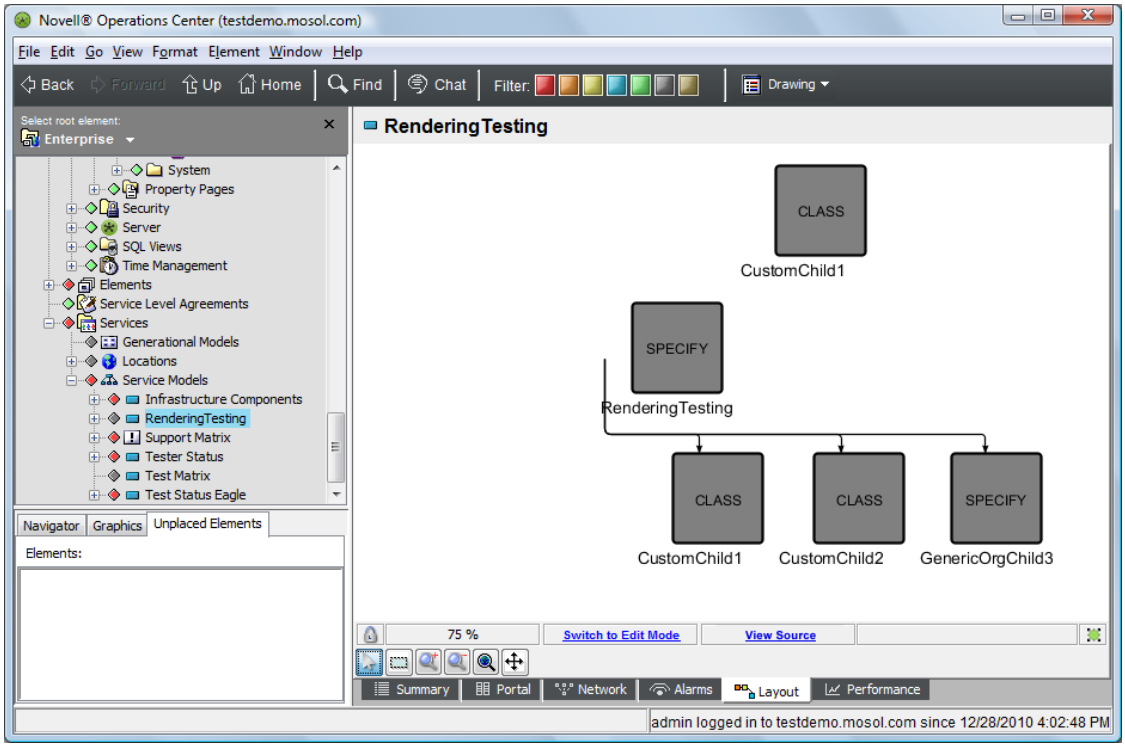


4 Change the view back to the *Drawing* channel.

5 In the *Explorer* pane, click *Service Models* > *RenderingTesting* and verify that all element instances of the *CustomRendering* custom class now use the *RenderingTestClass* node style.

This feature allows you to control node styles from the class level.

The view should resemble the following:



The elements whose class is *CustomRendering* are easily identified by having `CustomClass` in their names. Only these elements should use the *RenderingTestClass* node style that shows CLASS inside the node.

6 Refresh the Operations Center dashboard page and verify the same view displays.

A.8 Applying a Graphic to a Class

- Section A.8.1, “Changing the Class Node Style to a Custom Graphic,” on page 118
- Section A.8.2, “Applying a Node Style at the Element Instance Level,” on page 119
- Section A.8.3, “Applying a Graphic at the Element Instance Level,” on page 121
- Section A.8.4, “Overriding a Single Element Node Style,” on page 122
- Section A.8.5, “Clearing Previously Set Node Styles and Graphics,” on page 124

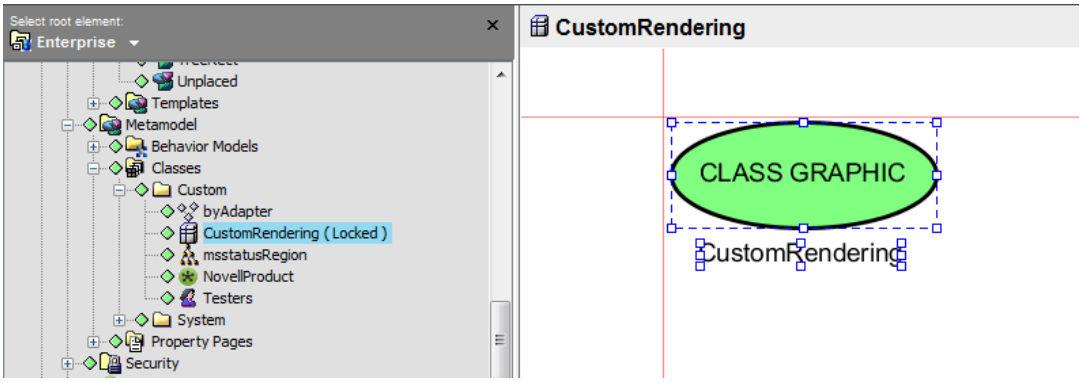
A.8.1 Changing the Class Node Style to a Custom Graphic

The following section of the example changes the node style of a class to a custom graphic and tests the `element class | graphic` switch.

To change a class node style to a custom graphic:

1. In the *Explorer* pane, under *Metamodel* > *Classes*, select the *CustomRendering* custom class.
2. On the console toolbar, switch to the *Element Graphic* drawing channel.
3. Add the following drawing elements to the drawing:
   - **3a** Draw an ellipse in the same location as the illustration below.

     It might be necessary to pan the drawing to see the red axes, because they initially display in the upper left-hand corner. Notice that the origin of the ellipse is relative to the red axes that represent the (0,0) coordinate origin of the drawing.
   - **3b** Bind the ellipse's fill color to element condition. The ellipse's fill changes to green.
   - **3c** Add a label named CLASS GRAPHIC inside the ellipse.
   - **3d** Add a label beneath the ellipse and bind that label to the *Element Name* option.

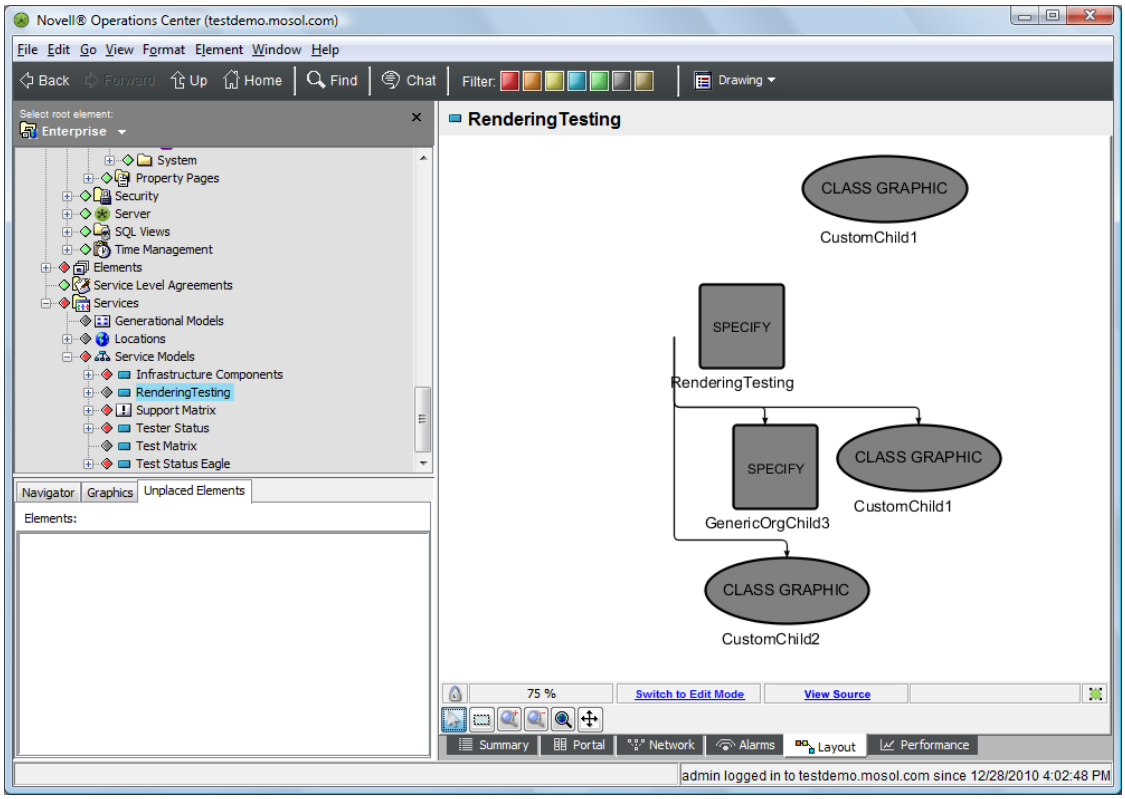


4. Save the drawing.

5 Return to the *RenderingTesting* element and verify that all element instances of the *CustomRendering* custom class now use the *CustomRendering* custom drawing graphic.

This feature allows the user to control custom graphics from the class level.

The drawing should resemble the following:



6 Refresh the Operations Center dashboard page and verify the same view displays.

A.8.2 Applying a Node Style at the Element Instance Level

The following example applies a node style at the element instance level and tests the `element | node style` switch.

To apply a node style at the element instance level:

1 In the *Explorer* pane, select the element *RenderingTesting* > *CustomChild1*.

2 Switch to the Element Node Style drawing channel.

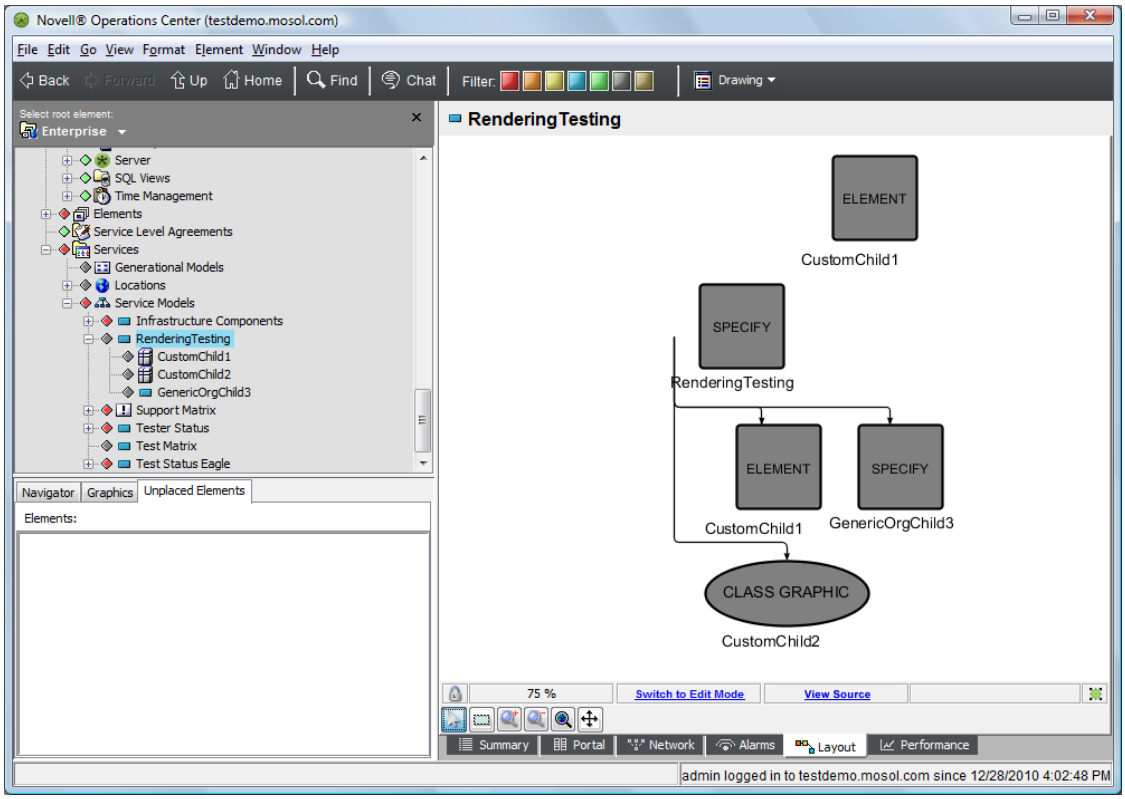
3 In the *Explorer* pane, under *Administration* > *Graphics* > *Node Styles*, drag and drop the *RenderingTestElement* node style to the Layout view.

This should set the node style to *RenderingTestElement* for the *CustomChild1* element class.

4 Return to the *RenderingTesting* view and verify that all copies of the *CustomChild1* element now use the *RenderingTestElement* node style.

This allows you to control node styles from the element instance level. In every drawing where this element appears, this graphic is used, unless it is overridden by a higher priority rendering.

The drawing should resemble the following:



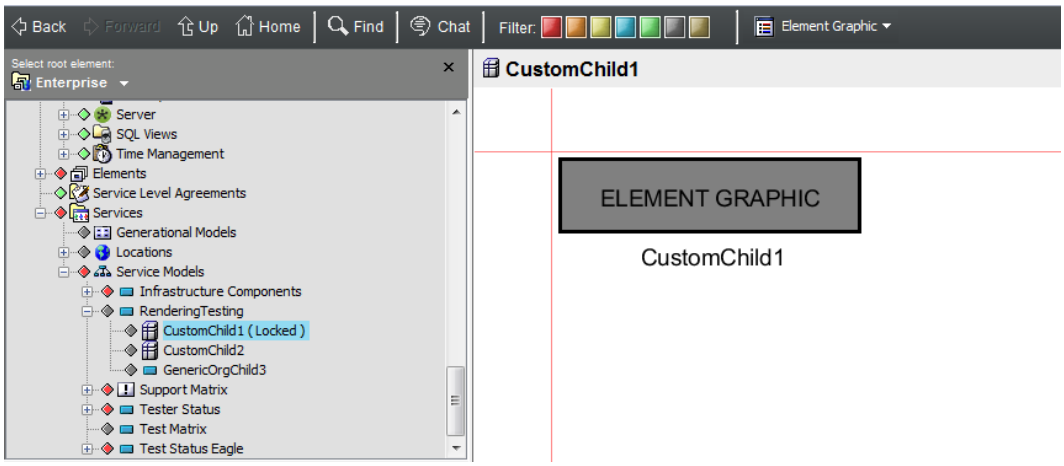
5 Refresh the Operations Center dashboard page and verify the same view displays.

A.8.3 Applying a Graphic at the Element Instance Level

The following section of the example applies a graphic at the element instance level and test the `element | graphic` switch.

To apply a graphic at the element instance level:

1. In the *Explorer* pane, select the *RenderingTest* > *CustomChild1* element.
2. Switch to the *Element Graphic* drawing channel.
3. Edit the drawing using the following steps:
   - **3a** Draw a rectangle in the same location as the following:



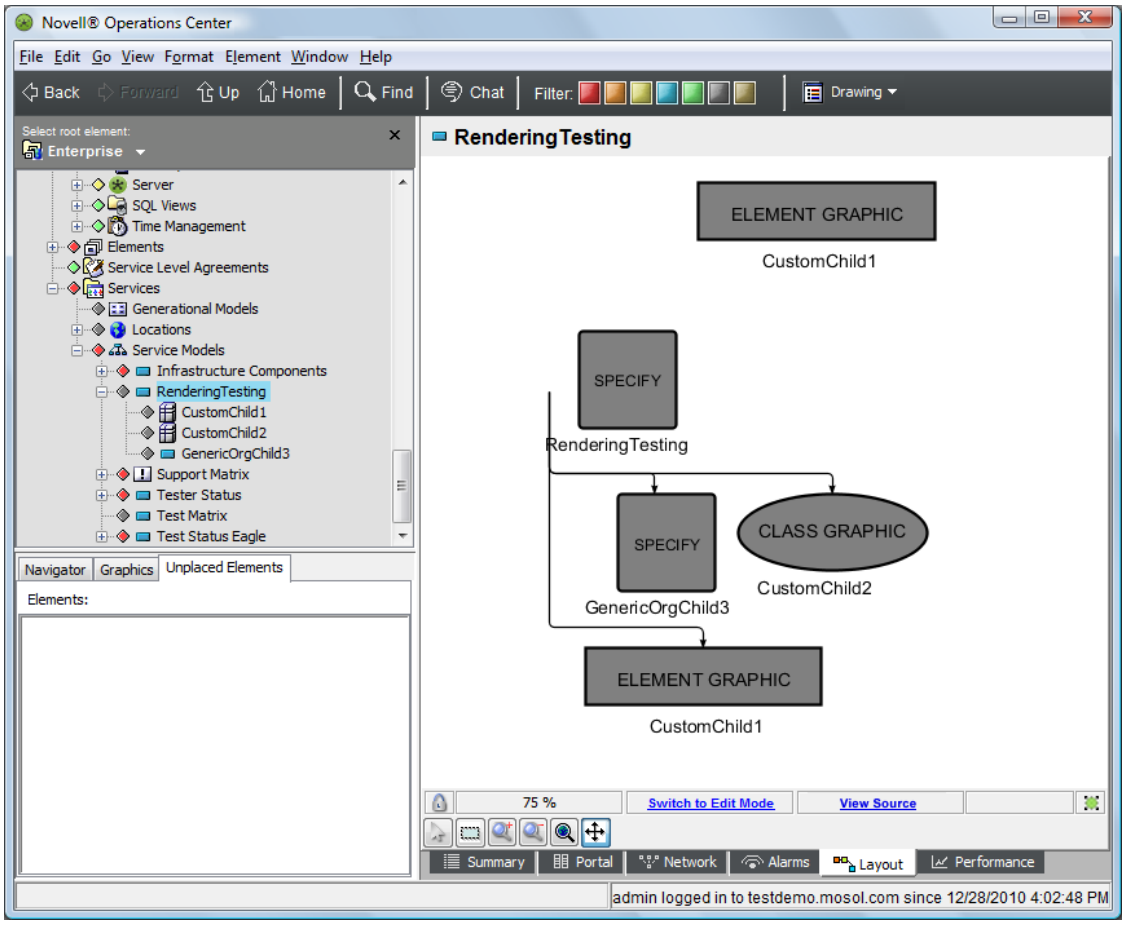
     It might be necessary to pan the drawing to see the red axes as they initially display in the upper left-hand corner. Notice how the origin of the rectangle is relative to the red axes that represent the (0,0) coordinate origin of the drawing.

   - **3b** Bind the rectangle's fill color to element condition.
   - **3c** Add a label named ELEMENT GRAPHIC inside the rectangle.
   - **3d** Add a label beneath the rectangle and bind that label to the *Element Name* option.
4. Save the drawing.
5. Return to the *RenderingTesting* view.

6 Verify that all copies of the *CustomChild1* element now use the *CustomChild1* custom graphic.

This allows you to control graphics from the element instance level. In every drawing where this element appears, this graphic is used, unless it is overridden by a higher priority rendering.

The drawing should resemble the following:



7 Refresh the Operations Center dashboard and verify the same view displays.

A.8.4 Overriding a Single Element Node Style

The following section of the example overrides the node style for a single element and tests the `bind | override` switch.

To override the node style for a single element:

1 In the *Explorer* pane, select the *RenderingTesting* element.

2 On console toolbar, verify the *Drawing* channel is selected.

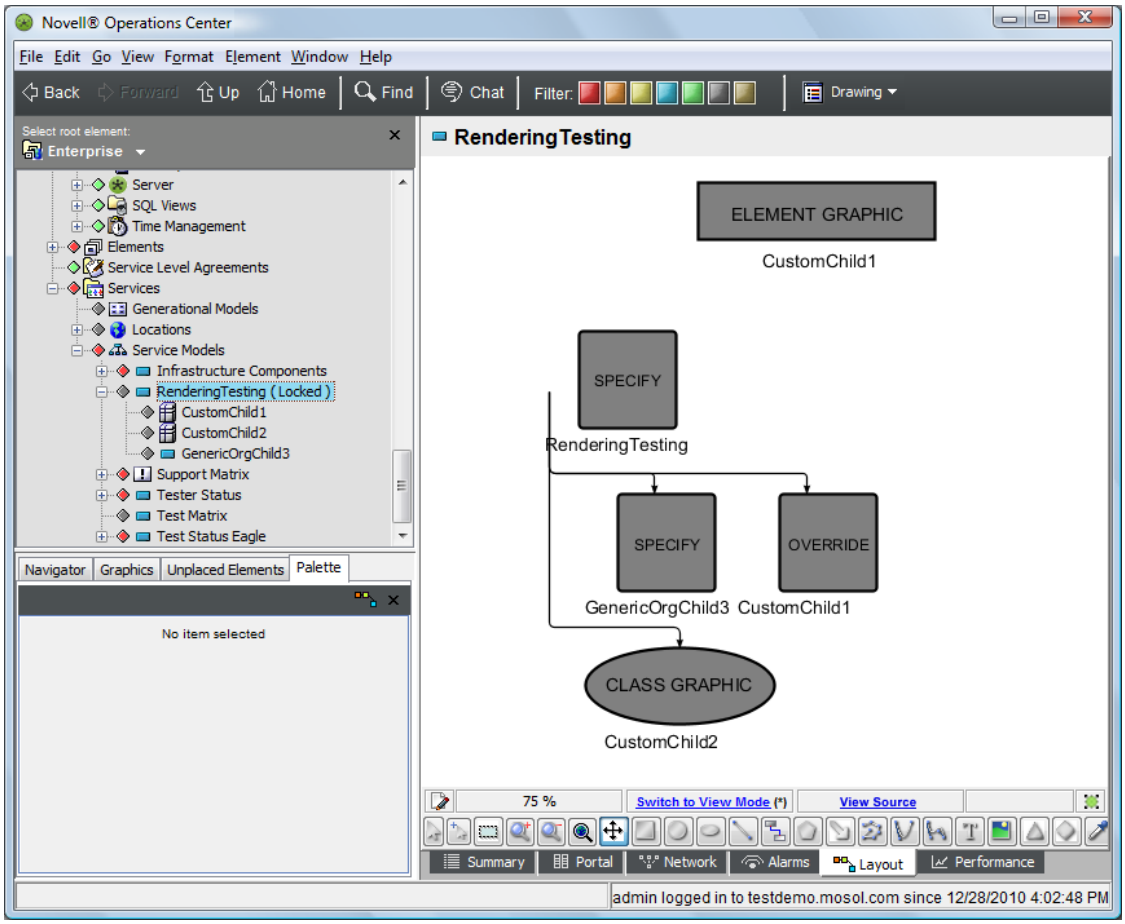
3 In the *Explorer* pane, under *Administration* > *Graphics* > *Node Styles*, drag and drop the *RenderingTestOverride* node style to the *CustomChild1* element in the Layout view.

The *CustomChild1* element is part of the nested tree layout.

4 Select *This Node Only* from the pop-up menu.

The node style changes to *RenderingTestOverride* for only one copy of *CustomChild1*. This override setting only applies to the binding that caused the element to display in the drawing. The other copy of *CustomChild1* still uses the custom element graphic set in the previous section.

The drawing should resemble the following:



5 Save the drawing.

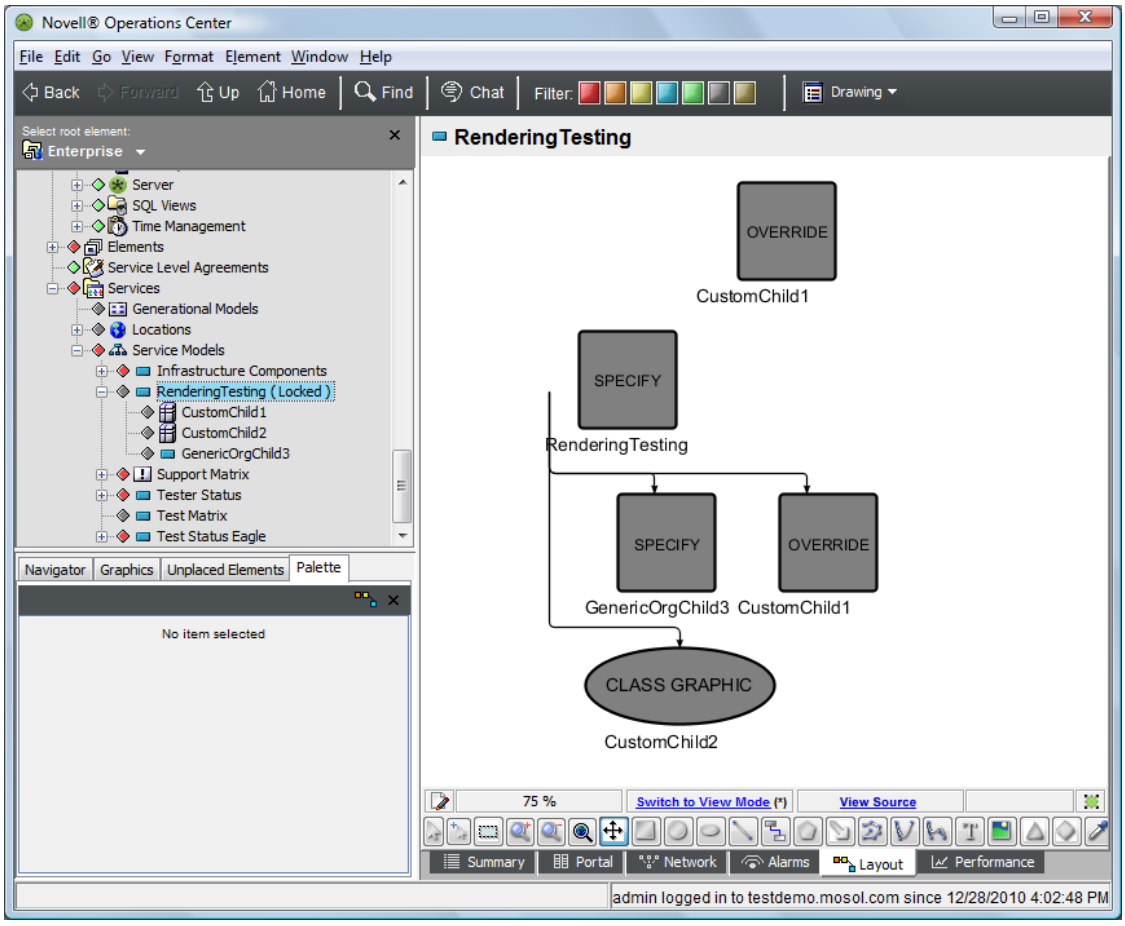
6 Refresh the Operations Center dashboard page and verify the same view displays.

7 While still viewing the *RenderingTesting* element, drag and drop the *RenderingTestOverride* node style to the orphaned *CustomChild1* element in the layout that is not part of the nested tree layout.

8 Select *This Node Only* from the pop-up menu.

This should set the node style to *RenderingTestOverride* for only this copy of the *CustomChild1* element in this drawing.

The drawing should resemble the following:



Now, each copy of the *CustomChild1* element has its own override rendering.

9 Save the drawing.

10 Refresh the associated portal and verify the same view displays.

A.8.5 Clearing Previously Set Node Styles and Graphics

The following sections cover how to clear the various types of node styles and graphic switches.

NOTE: There is currently no option in the Operations Center console for clearing the Ambient setting or the `binding|specify` setting. These settings can be cleared by directly editing the SVG source, which is explained in Appendix C, "Editing the Source SVG Code," on page 155.

- "Clearing the Bind | Override Switch" on page 125
- "Clearing the Element | Graphic Switch" on page 127
- "Clearing the Element | Node Style Switch" on page 128

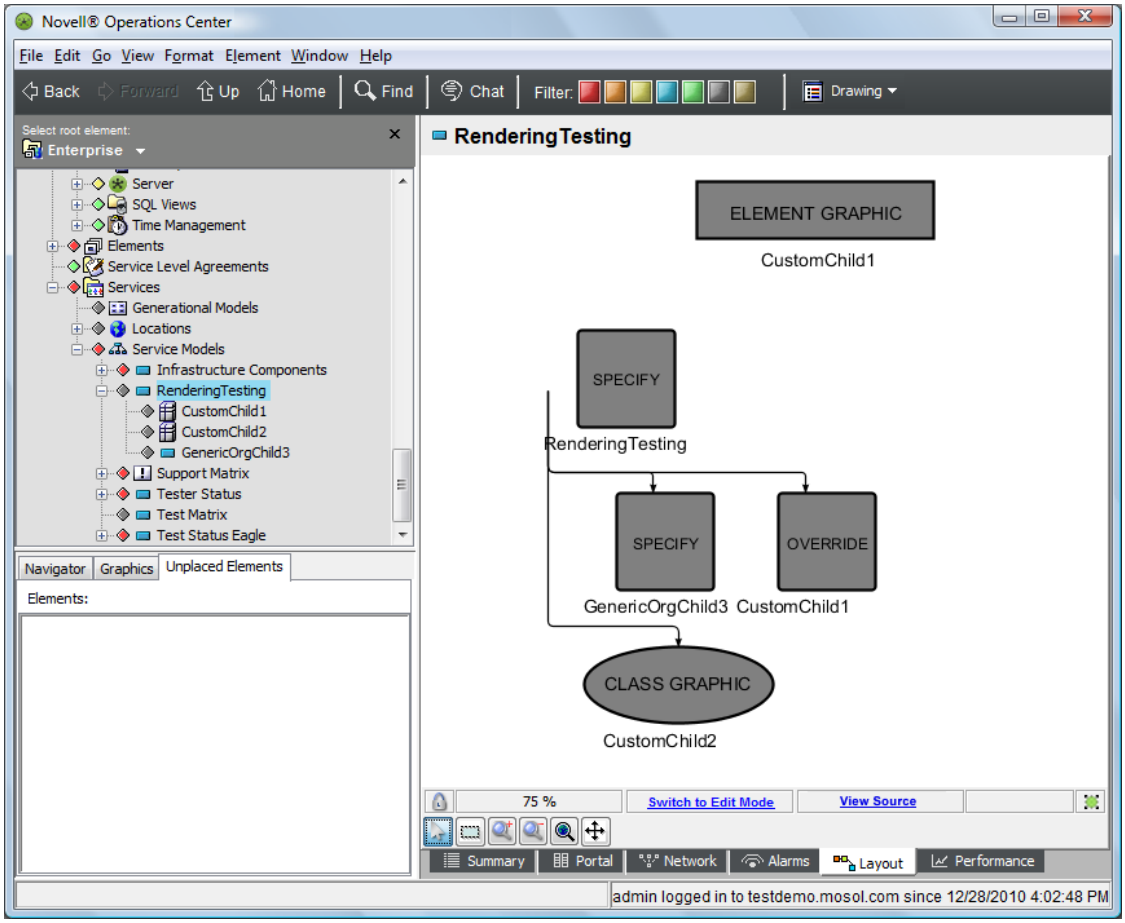
- "Clearing the Element Class | Graphic Switch" on page 129
- "Clearing the Element Class | Node Style Switch" on page 130

Clearing the Bind | Override Switch

To clear the `bind | override` switch:

1. In the *Explorer* pane, select the *RenderingTesting* element.
2. On the console toolbar, verify the *Drawing* channel is selected.
3. Switch to Edit mode.
4. Right-click the orphaned *CustomChild1* element in the layout that is not part of the nested tree layout.
5. Select *Remove Node Style Override* from the menu, then click *Yes* to remove the override.
6. Verify that the override is removed from only this copy of *CustomChild1*.

   The drawing should resemble the following:

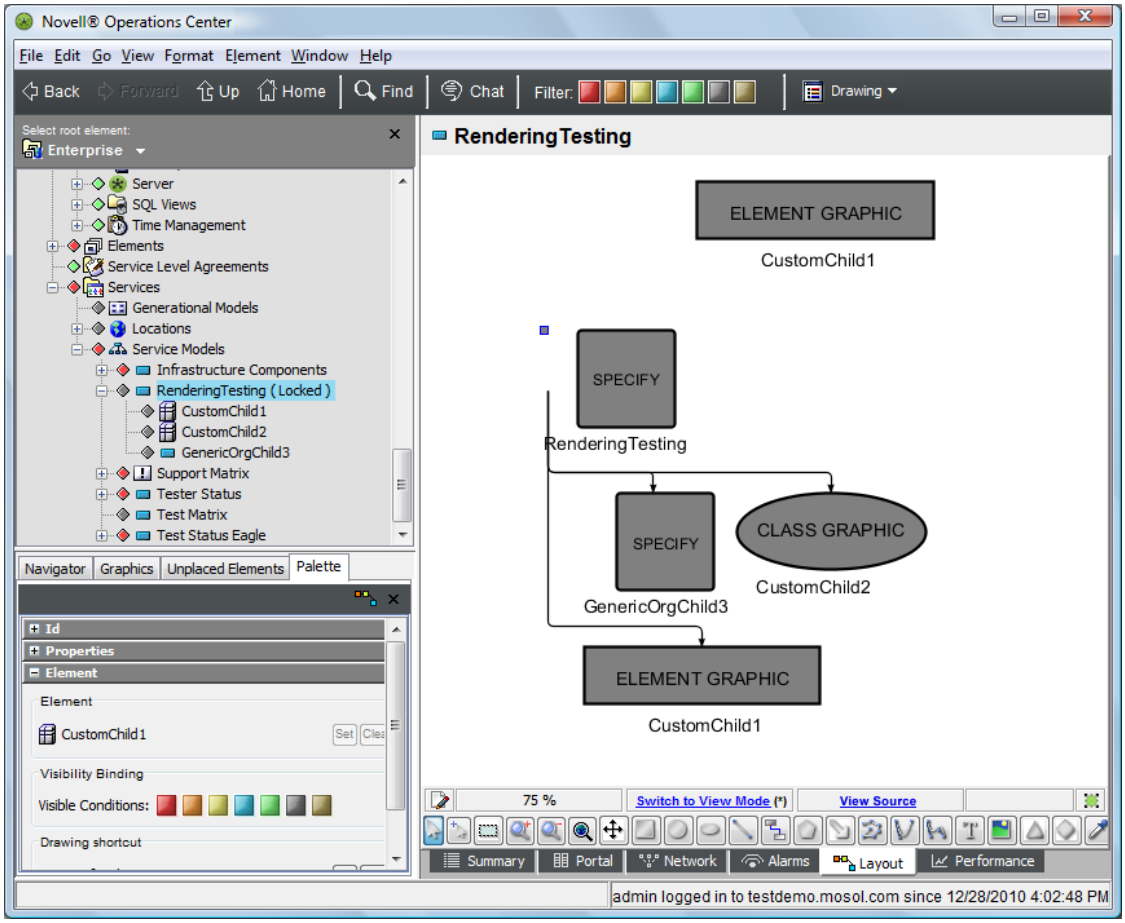


   There might be a short delay before the old rendering is cleared from the graphics cache.
7. Save the drawing.
8. Refresh the Operations Center dashboard page and verify the same view displays.
9. Switch to Edit mode and right-click the *CustomChild1* element that is part of the nested tree layout.

10 Select *Remove Node Style Override* from the menu, then click *Yes* to remove the override.

11 Verify that the override is removed from only this copy of CustomChild1.

There might be a short delay before the old rendering has been cleared from the graphics cache. The drawing should resemble the following:



12 Save the drawing.

13 Refresh the Operations Center dashboard page and verify the same view displays.

Clearing the Element | Graphic Switch

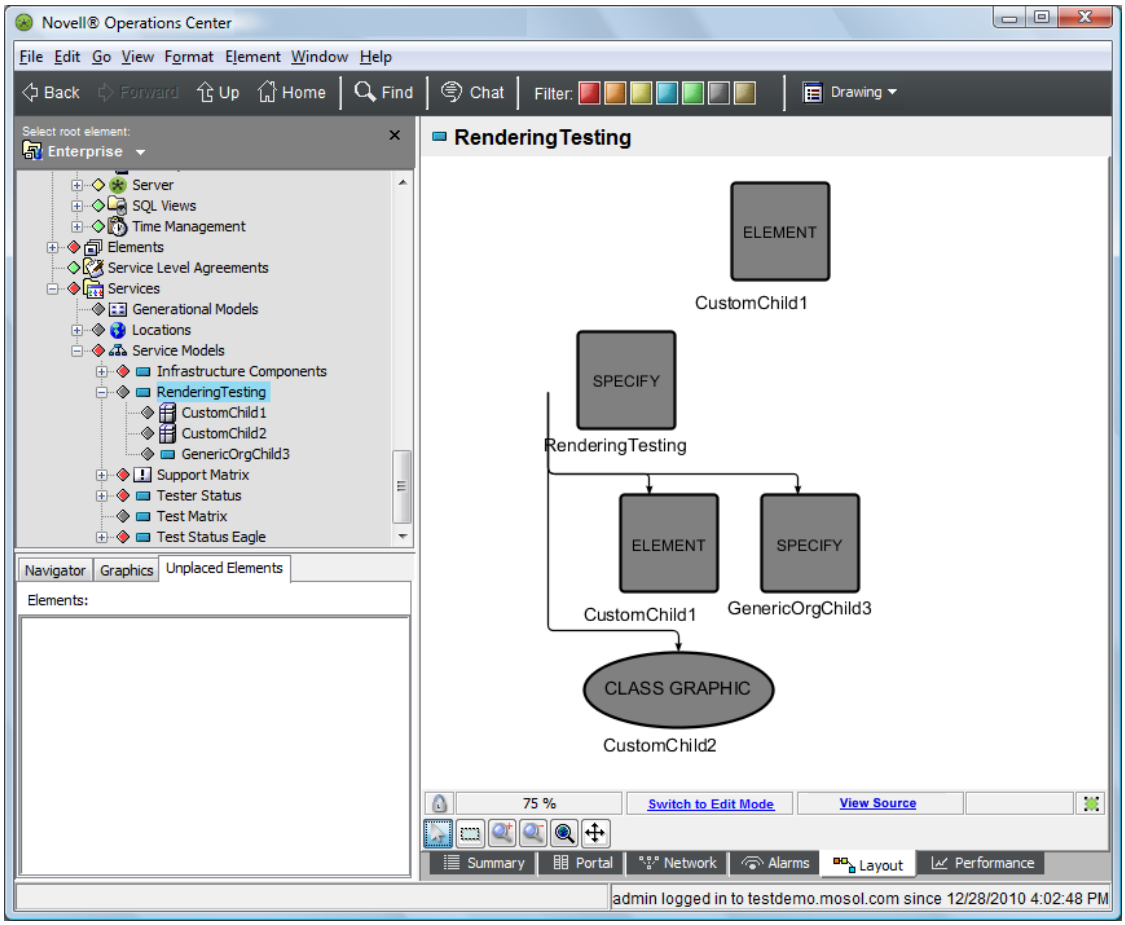
To clear the `element | graphic` switch:

1. In the *Explorer* pane, select the *CustomChild1* element.
2. On the console toolbar, switch to the *Element Graphic* drawing channel.
3. Right-click the background, then select *Clear Element Graphic*.
4. Click *Yes* to accept.
5. Verify that the Element Graphic drawing channel is cleared.
6. Return to the *RenderingTesting* view.

   All copies of the *CustomChild1* element have reverted to the *CustomChild1* element instance node style.

   There could be a short delay before the old rendering is cleared from the graphics cache.

   The resulting drawing should resemble the following:



7. Refresh the Operations Center dashboard and verify the same view displays.

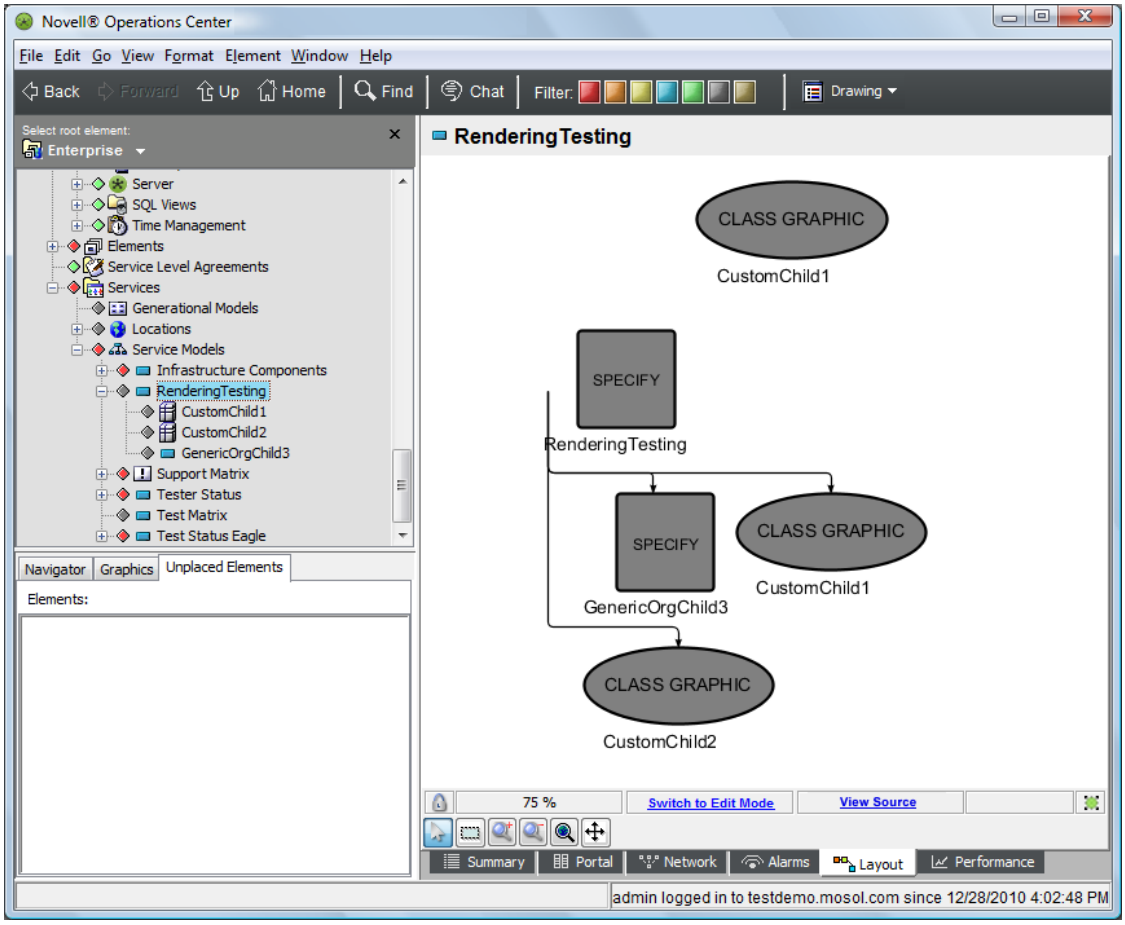
Clearing the Element | Node Style Switch

To clear the `element | node style` switch:

1. In the *Explorer* pane, select the *CustomChild1* element.
2. On the console toolbar, switch to the *Element Nodestyle* drawing channel.
3. Click *Clear* in the upper right-hand corner of the view.
4. Verify the node style clears and the text in the center changes to none.
5. Return to the *RenderingTesting* view and verify all copies of the *CustomChild1* element have reverted to their class definition's custom graphic (defined for the *CustomRendering* custom class).

   There could be a short delay before the old rendering is cleared from the graphics cache.

   The resulting drawing should resemble the following:



6. Refresh the associated portal and verify the same view displays.

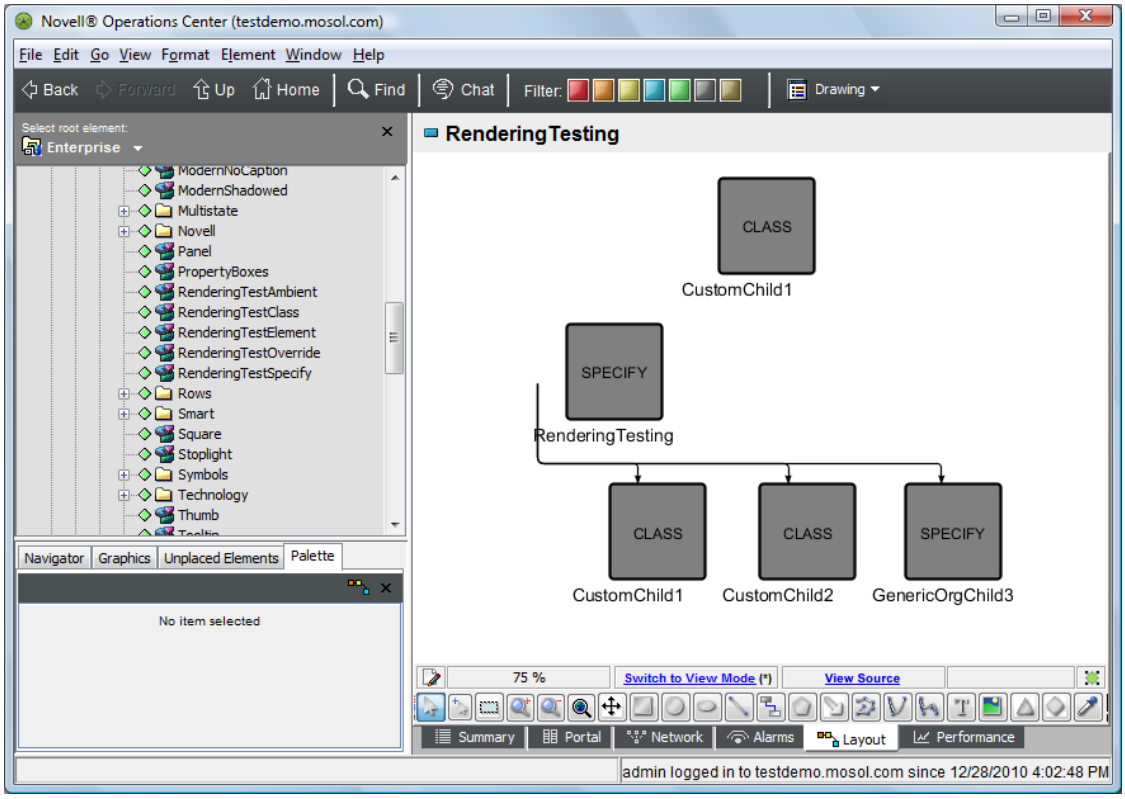
Clearing the Element Class | Graphic Switch

To clear the `element class | graphic` switch:

1. In the *Explorer* pane, expand *Administration* > *Metamodel* > *Classes* > the *CustomRendering* custom class.
2. On the console toolbar, switch to the *Element Graphic* drawing channel.
3. Right-click the drawing, then select *Clear Element Graphic.*
4. Click *Yes* to accept.
5. Verify that the Element Graphic drawing channel is cleared.
6. Return to the *RenderingTesting* Layout view and verify that all copies of the *CustomChild1* element have reverted to their *CustomRendering* class definition's node style.

   There can be a short delay before the old rendering is cleared from the graphics cache.

   The resulting drawing is shown in the following:



7. Refresh the associated portal and verify the same view displays.

Clearing the Element Class | Node Style Switch

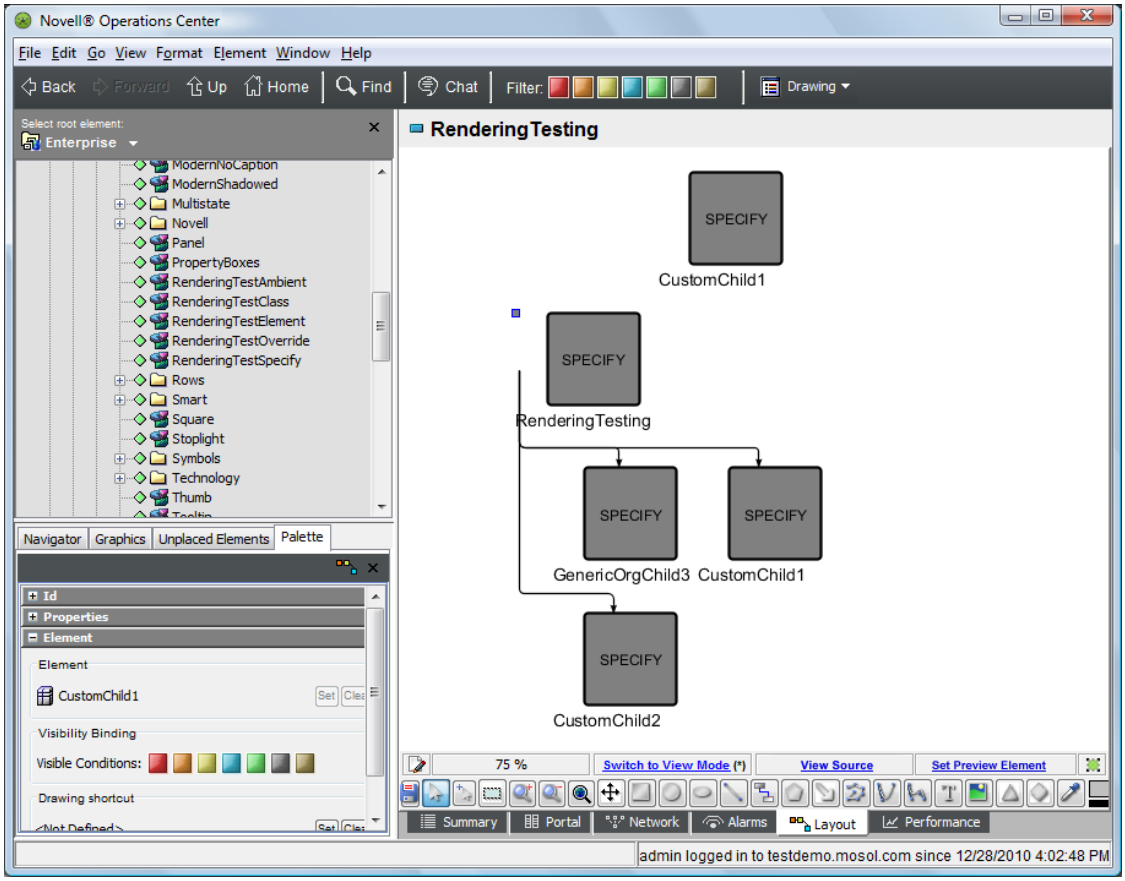
To clear the `element class | node style` switch:

1. In the *Explorer* pane, select the *CustomRendering* class.
2. On the console toolbar, switch to the *Element Nodestyle* drawing channel.
3. Click *Clear* in the upper right-hand corner of the view.
4. Verify the node style clears and the text in the center changes to none.
5. Return to the *RenderingTesting* view and verify that all copies of the *CustomChild1* element have reverted to the SPECIFY node style.

   At this point the Ambient node style is using the *RenderingTestSpecify* node style and the binding node style is also set to *RenderingTestSpecify*.

   There could be a short delay before the old rendering is cleared from the graphics cache.

   The resulting drawing should resemble the following:



6. Refresh the associated portal and verify the same view displays.

B Bind Language API Reference

The bind language is based on a small set of orthogonal concepts. The goal is to create a simple means of marking up XML with constructs to generate structured content, lay out content, and update content. The bindings are intended to do their work both on a functional transform basis and on an incremental basis. Incremental updates are present to allow the efficient updating of a real-time display, and to facilitate incorporation into editing environments.

The three primary forms/phases are:

- Templates
- Layout
- Values

During the template phase the *XML* tree is recursively examined for template constructs, which generate and prune content according to instructions. When all template activity has completed, the layout phase adjusts that content according to layout instructions. Then on an ongoing basis, specific values/attributes within the *XML* tree are updated. If conditions dictate that a structural change occurs (via template activity), then the template and layout phases are repeated.

Using the bind language, you can generate and alter just about anything in an SVG drawing. Everything in SVG is text, so everything is fair game. You can adjust the size of objects, alter their colors, hide and show graphics, change scales and transforms, and anything else that comes to mind. Before we get into the details of the bind language let's outline a series of common scenarios and their solutions. Some of the commands won't make sense until you understand the overall language, but the next sections should give you a more solid idea of what can be built, and how to go about doing it.

- Section B.1, "Bind Task Examples," on page 132
- Section B.2, "Bind Templates," on page 143
- Section B.3, "Layout," on page 146
- Section B.4, "Bind Values," on page 149
- Section B.5, "Resource Generation," on page 152
- Section B.6, "Bind Promotion," on page 152
- Section B.7, "General Bind Annotations," on page 153
- Section B.8, "Script Objects," on page 153

B.1 Bind Task Examples

This section discusses how the underlying SVG/XML level works. When you understand how these tasks are accomplished at the XML level, you'll be in a much better position to understand what's working or not working, and why that is so. And, you'll be able to create custom visualizations and interactions much more easily.

- Section B.1.1, "Adding Tooltips or Pop-Ups to Portlets in the Dashboard," on page 132
- Section B.1.2, "Tying an Element to a Drawing Shape," on page 138
- Section B.1.3, "Changing Shapes or Showing Objects Based on Element Condition," on page 139
- Section B.1.4, "Changing Shapes Based on Element Property," on page 139
- Section B.1.5, "Displaying Element Name or Element Property," on page 140
- Section B.1.6, "Displaying Element Children," on page 141
- Section B.1.7, "Formatting Numbers, Dates, and Time," on page 142

B.1.1 Adding Tooltips or Pop-Ups to Portlets in the Dashboard

In the dashboard, the Layout portlet is used to display the same information (or drawings) that you have in the console's Layout view. You can modify the Layout view drawing code to leverage tooltip and pop-up functionality in the dashboard.

The following sections cover adding tooltips or pop-ups to portlets. This functionality applies to the dashboard.

- "Tooltips for the Layout Portlet" on page 132
- "Pop-ups in the Layout Portlet" on page 134

Tooltips for the Layout Portlet

A tooltip can be displayed when the mouse is hovered over a specific element in the Layout portlet in the dashboard. For more information about the Layout portlet and the dashboard, see the *Operations Center 5.5 Dashboard Guide*.

Although the tooltip is created in the Layout View, it cannot be displayed in a Layout View drawing in the Operations Center console.

The tooltip must be manually inserted into an SVG drawing. The mechanism to create the tooltip requires a specific structure as follows:

```
<g tooltiptype="outer" preserveId="true" display="none">
<g id="uniqueid" tooltiptype="inner" preserveId="true" display="inline"
timeout="7000">
    <…tooltip content goes here…>
   </g>
 </g>
```

This structure consists of an inner and an outer SVG Group object. The purpose of the outer Group object is to hide the inner Group object when the tooltip should not be displayed. A different mechanism directly refers to the inner Group object when it is time to display the tooltip.

The outer group must have a `tooltiptype` of `"outer"` and the inner group must have a `tooltiptype` of `"inner"`. Both groups must have the `preserveId` set to True. The `display` attribute is required for both groups. The outer group could toggle the `display` attribute from `"none"` to `"inline"` at design time in the Operations Center console in order to preview how the tooltip can

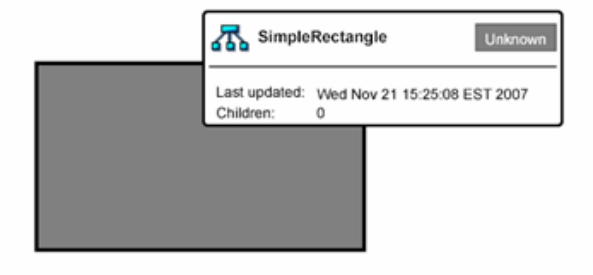
render. The inner group must have `display` set to `"inline"`. The inner group can supply an optional attribute called `timeout`, which sets the display time of the tooltip in milliseconds. The default timeout value is 5000 milliseconds. The inner group MUST have an `id` that is unique within the current document. The tooltip is referenced by this `id`.

The tooltip content can be very flexible, but because the Layout portlet renders the Layout contents as a snapshot, each tooltip must be constructed at rendering time by the dashboard to ensure the tooltip is fully rendered. The tooltip contents can also display "bound" element data. The element that is "bound" is determined by the currently inherited "dname" value where the tooltip structure is defined.

The tooltip for a drawing item is determined by walking up the parent XML hierarchy. At each parent level, if a group object with "tooltip" set to "outer" is encountered as a child of that parent, this tooltip is used. The tooltip cannot be an immediate child of the `<svg>` root drawing element.

Figure B-1 shows a simple tooltip showing an element's "name" when the mouse is hovered over the rectangle:

Figure B-1 *Tooltip Displayed in a Layout Portlet in the Dashboard*



The tooltip in the example is defined as follows:

```
<svg dname="org=SimpleRectangle/org=tooltip/org=testing/root=Organizations"
MOS_EnableMouseEvents="true">
  <g>
    <rect stroke="#000000" stroke-linecap="butt" width="245"
fill="rgb(128,128,128)" stroke-dasharray="none" filter="none" stroke-width="3"
height="141" undofill="#c0c0c0" x="0" stroke-linejoin="miter" opacity="1" y="0">
      <bind:fill property="condition" default="" set="" when=""/>
      </rect>
   <g tooltiptype="outer" preserveId="true" display="inline" transform="matrix( 1,
0, 0, 1, 95, 75)">
      <g tooltiptype="inner" id="rect_caption" preserveId="true">
        <g>
          <rect x="-15.847" y="57.862" fill="white" text-anchor="middle"
width="86.1938" height="16.289" bind:container="true" stroke="gray"
overflow="visible"/>
          <bind:layout name="wrap" pad="2" id="103"/>
          <text x="27" font-size="11px" y="70" fill="black" text-anchor="middle"
gautolayout="103" stroke="none" overflow="visible"
xml:space="preserve">SimpleRectangle<bind:value property="name"/></text>
          </g>
        </g>
      </g>
   </g>
  </svg>
```

In this example, the rectangle and the tooltip outer group element are wrapped inside of another group element. This is done because of how the tooltip is determined by walking up the parent XML hierarchy.

Some of the root <svg> attributes have been omitted for this example. However, there is a required attribute on the root <svg> element. This is the MOS_EnableMouseEvents, which must be set to True in order to activate the mouse handling mechanism in the Layout portlet in the dashboard.

Tooltips can also use node styles as the tooltip content. This tooltip is defined by the following nested outer/inner tooltip structure:

```
<g tooltiptype="outer" preserveId="true" display="inline" transform="matrix( 1, 0,
0, 1, 124, -40)">
    <g id="rect_caption" tooltiptype="inner" preserveId="true" transform="matrix(
1, 0, 0, 1, 0, 0)" display="inline">
        <g>
            <g bind:graphic="Nodes__Detailed"/>
        </g>
    </g>
</g>
```

Using node styles for constructing tooltips is the recommended method for implementing complex tooltips. For example, the tooltip shown in Figure B-1 on page 133 is a node style.

Pop-ups in the Layout Portlet

In the dashboard, the Popup definition allows you to open a new dialog box or Web page from the Layout portlet. Although the pop-up definition is defined inside the Layout View's source code, the pop-up is not available as a feature for use from the Layout View of the Operations Center console—the resulting pop-up is only active in the dashboard depending on how it is defined.

For more information about the Layout portlet and the dashboard, see the *Operations Center 5.5 Dashboard Guide*.

As mentioned above, pop-ups are defined differently depending on whether it is to apply to the dashboard. The following sections describe the Popup definition and show how to implement for the dashboard:

- "Understanding Pop-Up Definition" on page 134
- "Creating a Pop-Up for the Dashboard" on page 135

Understanding Pop-Up Definition

The pop-up definition is used to open a new dialog box or Web page. It contains a parameter identifying the DName of the selected item used to pre-populate data in the new window or Web page.

A pop-up URL template is a group object with the following required parameters:

- **portalPopupURLTemplate:**
  - Defines the actual URL template itself.
  - Refers to an existing URL within the current portlet definitions.
  - The DName must be parameterized in the template with the following replacement tag:

    [INSERT_DNAME_HERE]

This instructs the pop-up mechanism to insert the currently active DName (derived from the click) into the URL when launched.

NOTE: Never replace [INSERT_DNAME_HERE] with an actual DName value, this is the mechanism for Operations Center to dynamically insert the DName of the active element.

- The parameter cannot contain ampersand (&) characters. Replace them with `&`. The & character is a reserved character in SVG and cannot appear in an attribute value.

- **popupTargetFrameIds:** Identifies the target dialog box in which to open the pop-up URL.

Optionally, the following parameters can be defined:

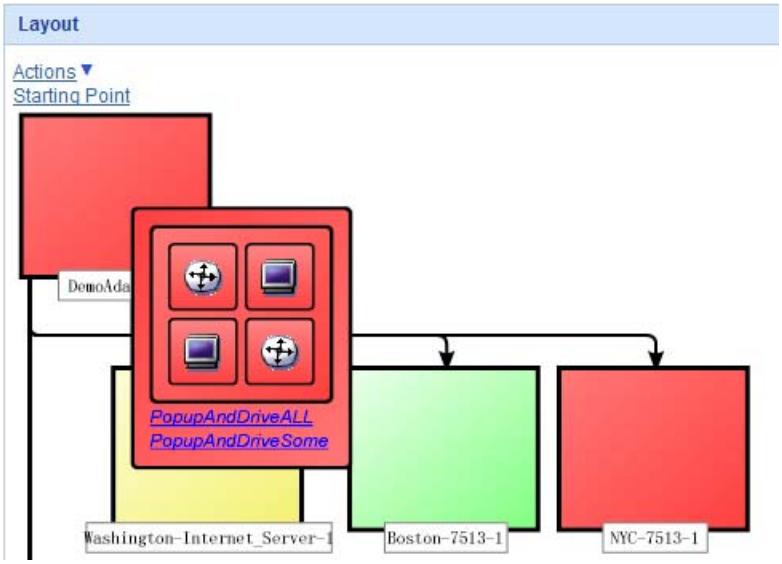
- **popupURLParams:** Defines the parameters used to display the pop-up dialog box. For example, `popupURLParams="toolbar=no,location=no,menubar=no,resizable,scrollbars"`.
- **popupTargetWindow:** Defines the name of the browser window. For example, `popupTargetWindow="_blank"`.

If a direct parent of any child element contains a group object with a `portalPopupURLTemplate` attribute, this pop-up URL Template is used. The group object with a `portalPopupURLTemplate` attribute must be a direct parent of any child that should inherit that pop-up.

Creating a Pop-Up for the Dashboard

Figure B-2 is an example of a pop-up implemented for the dashboard's Layout portlet. A tooltip provides hyperlinks for additional options.

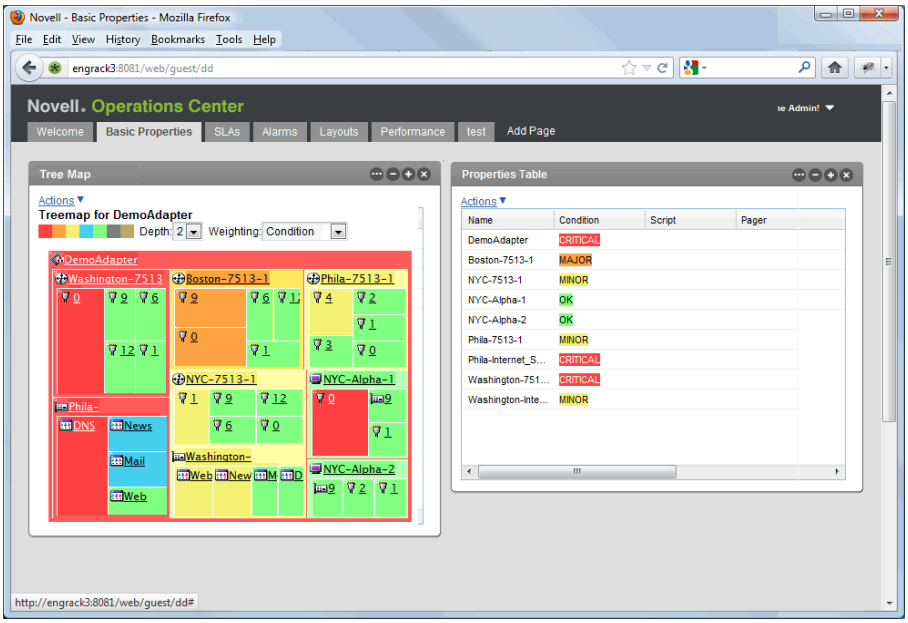
Figure B-2 *Pop-up in Layout Portlet in the Dashboard*



These links contain the DName for the selected element so they can use the `popup` definition to launch and populate other portlets. Clicking a link opens the selected element in a different portlet. For details on portlet structures, see the *Operations Center 5.5 Dashboard Guide*.

For example, clicking *PopupAndDriveALL* in the illustration above opens a new the dashboard page where all portlets on the page are focused to the selected element, as shown in Figure B-3:

Figure B-3 *Clicking an option in the portal pop-up opens this dashboard page in a new browser window.*



When creating a pop-up link in the dashboard, the portalPopupURLTemplate can be used to drive the current element on the page by passing in the portlet's namespace or Portal Identifier with the element DName.

The following code shows syntax using the `popup` parameters to drive all portlets on the dashboard page. Keep in mind it is only a portion of the entire set of Bind language used to create the Layout view. View the Layout View's source code using the *View Source* option in the Operations Center console.

```
<g>
   <g portalPopupURLTemplate="/web/communitya/
page1?global_identity=[INSERT_DNAME_HERE]" popupTargetWindow="our_popup"
popupURLParams="toolbar=no,location=no,menubar=no,resizable,scrollbars">
      <text x="154" font-size="10pt" y="70" text-decoration="underline"
transform="matrix( 1, 0, 0, 1, -146, -16)" fill="blue" filter="none" font-
style="italic" opacity="1" stroke="none">PopupAndDriveALL</text>
   </g>
</g>
```

IMPORTANT: The [INSERT_DNAME_HERE] section of the code should never be manually modified. This is for internal use and instructs Operations Center to dynamically use the DName of the current element.

Links can also be configured to drive only those portlets as specified. Specify the exact portlets to update in the page by including the namespace or identifier for each portlet we wish to drive by appending the `portalPopupURLTemplate` value with a `portlet_ids` parameter such as `&portlet_ids="id1,id2"`.

For example:

```
<g>
    <g portalPopupURLTemplate="/web/communitya/
page1?global_identity=[INSERT_DNAME_HERE]&amp;portlet_ids=layoutA"
popupTargetWindow="our_popup"
popupURLParams="toolbar=no,location=no,menubar=no,resizable,scrollbars">
     <text x="154" font-size="10pt" y="70" transform="matrix( 1, 0, 0, 1, -146, -
16)" text-decoration="underline" fill="blue" filter="none" font-style="italic"
opacity="1" stroke="none">PopupAndDriveSome</text>
   </g>
</g>
```

Using the dashboard portlet identifier feature, you can create user-friendly names for the portlets which is much easier (and shorter) to reference than the namespace. The example above takes advantage of this feature.

NOTE: Some of the root `<svg>` attributes have been omitted for these examples. It is important to note that a `MOS_EnableMouseEvents` must be added as an attribute on the root `<svg>` element and set to *True* in order to activate the mouse handling mechanism in the Layout portlet for the dashboard.

To determine the portalPopupURLTemplate value and the optional portlet identifiers in the dashboard:

1. In Internet Explorer, access the dashboard and go to the page.

   NOTE: Only use Internet Explorer to perform this procedure.

   ---

2. To determine the portalPopupURLTemplate value:

   2a From the Web browser address/location bar, copy the URL which might look something like the following (or in some cases, much simpler):

   `http://serverName:8080/web/guest/popuptemplate`

   2b Remove the server prefix and the extra portlet parameters from the URL (basically retaining the community and page portion of the URL) and insert them into the following template:

   `relative_server_path``?global_identity=`*`[INSERT_DNAME_HERE]`*

   to obtain the following for the example from step 2:

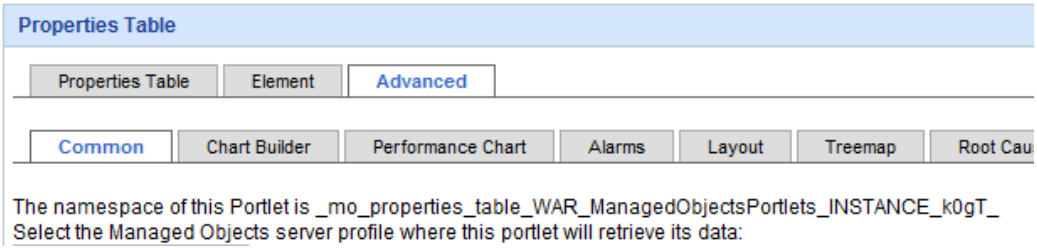
   `/web/guest/popuptemplate?global_identity=`*`[INSERT_DNAME_HERE`*`]`

3. (Optional) To determine the Namespace or Portal Identifier to drive a specific portlet(s):

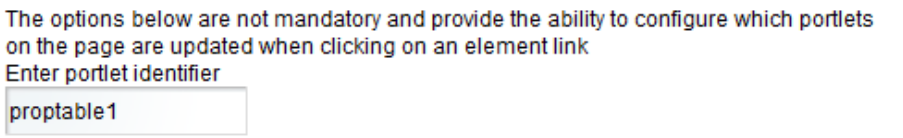
   3a Open the *Preferences* for the portlet to drive and navigate to the *Advanced* tab. The *Common* tab opens under *Advanced*.

   3b Copy the portlet namespace or identifier.

   The namespace is shown at the top of the Common tab:



(Optional) Scroll down to locate or specify the Portlet Identifier. The Portlet Identifier can be easier to use and remember than the portlet namespace as it creates a "friendly URL":



3c In the Layout View's Source, when defining the `portalPopupURLTemplate` as shown earlier in this section, specify the namespace or identifier using the `portlet_ids` parameter to be passed to the page. For example:

```
&amp;portlet_ids=portlet1,portlet2
```

Use a comma delimited list for multiple portlet namespace or identifiers.

For example, by appending our previous example and using friendly URLs (Portlet Identifiers), we have:

```
/web/guest/
popuptemplate?global_identity=INSERT_DNAME_HERE&amp;portlet_ids=proptable4
2,properties42
```

4 In the Layout View's Source, set the `portalPopupURLTemplate` to the value as obtained as a result of Step 2 or Step 3.

B.1.2 Tying an Element to a Drawing Shape

You can tie an element to a drawing shape by setting the DName attribute of the shape:

```
<g DName='root=Organizations'>
  <rect width='100' height='100'>
    <bind:condition/>
  </rect>
</g>
```

SVG ignores this attribute, but the layout system can use it to know which element to bind to the shape.

When set, the DName attribute affects every shape beneath it in the *SVG* or *XML* tree. But you can set other DNames beneath it in the tree.

A DName attribute overrides whatever is set above it.

B.1.3 Changing Shapes or Showing Objects Based on Element Condition

The following sections cover changing shapes and hiding/showing objects:

- "Shape Background Color" on page 139
- "Shape Border Color" on page 139
- "Hiding and Showing Objects" on page 139

Shape Background Color

To change a shape's background color to reflect an element's condition, use the `<bind:condition>` tag inside the shape tag:

```
<rect width='100' height='100' stroke='black'>
  <bind:condition/>
</rect>
```

Shape Border Color

To change a shape's border color to reflect an element's condition, use the `<bind:stroke>` tag inside the shape tag:

```
<rect width='100' height='100' stroke='blue' fill='white' stroke-width='5'>
  <bind:stroke property='condition'/>
</rect>
```

Hiding and Showing Objects

To set the visibility of a shape or object based on element condition, use the `<bind:visibility>` tag:

```
<rect width='100' height='100' visibility='hidden'>
  <bind:visibility when='critical,major' set='visible' else='hidden'/>
</rect>
```

B.1.4 Changing Shapes Based on Element Property

The following sections cover changing shape characteristics based on element property values:

- "Shape Size" on page 139
- "Scaling by a Factor when Setting a Numeric Property" on page 140
- "Shape Style Based on Element Property Value" on page 140
- "Shape Transparency Based on Element Property Value" on page 140

Shape Size

To set the size of a shape based on property value, use the `<bind:width>` tag, then specify a property attribute:

```
<rect width='50' height='20'>
  <bind:width property='users'/>
</rect>
```

The shape width changes based on the value of the property.

Scaling by a Factor when Setting a Numeric Property

To scale an attribute by a factor using an element property value, specify a value to multiply against the property value:

```
<rect width='50' height='20'>
  <bind:width>element['users'] * 5</bind:width>
</rect>
```

Shape Style Based on Element Property Value

To change the style of a shape based on an element property, conditional statements can be used. In the following example, the border of the shape is grey:

```
<rect stroke-width='1' stroke='grey' width='100' height='100'>
  <bind:stroke-width>
<![CDATA[
if (element['users'] < 50) 1 else 5
]]>
  </bind:stroke-width>
  <bind:stroke>
<![CDATA[
if (element['users'] < 50) 'green' else 'red'
]]>
  </bind:stroke>
</rect>
```

If the `users` property on the element is greater than 50, the shape border increases to a width of 5 pixels and changes from green to red.

The following partial code example shows a simplified ranging syntax:

```
<rect width='100' height='100'>
  <bind:stroke-width property='users' range='< 50 :green;< 80 :orange' else='red'/>
</rect>
```

Shape Transparency Based on Element Property Value

To change an objects transparency based on an element property, use the `<bind:opacity>` tag. In the following example, the property value is divided by 100 to set a percentage value of opacity.

```
<rect fill='green' stroke='black' opacity='1'>
  <bind:opacity>element['users'] / 100.0</bind:opacity>
</rect>
```

B.1.5 Displaying Element Name or Element Property

The following sections cover how to display and element name or property value:

- "Element Name" on page 141
- "Element Property Value" on page 141
- "Default Property Value" on page 141
- "Controlling Width of Text" on page 141

Element Name

To show the name of an element, use the <bind:value> tag, setting the bind property to 'name':

```
<text font-size='18' text-anchor='middle'>
  <bind:value property='name'/>
</text>
```

Element Property Value

To show the value of a specific element property, use the <bind:value> tag, setting property to the element property name:

```
<text>
  <bind:value property='responseTime'/>
</text>
```

Default Property Value

If a property doesn't exist, you can set the default value by specifying a default attribute on the <bind:value> tag:

```
<text text-anchor='middle'>
  <bind:value property='responseTime' default='not available'/>
</text>
```

Controlling Width of Text

Control the width of wrapped text by adding a wrap tag to a <bind:value> on a text-based content bind. The value of the wrap tag controls the width of the wrapped text. The wrapper attempts to break on spaces, but it breaks a word if it is longer than the specified wrap width.

This example sets the wrap width to 75:

```
<text overflow="visible" x="27" y="65" font-size="10" text-
anchor="middle">ClassicWrap<bind:value property="name" wrap="75"
texthash="1.52892531E9" undo="Classic Small"/>
</text>
```

Notice that the wrap attribute must be contained inside of the <bind:value> tag.

B.1.6 Displaying Element Children

The following sections cover how to display the number of children or the children along a path of a shape:

- "Number of Children" on page 142
- "Number of ORG Children" on page 142
- "Children along a Path" on page 142

Number of Children

To show the number of children for the element, set the `property` attribute to `childCount` in the `<bind:value>` tag:

```
<text>
  <bind:value property='childCount'/>
</text>
```

Number of ORG Children

To show the number of children for the element, set the `property` attribute to `orgChildCount` in the `<bind:value>` tag:

```
<text>
  <bind:value property='orgChildCount'/>
</text>
```

Children along a Path

Children can be displayed for any element. By using or modifying the following code, the children can be displayed along the circumference of a circle or the edge of any shape (for example, rectangles, lines, paths, or curves).

In this example, we show setting them to display along the edge of a circle:

```
<circle id='circlepath' fill='none' stroke='grey' r='500'/>
<bind:children graphic='Nodes__Classic'>
  <bind:layout path='#circlePath'/>
</bind:children>
```

B.1.7 Formatting Numbers, Dates, and Time

The following section cover how to format numbers, dates and times:

- "Numbers" on page 142
- "Time" on page 143
- "Date" on page 143
- "Date-Time" on page 143
- "Custom Date-Time" on page 143

Numbers

Format numbers using Java's Formatter class. For example:

```
<text>
  <bind:value property='childCount' format='Number of children: %1$d'/>
</text>
```

The above formats an integer property. To format a floating point property, use `%1$f` instead of `%1$d`.

Java's Formatter class documentation can be referenced at http://java.sun.com/j2se/1.5.0/docs/api/java/util/Formatter.html (http://java.sun.com/j2se/1.5.0/docs/api/java/util/Formatter.html).

Time

```
<text>
  <bind:value property='lastUpdate' format='time'/>
</text>
```

Date

```
<text>
  <bind:value property='lastUpdate' format='time'/>
</text>
```

Date-Time

```
<text>
  <bind:value property='lastUpdate' format='datetime'/>
</text>
```

Custom Date-Time

In addition to the shortcuts supplied above (which do locale-specific date and time formatting automatically), you can specify the exact format of a date-time string by supply a format compatible with Java's SimpleDateFormat class:

```
<text>
  <bind:value property='lastUpdate' format='EEE, d MMM yyyy HH:mm:ss Z'/>
</text>
```

B.2 Bind Templates

The bind language provides an ability to create copies of template objects, based on the shape of the input data. Diagrams are frequently composed of sections, where those sections are filled with instances of various elements related to the section. Similar visual representations are often used to for display, but each figure is customized in certain ways. An icon representing a server can be the same across all server depictions, but the fill color of the icon could be derived from the server's state.

The following subsections describe the attributes, commands, and variables for bind templates:

- Section B.2.1, "Bind Attributes," on page 144
- Section B.2.2, "Template Subcommands," on page 145
- Section B.2.3, "Layout Bind Commands," on page 145
- Section B.2.4, "Layout Target Element Attributes," on page 146
- Section B.2.5, "Generated Copy Variables," on page 146

B.2.1 Bind Attributes

The following are valid bind attributes:

- **target:** Contains an element reference to the element that will be the parent *SVG* element to any newly created child objects.

  By default this is the parent element of the `bind:templates` command. Several steps are performed to attempt to locate a different element target:

  1. If the element reference begins with the # character, the rest of the reference designates an XML element directly by ID lookup.
  2. If the element reference is nonempty and does not begin with the # character, it is interpreted as an XPath expression rooted at the command. XPath provides a powerful ability to use relative navigation and querying to specify the element to use.
  3. If the element reference is empty, the siblings of the command is searched to see if any of them possess a `bind:container` attribute. If more than one sibling has the `bind:container` attribute, the closest is chosen (by distance).
  4. Otherwise, the parent of the command is selected as the target.

- **graphic:** Contains an element reference to the graphic element that is copied to the target.

  The same steps for element reference resolution apply, excluding 3 and 4.

- **lookup:** Specifies a strategy for looking up a graphic to insert.

  This is a comma-delimited list, order-sensitive, with the following possibilities. Each strategy can have a drawing identifier, specified as strategy?drawing.

  - **element:** Retrieve a drawing from the element being laid out.
  - **class:** Retrieve a drawing from the class of the element being laid out.
  - **bind:** Use the graphic specified on the bind command.
  - **icon:** Try to bind in the large icon of the source.
  - **smallicon:** Bind in the small icon of the source.
  - **url(urlIdentifier):** Use the graphic reachable by the url.

- **source:** Designates the source of element information to copy.

  Allowable values are:

  - **children:** All NAM and ORG children of the element.
  - **nam:** All NAM children.
  - **org:** All ORG children.
  - **element:** The nearest single element DName.
  - **properties:** The properties of the current element.
  - **clipart:** Names a clipart object to introduce into the diagram. This is normally used only by the UI while dragging clipart into the drawing.
  - **[future] offers:** The service level offers an element participates in.

- **classFilter:** A regular expression that must match against the element class name of the sources.

  If you want to match against `host` or `router`, you might enter `host|router`.

- **nameFilter:** A regular expression used to filter the full names of the sources.

  For elements, it filters the DNames. For properties, it filters the property names.

- **scriptFilter:** You can provide a NOC script that receives element, DName, and so forth as variables.

  Return True if the element should be included, and False if it should not. Any other return value is considered to be False.
- **unplaced:** Provides a means of laying out elements that are not placed within the current drawing.

  You can set it to the following values:
  - **layout:** Shows elements that are not handled by the current layout command.
  - **drawing:** Shows elements that are not placed anywhere else in the current drawing.
- **resize:** Resizes the graphic.
- **transform:** Designates a transform that is applied to each of the cloned visual elements.

  This is useful for applying a uniform scaling across children, or for offsets.
- **source-port:** When connecting, use this port as the default source point for connection lines.

  Defaults to South.

  If a top-to-bottom graph is being laid out, the source port is South and the default `dest-port` is North.
- **dest-port:** When connecting, use this port as the default destination.

B.2.2 Template Subcommands

The following template subcommands are available:

- **bindfilter:** Filters out elements based on an expression.

  This can be used, for example, to exclude child elements that are not critical.
- **bind:sort:** Designates a sort ordering for child elements.

  This can be used to sort on child element name.

B.2.3 Layout Bind Commands

The following layout bind commands are available:

- **bind:templates:** This is the core `bind:templates` command.

  Each of the other commands listed here is transformed into a `bind:templates` command, internally.
- **bind:children:** Equivalent to `bind:templates` with `source=children` (all NAM and ORG children).
- **bind:org:** Equivalent to `bind:templates` with `source=org` (ORG children).
- **bind:nam:** Equivalent to `bind:templates` with `source=nam` (NAM children).
- **bind:element:** Equivalent to `bind:templates` with `source=element` (binds self, the nearest scoped object).
- **bind:link:** Use link objects to direct connections within the current child/layout groupings.
- **bind:properties:** Equivalent to `bind:templates` with `source=properties` (current object's properties).

B.2.4 Layout Target Element Attributes

Elements that are siblings of the `bind:templates` command are eligible for layout. Certain attributes allow further control of this process. They are as follows:

- **bind:group:** A given layout can be performed with respect to a group of child objects.

  Only those child objects tagged with the specified group are included in the layout. This allows multiple layouts to be performed on a parent object, if that is desired. It also provides an easy mechanism to exclude elements from a layout (simply assign a different group).

  The `bind:group` is optional. If the `bind:layout` command does not specify a group, the default group is assumed.

  All elements that are not tagged with a `bind:group` attribute are assumed to be part of the default group.
- **bind:container:** Designates an element as a possible container for template copies.

  If a `bind:templates` command does not specify a target, the nearest `bind:container` to it (if one exists) is selected as the parent object for the newly created copies.

B.2.5 Generated Copy Variables

As each clone element is created, certain additional attribute variables are added to the generated element to make further binding easier to perform. These variables create certain curves and make them available to further binding commands. They can be used to create certain effects that change when an object is cloned along a path or into a container, such as fading out or scaling down.

- **alpha:** Linear, from 0.0 to 1.0, evenly spaced and incremented for each child.
- **log:** Logarithmic from 0.0 to 1.0, shifting distribution towards end of scale.
- **sigma:** S-curve, distribution toward both ends.

B.3 Layout

Layout operates in a context, which is the siblings of the `bind:layout` element.

Every sibling element with graphical content (anything visual) with the same `bind:group` is part of the layout.

Any relationships between the elements (parent-child, and so on) are visible as lines drawn between the visual elements.

Because layout operates on siblings it is common to place the layout command within a group:

```
<g>
  <bind:layout name="tree"/>
  <rect bind:container="true" width="1"
    height="1" stroke="blue" fill="none"/>
  <rect width="50" height="25"/>
  <circle r="25" stroke="blue" fill="none" />
</g>
```

The `bind:layout` operates against the second rectangle and the circle. The first rectangle has `bind:container=true` specified. When a rectangle is tagged with `bind:container`, it is excluded from the layout. After layout is completed the bound of the laid-out graphics is computed, and the container rectangle is stretched to fit around them. This provides a convenient way of wrapping automatically laid-out graphics.

It's frequent to want only the wrapping behavior and no other layout. Just specify `name="wrap"` in your `bind:layout` command if you want this. We often want to apply automatic layout to only those elements that have been automatically generated by a variant of the `bind:template` element.

You can place the `bind:layout` element inside a `bind:template` (or `bind:children`) element, as follows:

```
<g>
  <bind:children graphic="Nodes__Classic#node" depth="2">
    <bind:layout name="ortho"/>
  </bind:children>
  <rect bind:container="true" width="1" height="1"
    stroke="blue" fill="none"/>
  <circle r="25" stroke="blue" fill="none" />
</g>
```

The `rect` and `circle` elements are not automatically laid out. Only visual elements generated by the `bind:children` command are.

Bind layout attributes:

- **resize:** Most layout methods move the child objects, as necessary.

  It is often desirable that a parent figure wrap the children, after layout.

  By default the bind engine attempts to resize the parent object; otherwise, it searches for a `bind:container` element.

- **bind:group:** Group tag for elements that should be laid out.

  As with templates, a default group is present.

- **name:** Uses the named layout algorithm.

  First, this looks for saved layouts. If an appropriate layout is found, it is used. Otherwise, the following are possible:

  - tree
  - hier
  - circle
  - balloon
  - grip
  - organic
  - grid

Property	Default	Description
columns	4	The number of columns in the grid.
columnWidth	100	The width of the columns that are created.
rowHeight	130	The height of the rows that are created.
gap	5	Gap between rows and columns.

- wrap

  Wraps a rectangular shape around the other children being laid out.

  This is generally used to surround a group of child objects with a rectangle after the child objects have been laid out.

  The rectangular object should have the `bind:container` attribute set to True so wrap will know which shape to adjust.

Property	Default	Description
pad	3	Sets overall padding when wrapping a rectangular shape around another.
xpad	0	Sets padding on the left and right when wrapping.
ypad		Sets padding on the top and bottom when wrapping.

- flow

Property	Default	Description
width	400	If specified, ensures that the resulting layout will be the specified width. This is an easy way to flow a series of elements into a grouping, which will wrap if they cannot fit into the width specified.

- border

  When border is specified, the child elements can have a constraint attribute. This attribute should be one of north, east, south, west, or center.

  Elements on the edges are pushed to that edge. The element labeled `center` occupies the rest of the space.

- table

  If none is specified, a default of `hier` is used.

  Table performs box layout on subelements. The syntax is somewhat different from a conventional table layout, as we should ensure that the SVG engine can "see" the graphic elements we are arranging. Instead of putting our graphical elements inside *TD* and *TR*

elements, we put our table bindings inside the graphical elements. If a width or height attribute is provided for a `tableRow` or `tableColumn`, that row or column is set to that width. Otherwise, the maximum height and/or width for the column/row is used.

```
<g>
  <rect bind:container="true" width="100" height="100"/>
  <bind:layout name="table"/>
  <g>
    <bind:tableRow/>
    <text>Column 1</text>
    <text>Column 2</text>
    <text>Column 3</text>
  </g>
  <g>
    <bind:tableRow/>
    <g><circle radius="25"><bind:condition/></circle></g>
    <g><circle radius="50"><bind:conditiongradient/></circle></g>
    <g>
      <bind:layout name="table"/>
       <g>
         <bind:tableRow/>
         <text>Column 1</text>
         <text>Column 2</text>
         <text>Column 3</text>
      </g>
    </g>
  </g>
</g>
```

- **path:** An element reference (as above, minus 3 and 4). If specified, it indicates that the cloned objects are laid out alone the specified shape's path. Implies `type="path"`.
- **spacing:** One of alpha, log, sigma, or a number indicated the fixed spacing number to use.

  When cloned objects are laid out along a path, the specified curve is used to distribute them.

  If a fixed spacing number is provided, objects are spaced according to that until the length of the path is used up. Subsequent objects are discarded.
- **stroke:** As in SVG, specifies the stroke paint used to draw connections. This can be a color, or any other legal SVG paint.
- **stroke-width:** As in SVG, specifies the width of the strokes used to draw connections.
- **stroke-opacity:** Transparency of connection strokes.
- **stroke-dasharray:** Specifies dash pattern for connections.
- **stroke-dashoffset:** Dash pattern offset for connections.
- **stroke-linecap:** Endpoint specification for connections.
- **stroke-linejoin:** Join specification for connections.
- **stroke-miterlimit:** Miter spec for connection joins.
- **marker-start:** Start point marker.
- **marker-mid:** Marker used for each vertex on connectors, except the first and last.
- **marker-end:** Marker used on the last point of a connection.

B.4 Bind Values

Binding of values is one of the core functions of the dynamic SVG binder. You can create a bind command that sets the attribute of its parent element, or sets the text value. The bind command does two things: It specifies how to generate the value to be set, and also specifies where that value is to be placed. Binds are probably the single most used command, so the default behavior of any element in the bind namespace is to set an attribute of that name:

```
<bind:anything value-expressions.. />
```

You can also specify the attribute you want to set by including it on the general form of the command:

```
<bind:value attr="attrname" value-expressions... />
```

It's common to set the text content of the parent element, rather than set an attribute. A `bind:value` without an attribute does this:

```
<bind:value value-expressions... />
```

Value-expressions generate the strings that are placed as text content or as attributes. You can gather properties from bound elements, use special property forms for convenience, execute velocity expressions, and execute `xpath` expressions.

- Section B.4.1, "Bind Value Attributes," on page 150
- Section B.4.2, "Using NOC Script in Bind:Value Tag," on page 151

B.4.1 Bind Value Attributes

Bind values attributes:

- **attr:** Names the attribute on the parent element that is set by this binding command.

  When not present and no special form is being used, the text content of the parent element is set.
- **bulkattrs:** Names a list of comma separated element properties to request and cache in a single request. Requires the use of NOC script to display the values.

  Must use `cachingelement` instead of `element` to request the attriubes at the NOC script level. This element provides the cached attributes specified by the `bulkattrs` attribute.

  For example, see Section B.4.2, "Using NOC Script in Bind:Value Tag," on page 151. For information about available script objects, see Section B.8, "Script Objects," on page 153
- **property|prop|p:** Specifies a property to retrieve on the bound element, or specifies the name of a specially handled property.:
  - **childcount:** The number of NAM children of the element.
  - **orgchildcount:** The number of ORG (contributing) children of the element.
  - **condition:** Translates the condition of the element into an SVG color.
  - **conditionhighlight:** Translates the condition of the element into a somewhat lighter SVG color.
  - **conditiontext:** Name of the condition.
  - **conditionbackground:** Background color, usually the same as condition.
  - **conditionforeground:** Color for text foreground — make this the color of `conditiontext` if the text is written against a fill of condition.

- **conditiongradient:** Translates the condition of the element into a nice horizontal gradient reference. When specifying `conditiongradient` you can also specify the prefix attribute. The bind engine appends the integer value of the `ElementCondition` to the prefix you specify:

  ```
  <bind:value attr="fill" property="conditiongradient"
    prefix="Static__statics#square_">
  ```

  If the element's condition is critical, change the fill property to:

  ```
  fill="url(Static__statics#square_1)"
  ```

- **DName:** Retrieves the DName of the element.
- **name:** Retrieves the name of the element.
- **image:** Creates an in-line reference to the element's large icon.
- **smallimage:** Creates an in-line reference to the element's small icon.
- **scope:** Instructs the bind engine to perform a scoped lookup for the property name. This is normally used only with the `bind:properties` template expression, where the cloned elements are generated with property attributes set on them.

- **value|val|v:** Contains a NOC Script expression that is evaluated.

  The result is a string that is set to the named attribute.

- **xpath|xp|x:** Specifies an `xpath` expression.

  The `xpath` expression is executed with the binding element as its context. The resulting nodes will have their text contents extracted, appended together, and set as the value of the attribute.

- **format:** Supplies additional formatting for some kinds of objects.

  For `java.util.Date`, this can be set to time, date, date/time, or to any date/time formatting string that is acceptable to the java.text.SimpleDateFormat class. For other, nondate objects, the format string is passed to Java's String.format method, allowing arbitrary formatting. For the format string syntax and possibilities, see the Java documentation.

- **when:** If specified, the result of the value-expressions must match one of the entries in the when clause, which is a comma-separated list.

  If a match is not found, the default value (defined next) is used.

- **default, else:** Defines a value to be use if the expression results in an empty string, or if the when clause fails.

- **set:** Used in conjunction with when to provide a simple "if" capability.

  If the when condition matches, the value of the set attribute is returned. If the condition does not match, the default attribute's value is used.

B.4.2 Using NOC Script in Bind:Value Tag

Using NOC script inside the Bind:Value tag, you can perform a single request to display the values of element multiple properties in the Layout view.

As described above, this mechanism makes a single request for needed element properties for a given drawing instead of making several individual requests. This can be especially beneficial in cases when a node style displays multiple property values; where instead of multiple requests, all values for all elements are returned by a single request.

For non-script bindings in the Layout view, these attributes are auto-detected by the engine. But, when using NOC script, the user must specify the element properties to be bulk requested in the `bulkattrs` attribute, then use `cachingelement` to retrieve the values to a variable which is then used to display them.

In the following example, NOC script is used to display 5 custom attributes on an element:

```
<text x="14" font-size="14pt" y="131" fill="#000000" text-anchor="middle"
filter="none" opacity="1" stroke="none">
<bind:value  bulkattrs="PropertyName1,PropertyName2,PropertyName3,PropertyName4,
PropertyName5">

<![CDATA[
var val1 = cachingelement["PropertyName1"];
var val2 = cachingelement["PropertyName2"];
var val3 = cachingelement["PropertyName3"];
var val4 = cachingelement["PropertyName4"];
var val5 = cachingelement["PropertyName5"];
val1+":"+val2+":"+val3+":"+val4+":"+val5
]]>
```

When the code is rendered, the output is something like the following:

```
</
bind:value>Property1value:Property2value:Property3value:Property4value:Property5va
lue</text>
```

For information about available script objects, see Section B.8, “Script Objects,” on page 153

B.5 Resource Generation

The `bind:gradient` element has the following attributes:

- **condition:** Can be OK, Critical, Major, Minor, and so on.
- **type:** Is either “linear” or “radial.”

These create a simple linear, rectangular gradient based on the specified condition color.

A lighter version of the color is used for the other end of the gradient, if the ending color is not specified.

The `type` attribute defaults to linear.

B.6 Bind Promotion

Special bind commands are available to assist in the creation of example-based palettes. The `bind:promote` attribute tags parts of an SVG drawing so they can participate in the drawing editor's example palettes.

It has the following attribute:

- **bind:promote:** A comma-separated list of promotion commands, where each promotion command consists of the following:

  `sourceAttr[->destAttr]`

  When an element possessing a `bind:promote` attribute is selected in the example palette, the value of the source attribute is copied to the currently selected objects in the drawing.

  If the arrow (->) and destAttr are present, the attribute is renamed.

B.7 General Bind Annotations

Certain bindings are placed as attributes throughout the SVG model. These set context for operations, and direct how certain functions should be performed. They are independent of the various `bind:*` elements. These can exist at arbitrary points in the XML structure.

General bind annotations:

- **bind:highlight:** Signifies a match for palette handling.

  The bind engine adds `bind:highlight` to elements in a palette that match against the element passed to the highlight method. It is set to `Some` if at least one element in the selection matches; otherwise, it is set to `All` if all elements in the selection match.

- **bind:DName:** Sets the DName, which is used as the root object for most commands.

  This is a scoped attribute. When a command requires a DName (or other object identification) to continue, it searches from the binding location upwards to parents until a binding is found.

B.8 Script Objects

The following script objects are available when using NOC scripts for custom Layout views:

- **`element`**: Operations Center element, which is documented in the Operations Center 5.5 Scripting Guide.
- **`dname`**: Dname derived from the element in the current scope of the executing layout view script.
- **`cachingelement`**: A helper element used to access bulk-requested element attributes. See Section B.4, "Bind Values," on page 149.
- **`document`**: A reference to the current layout view's XML document. This document is of type `org.w3c.dom.svg.SVGDocument` which is a subclass of `org.w3c.dom.Document`.

These script objects are for use with NOC Script. For more information about NOC Script, see the *Operations Center 5.5 Scripting Guide*.

Note that `binding`, `target`, `mode` `layoutElement`, and `util` objects are intended for internal use and are subject to change in a future release.

C Editing the Source SVG Code

If you are familiar with SVG code, you might consider editing the SVG source code that controls the Operations Center Layout view. Some features are not available through the Layout view menus, but can be implemented using SVG code. One such feature is the opacity attribute, which uses the element condition to determine its transparency in the Layout view. For example, if the element is in an OK state, then the condition does not display (it changes to transparent); however, if it is CRITICAL, it does display (it shows at 100% opacity.

IMPORTANT: Operations Center is not responsible for the behavior of user-modified SVG code. By editing the SVG code, you accept all responsibility for the results. Consider saving a backup of the current drawing before editing the SVG code. Use Appendix B, "Bind Language API Reference," on page 131 as a reference when editing.

- Section C.1, "Editing the SVG Code and Regenerate the Drawing," on page 155
- Section C.2, "Using the Opacity Attribute," on page 156
- Section C.3, "Using Relative DNames for Node Attribute Matching," on page 156
- Section C.4, "Looking at the Source Code with a Toggling Graphic Node Style Example," on page 158
- Section C.5, "Looking at the Source Code with a Child Container Example," on page 168

C.1 Editing the SVG Code and Regenerate the Drawing

The *View Source* link allows you to examine and directly edit the SVG code associated with a drawing. The *Highlight in Source View* option automatically marks the SVG code for a selected object in the drawing.

To edit the SVG code:

1. In Edit mode, do one of the following:
   - Right-click an element, then select *Highlight in Source View*. The Source view displays the highlighted code associated with the element.
   - Click *View Source*.
2. Right-click the background of the *Source* view to access editing options, including:
   - **Find, Find Next:** Locates specified text string.
   - **Highlight Syntax:** Highlights selected text string.
   - **Insert CDATA Marks:** Saves typing time by entering:

     ```
     <![CDATA[ ]]>
     ```
   - **Insert Bind Script Template:** Saves typing time by entering bind command syntax, which you can then customize.

- **Remove Generated Elements:** Removes all text related to generated elements in the drawing.
- **Turn on Line Wrap:** Enables line wrapping for easier viewing.
- **Small/Medium/Large:** Changes text size in Source View.

3 Click  (*Save*).

4 Click *Back to Preview* to switch from the code view to the Layout view drawing.

The XML validator examines the edited SVG code and attempts to highlight any problematic code.

5 Right-click the drawing background, then select *Regenerate Drawing*.

C.2 Using the Opacity Attribute

The opacity attribute determines the element's transparency based on the element condition. The following code example shows an SVG rectangle object whose opacity attribute is set by the Layout bind engine, based on the bound element's condition. Multiple options could be specified in the when attribute to allow banding of the opacity levels. The following example creates four bands across the set of element conditions, where opacity is 100% for elements with a CRITICAL condition and 20 percent for elements with an UNKNOWN condition.

```
<rect viewlink="org=Parent/root=Organizations" stroke="#000000"
visibility="visible" dname="org=Parent/root=Organizations" width="213"
fill="rgb(255,64,64)" stroke-width="3" height="163" opacity="1.0" x="-253"
y="364">
    <bind:fill property="condition" default="" undo="#c0c0c0" set="" when=""/>
    <bind:opacity property="condition" default="1.0"
when="Unmanaged+Unknown=0.2,Info+OK=0.4,Major+Minor=0.6,Critical=1.0"/>
</rect>
```

Another option is changing the rectangle's line width (1, 2, 3, and so on) according to the element condition. Use the following code:

```
when='ok+info=1,minor=2,major+critical=3' default='1'/>
```

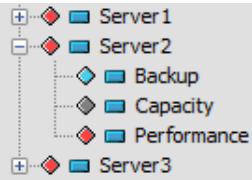
C.3 Using Relative DNames for Node Attribute Matching

When you have element structures that are reused in various hierarchies, it is possible to use relative child dname references when creating node styles; that can then be applied to any similar element hierarchy and will apply rendering rules without further customization.

Using this feature, the node style might contain a series of shapes, each filled with the condition of a different child element. Or, a series of text labels, each updated with an element property value for a different child element. What makes this feature unique is that each child dname reference is added as a relative reference. This means that the node syle can be applied to any hierarchy that has the same set of child elements.

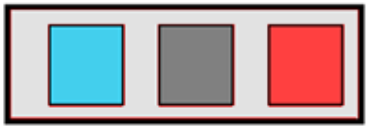
For example, let's say we have various server-related parent elements and each has the same three child elements to surface information about *Backup, Capacity, and Performance*; as shown in the figure below.

Figure C-1 *Hierarchy Structure with Standard Set of Child Elements*



And, we create a node style that displays the condition color of the three child elements: *Backup*, *Capacity*, and *Performance*.

Figure C-2 *Node Style showing condition of Backup, Capacity and Performance Child Elements*



In our source code, we bind each rectangle to each child element's relative dName so that the node style can be reused for other server parent elements. Notice in the code sample below, the dname attribute for rectangle one is `dname="org=Backup/.."` which is a relative reference to look for any ORG child element named `Backup`.

```
  <rect stroke="#000000" id="0" stroke-linecap="butt" dname="org=Backup/.."
width="38" dname_relid="-1" fill="rgb(128,128,128)" bind:marks="missing" stroke-
dasharray="none" filter="none" stroke-width="1" height="42" undofill="#c0c0c0"
x="21" stroke-linejoin="miter" opacity="1" y="108">

    <bind:fill property="condition" default="" set="" when=""/>

  </rect>

  <rect stroke="#000000" id="1" stroke-linecap="butt" dname="org=Capacity/.."
width="38" dname_relid="-1" fill="rgb(128,128,128)" bind:marks="missing" stroke-
dasharray="none" filter="none" stroke-width="1" height="42" undofill="#c0c0c0"
x="78" stroke-linejoin="miter" opacity="1" y="108">

    <bind:fill property="condition" default="" set="" when=""/>

  </rect>

  <rect stroke="#000000" id="2" stroke-linecap="butt" dname="org=Performance/.."
width="38" dname_relid="-1" fill="rgb(128,128,128)" bind:marks="missing" stroke-
dasharray="none" filter="none" stroke-width="1" height="42" undofill="#c0c0c0"
x="135" stroke-linejoin="miter" opacity="1" y="108">

    <bind:fill property="condition" default="" set="" when=""/>

  </rect>
```

C.4 Looking at the Source Code with a Toggling Graphic Node Style Example

In this example, we'll build a new node style and take a look at what happens as the source code and bind commands are built in the background.

We'll set the node style to display different graphic displays based on element state, such as:



Shows recycle symbol when state is OK.



Shows a yellow caution symbol when element is any state other than OK or Critical.



Shows a red/round attention symbol when state is Critical.

In order to set this up, we'll create a "toggle" between graphics based on the element condition.

To do this, we will take advantage of a stack container, which allows you to overlay multiple objects directly on top of each other. Bind rules are then created for each graphic to indicate when it is visible or hidden based on specific element conditions.

If these bind rules are not set properly, two graphics can possibly show for a particular state. In some cases, this might be desirable. You can have an extra adornment for some states in addition to the main graphic toggle and a condition color change.

To build a node style that toggles between various graphics based on state:

1. In the *Explorer* pane, expand *Administration* > *Graphics* > *Nodes*.
2. Right-click *Nodes*, then select *Create Node Style*.

   Although node style names can contain spaces, because of encoding of source code, note that names with no spaces are easier to read if you are planning to work directly in the source code.
3. Click the *Layout* tab to open the *Layout* view.

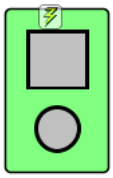
   A simple default object displays. The *Layout* view is in Edit mode by default.

4. Right-click the drawing background, then select *Clear Drawing* to start building the node from scratch.

   The default object is removed.

5 Right-click the Layout view background, then select *Customize* > *Container* > *Add* > *Stack Layout Container*. A stack layout container shows in the view:



A stack container "stacks" one or more objects in a container. Each object is directly on top of the other with the order determine by the order that objects are added (first object is on top). The background of the stack container has a bind rule, which tells it to update background color based on current element condition.

6 Click *View Source* to view the source code in the layout editor:

```
<?xml version="1.0" encoding="ISO-8859-1"?><svg xmlns:xlink="http://
www.w3.org/1999/xlink" xmlns="http://www.w3.org/2000/svg" xmlns:bind="http://
www.managedobjects.com/bind" bind:setup="1151445190011" MOS_PanX="448"
contentScriptType="text/ecmascript" zoomAndPan="magnify" MOS_PanY="232"
contentStyleType="text/css" dname="graphic=ToggleIndicator/
graphicsFolder=Development/graphicsFolder=Nodes/graphics=Graphics/
root=Administration" version="1.0" width="1000.0" dname_relid="1"
preserveAspectRatio="xMidYMid meet" height="1000.0" MOS_World="1"
MOS_Scale="1.0" overflow="visible">
  <defs/>
```

The next section of code is the opening tag for the container and default container attributes:

```
  <g transform="translate(-158.0,114.0)">
    <bind:layout minWrapHeight="30" wrap="true" pad="15" id="1225bc948ab"
gap="10" name="grid" align="center" columns="1" minWrapWidth="30">
      <bind:value default="stack" property="condition" set="stack"
default.edit="grid" attr="name" ignoregenerated="true" set.edit="grid"
when="Unknown,Critical,Major,Minor,Info,OK,Unmanaged"/>
    </bind:layout>
```

Now, the top object is formed (square shape):

```
    <rect x="-11.5" y="-11.5" fill="rgb(128,255,128)" width="78"
undofill="lightgray" rx="4" opacity="1" bind:container="true" height="124"
ry="4" stroke="black" stroke-width="1.5">
      <bind:fill property="condition" default="" set="" when=""/>
    </rect>
    <rect gautolayout="1225bc948ab" stroke="#000000" stroke-linecap="butt"
transform="matrix(1, 0, 0, 1, 5, 5)" width="45" fill="#c0c0c0" stroke-
dasharray="none" filter="none" stroke-width="3" height="43" opacity="1" x="0"
stroke-linejoin="miter" y="0"/>
```

And then, the bottom object (round shape):

```
    <circle transform="matrix(1, 0, 0, 1, 27.5, 79.5)" fill="#c0c0c0"
gautolayout="1225bc948ab" filter="none" cx="0" stroke-dasharray="none"
r="16.5" opacity="1" cy="0" stroke="#000000" stroke-width="3"/>
  </g>
```

And lastly, the closing tag for the outer container:

```
</svg>
```

7 Click *Back to Preview* to work with objects in the layout editor.

8 Right-click the container, then select *Change Group* to allow the objects inside the container to be modified.

9 To delete the default shapes inside the container, select each shape, then click *Delete*.

The source code updates as shown:

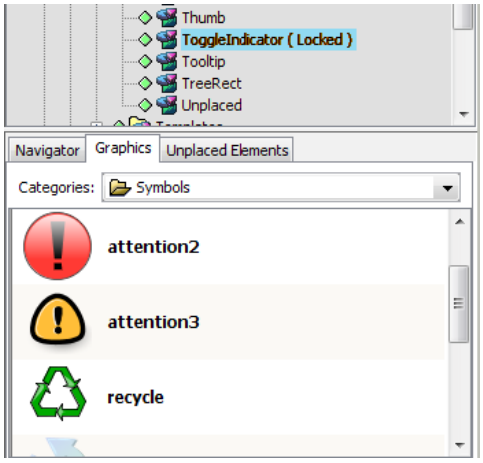
```
<?xml version="1.0" encoding="ISO-8859-1"?><svg xmlns:bind="http://
www.managedobjects.com/bind" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns="http://www.w3.org/2000/svg" MOS_PanX="0" contentScriptType="text/
ecmascript" zoomAndPan="magnify" MOS_PanY="0" contentStyleType="text/css"
dname="graphic=ToggleIndicator/graphicsFolder=Development/
graphicsFolder=Nodes/graphics=Graphics/root=Administration" version="1.0"
preserveAspectRatio="xMidYMid meet" MOS_World="1" MOS_Scale="1.0"
overflow="visible">
```

With only the container code remaining (the 2 objects are no longer there):

```
  <g transform="translate(461.0,334.0)">
    <bind:layout minWrapHeight="30" wrap="true" xmlns:bind="http://
www.managedobjects.com/bind" pad="15" id="12260bc2b1c" gap="10" name="grid"
align="center" columns="1" minWrapWidth="30">
      <bind:value default="stack" property="condition" set="stack"
default.edit="grid" attr="name" ignoregenerated="true" set.edit="grid"
when="Unknown,Critical,Major,Minor,Info,OK,Unmanaged"/>
    </bind:layout>
    <rect x="-10.0166" y="-13.21" fill="lightgray" width="141.033"
xmlns:bind="http://www.managedobjects.com/bind" rx="4" opacity="1"
bind:container="true" height="124.6642" ry="4" stroke="black" stroke-
width="1.5"/>
  </g>
  <defs/>
</svg>
```

10 Select graphics from the clipart library and add them to the container.

10a Click the *Graphics* tab to open the graphics selector panel:



10b Locate the desired clipart image.

For this example, we chose the recycle, attention3 and attention2 images in the *Symbols* folder.

10c Click and drag each image from the graphics panel into the layout view background (do not try to drop directly on top of the container).

The graphics are automatically added to the container:



The source code updates as shown here. The code adds the three new graphics into the container just beneath it's `<rect></rect>` tag:

```
  <g transform="translate(321.0,299.0)">
    <bind:layout minWrapHeight="30" wrap="true" xmlns:bind="http://
www.managedobjects.com/bind" pad="15" id="122c25d2b41" gap="10" name="grid"
align="center" columns="1" minWrapWidth="30">
      <bind:value default="stack" property="condition" set="stack"
default.edit="grid" attr="name" set.edit="grid"
when="Unknown,Critical,Major,Minor,Info,OK,Unmanaged"/>
    </bind:layout>
    <rect bind:container="true" stroke="black" width="141.033"
fill="rgb(128,255,128)" stroke-width="1.5" height="315.3989"
xmlns:bind="http://www.managedobjects.com/bind" undofill="lightgray" x="-
10.0166" rx="4" opacity="1" y="-8.5239" ry="4">
      <bind:fill property="condition" default="" set="" when=""/>
    </rect>
```

The following section contains Object 1, recycle:

```
    <g transform="matrix(1, 0, 0, 1, 10.522090911865234,
0.692075252532959)" display="inline" instantiated="122c25d2b42"
bind:marks="Clipart" gautolayout="122c25d2b41" xmlns:bind="http://
www.managedobjects.com/bind" id="122c25d2b42_clipart" ghash="19c484af"
bind:graphicsource="clipart">
      <image width="100" xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="Clipart__Symbols__recycle" xlink:type="simple"
xlink:actuate="onLoad" height="100" preserveAspectRatio="xMidYMid meet"
xlink:show="embed" overflow="visible"/>
    </g>
```

The following section contains Object 2, attention3:

```
    <g transform="matrix(1, 0, 0, 1, 5.220731258392334, 93.2699203491211)"
display="inline" instantiated="122c25d2b44" bind:marks="Clipart"
gautolayout="122c25d2b41" id="122c25d2b44_clipart" ghash="fe87ffe9"
bind:graphicsource="clipart">
      <image width="100" xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="Clipart__Symbols__attention3" xlink:type="simple"
xlink:actuate="onLoad" height="100" preserveAspectRatio="xMidYMid meet"
xlink:show="embed" overflow="visible"/>
    </g>
```

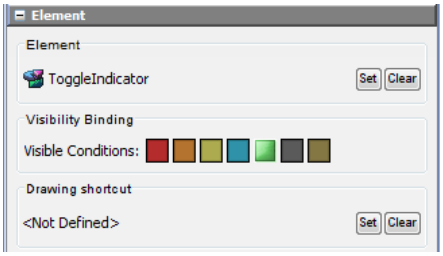
The following section contains Object 3, attention2:

```
    <g transform="matrix(1, 0, 0, 1, 20.500015258789062,
212.49998474121094)" display="inline" instantiated="122c25d2b45"
bind:marks="Clipart" gautolayout="122c25d2b41" id="122c25d2b45_clipart"
ghash="7d8870b4" bind:graphicsource="clipart">
      <image width="100" xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="Clipart__Symbols__attention2" xlink:type="simple"
xlink:actuate="onLoad" height="100" preserveAspectRatio="xMidYMid meet"
xlink:show="embed" overflow="visible"/>
    </g>
  </g>
```

11 Set the condition binds for each graphic.

11a Select a graphic object inside the container.

The object is selected and Layout palette panels display:



For instance, we'll select the recycle graphic first.

11b In the Palette panel (*Element* section), deselect any conditions when the recycle graphic is to be hidden.

In this case, we deselect all conditions except *OK* (green).

11c Repeat for all three graphics to set the desired conditions where the graphic is to be visible.

The source code updates as shown here. Under each graphic object, new bind rules are added for visibility. The first is hidden for all conditions except OK, the second is hidden when condition is OK or Critical, and the third is hidden for all conditions except Critical.

```
    <g transform="matrix(1, 0, 0, 1, 10.522090911865234,
0.692075252532959)" display="inline" instantiated="122c25d2b42"
bind:marks="Clipart" gautolayout="122c25d2b41" xmlns:bind="http://
www.managedobjects.com/bind" id="122c25d2b42_clipart" ghash="19c484af"
bind:graphicsource="clipart">
      <image width="100" xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="Clipart__Symbols__recycle" xlink:type="simple"
xlink:actuate="onLoad" height="100" preserveAspectRatio="xMidYMid meet"
xlink:show="embed" overflow="visible"/>
```

The new bind rule for visibility:

```
    <bind:value default="inline" property="condition"
MOS_bindtype="visibility" set="none" default.edit="inline" attr="display"
set.edit="inline" when="Unknown,Critical,Major,Minor,Info,Unmanaged"/>
    </g>

    <g transform="matrix(1, 0, 0, 1, 5.220731258392334, 93.2699203491211)"
display="inline" instantiated="122c25d2b44" bind:marks="Clipart"
gautolayout="122c25d2b41" id="122c25d2b44_clipart" ghash="fe87ffe9"
bind:graphicsource="clipart">
      <image width="100" xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="Clipart__Symbols__attention3" xlink:type="simple"
xlink:actuate="onLoad" height="100" preserveAspectRatio="xMidYMid meet"
xlink:show="embed" overflow="visible"/>
```

Another bind rule for visibility is added here:

```
   <bind:value default="inline" property="condition"
MOS_bindtype="visibility" set="none" default.edit="inline" attr="display"
set.edit="inline" when="Critical,OK"/>
    </g>

    <g transform="matrix(1, 0, 0, 1, 20.500015258789062,
212.49998474121094)" display="inline" instantiated="122c25d2b45"
bind:marks="Clipart" gautolayout="122c25d2b41" id="122c25d2b45_clipart"
ghash="7d8870b4" bind:graphicsource="clipart">
      <image width="100" xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="Clipart__Symbols__attention2" xlink:type="simple"
xlink:actuate="onLoad" height="100" preserveAspectRatio="xMidYMid meet"
xlink:show="embed" overflow="visible"/>
```

And, another is added here:

```
   <bind:value default="inline" property="condition"
MOS_bindtype="visibility" set="none" default.edit="inline" attr="display"
set.edit="inline" when="Unknown,Major,Minor,Info,OK,Unmanaged"/>
    </g>
```

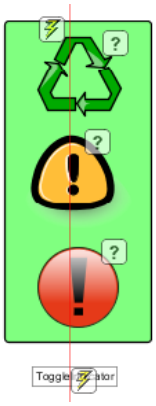
12 Right-click the Layout view background, then select *Save Group*.

The container group is saved and the Layout view background switches to white.

13 To add element name label to the node display, right-click the Layout background, then select *Customize* > *Add Caption*.

In this step we'll add a caption, which automatically updates with the element's name.

A caption label is added:



The source code updates as shown here. This code adds a `caption` tag directly under the node `container` tag but before the closing `</SVG>` tag:

```
   <?xml version="1.0" encoding="ISO-8859-1" standalone="no"?><svg
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns="http://www.w3.org/2000/svg"
xmlns:bind="http://www.managedobjects.com/bind" MOS_PanX="181"
contentScriptType="text/ecmascript" zoomAndPan="magnify" MOS_PanY="136"
contentStyleType="text/css" dname="graphic=ToggleIndicator/
graphicsFolder=Nodes/graphics=Graphics/root=Administration" version="1.0"
width="142.5330047607422" dname_relid="1" MOS_viewBox="300 279 162 336"
preserveAspectRatio="xMidYMid meet" height="316.8988952636719" MOS_World="0"
MOS_Scale="1.0" overflow="visible">
<g transform="matrix(1, 0, 0, 1, 10.522090911865234, 0.692075252532959)"
display="inline" instantiated="122c25d2b42" bind:marks="Clipart"
gautolayout="122c25d2b41" xmlns:bind="http://www.managedobjects.com/bind"
id="122c25d2b42_clipart" ghash="19c484af" bind:graphicsource="clipart">
      <image width="100" xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="Clipart__Symbols__recycle" xlink:type="simple"
xlink:actuate="onLoad" height="100" preserveAspectRatio="xMidYMid meet"
xlink:show="embed" overflow="visible"/>
      <bind:value default="inline" property="condition"
MOS_bindtype="visibility" set="none" default.edit="inline" attr="display"
set.edit="inline" when="Unknown,Critical,Major,Minor,Info,Unmanaged"/>
    </g>
    <g transform="matrix(1, 0, 0, 1, 5.220731258392334, 93.2699203491211)"
display="inline" instantiated="122c25d2b44" bind:marks="Clipart"
gautolayout="122c25d2b41" id="122c25d2b44_clipart" ghash="fe87ffe9"
bind:graphicsource="clipart">
      <image width="100" xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="Clipart__Symbols__attention3" xlink:type="simple"
xlink:actuate="onLoad" height="100" preserveAspectRatio="xMidYMid meet"
xlink:show="embed" overflow="visible"/>
      <bind:value default="inline" property="condition"
MOS_bindtype="visibility" set="none" default.edit="inline" attr="display"
set.edit="inline" when="Critical,OK"/>
    </g>
    <g transform="matrix(1, 0, 0, 1, 20.500015258789062, 212.49998474121094)"
display="inline" instantiated="122c25d2b45" bind:marks="Clipart"
gautolayout="122c25d2b41" id="122c25d2b45_clipart" ghash="7d8870b4"
bind:graphicsource="clipart">
      <image width="100" xmlns:xlink="http://www.w3.org/1999/xlink"
xlink:href="Clipart__Symbols__attention2" xlink:type="simple"
xlink:actuate="onLoad" height="100" preserveAspectRatio="xMidYMid meet"
xlink:show="embed" overflow="visible"/>
      <bind:value default="inline" property="condition"
MOS_bindtype="visibility" set="none" default.edit="inline" attr="display"
set.edit="inline" when="Unknown,Major,Minor,Info,OK,Unmanaged"/>
    </g>
```

The caption is added here above the container closing SVG tag:

```
 <g gautolayout="122c25d2b53" transform="translate(56.2292,252.638)">
      <rect x="-15.7293" y="53.862" fill="white" text-anchor="middle"
width="81.7163" bind:container="true" height="18.289" stroke="gray"/>
      <bind:layout name="wrap" pad="0" id="122c25d2b52"/>
      <text x="25.0" font-size="11px" y="67.0" fill="black" text-
anchor="middle" gautolayout="122c25d2b52" stroke="none"
xml:space="preserve">ToggleIndicator<bind:value property="name"/></text>
    </g>
</svg>
```

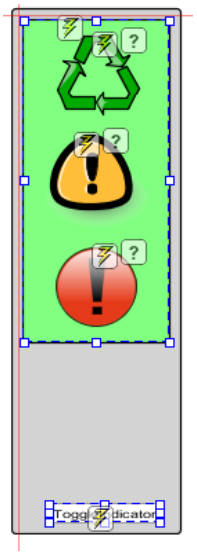
14 Position the caption to display at the edge of the node box.

You can force the caption to overlay the bottom of the node box by wrapping both the caption and the node container in an outer container, and then update the vertical gap to a negative value.

14a Do one of the following:

- Click and drag on the Layout background to select both the container and the caption.
- Press Shift while clicking to select each object, one at a time, until both are selected.

14b Right-click over one of the selected objects, then select *Customize* > *Container* > *Wrap in Container*:

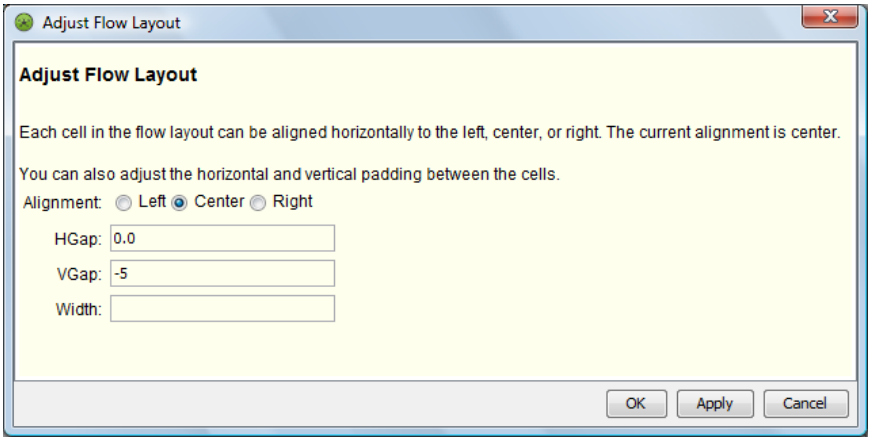


14c Right-click the new container, then select *Customize* > *Container* > *Layout* > *Set Layout* to open the Set Layout dialog box.

14d Select *Flow* from the drop-down list.

The extra space is removed from between the objects in the container.

14e Right-click the new container, then select *Customize* > *Container* > *Layout* > *Adjust Flow Layout* to open the Adjust Flow Layout dialog box.

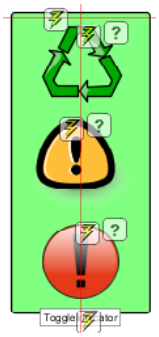


14f Update the *VGap* to `-5` to overlap the caption with the node container.

14g Click *Apply* to save and apply the setting, or click *OK* to save the setting and exit the dialog box.

14h Right-click the new container, then select *Customize* > *Container* > *Wrapping* > *Hide Wrapping Rectangle* to hide the outer container when the node style is in use.

The above selections result in the following configuration:



The source code updates as shown here. The code highlighted in yellow, show the new outer container that now groups the node container and caption (which is still below the node container):

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?><svg
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns="http://www.w3.org/2000/
svg" xmlns:bind="http://www.managedobjects.com/bind" MOS_PanX="181"
contentScriptType="text/ecmascript" zoomAndPan="magnify" MOS_PanY="136"
contentStyleType="text/css" dname="graphic=ToggleIndicator/
graphicsFolder=Nodes/graphics=Graphics/root=Administration" version="1.0"
width="142.5330047607422" dname_relid="1" MOS_viewBox="300 279 162 336"
preserveAspectRatio="xMidYMid meet" height="316.8988952636719"
MOS_World="0" MOS_Scale="1.0" overflow="visible">
  <defs/>
```

The new outer container starts here:

```
  <g>
    <bind:layout width="" wrap="true" pad="10" vgap="-5" id="122c25d2b53"
hgap="0.0" name="flow" gap="10" columns="1" align=""/>
    <rect x="-1.0E-4" y="-15" fill="lightgray" width="162.533" rx="4"
bind:container="true" height="350.289" ry="4" stroke="black" stroke-
width="1.5"/>

    <g gautolayout="122c25d2b53" transform="translate(20.7665,4.2739)">
     <bind:layout minWrapHeight="30" wrap="true" pad="15" id="122c25d2b41"
gap="10" name="grid" align="center" columns="1" minWrapWidth="30">
       <bind:value default="stack" property="condition" set="stack"
default.edit="grid" attr="name" set.edit="grid"
when="Unknown,Critical,Major,Minor,Info,OK,Unmanaged"/>
      </bind:layout>
      <rect bind:container="true" stroke="black" width="141.033"
fill="rgb(128,255,128)" stroke-width="1.5" height="315.3989"
undofill="lightgray" x="-10.0166" rx="4" opacity="1" y="-8.5239" ry="4">
       <bind:fill property="condition" default="" set="" when=""/>
      </rect>
      <g transform="matrix(1, 0, 0, 1, 10.522090911865234,
0.692075252532959)" display="inline" instantiated="122c25d2b42"
bind:marks="Clipart" gautolayout="122c25d2b41" id="122c25d2b42_clipart"
ghash="19c484af" bind:graphicsource="clipart">
       <image width="100" xlink:href="Clipart__Symbols__recycle"
xlink:type="simple" xlink:actuate="onLoad" height="100"
preserveAspectRatio="xMidYMid meet" xlink:show="embed" overflow="visible"/
>
       <bind:value default="inline" property="condition"
MOS_bindtype="visibility" set="none" default.edit="inline" attr="display"
```

```
set.edit="inline" when="Unknown,Critical,Major,Minor,Info,Unmanaged"/>
      </g>
      <g transform="matrix(1, 0, 0, 1, 5.220731258392334,
93.2699203491211)" display="inline" instantiated="122c25d2b44"
bind:marks="Clipart" gautolayout="122c25d2b41" id="122c25d2b44_clipart"
ghash="fe87ffe9" bind:graphicsource="clipart">
        <image width="100" xlink:href="Clipart__Symbols__attention3"
xlink:type="simple" xlink:actuate="onLoad" height="100"
preserveAspectRatio="xMidYMid meet" xlink:show="embed" overflow="visible"/
>
        <bind:value default="inline" property="condition"
MOS_bindtype="visibility" set="none" default.edit="inline" attr="display"
set.edit="inline" when="Critical,OK"/>
      </g>
      <g transform="matrix(1, 0, 0, 1, 20.500015258789062,
212.49998474121094)" display="inline" instantiated="122c25d2b45"
bind:marks="Clipart" gautolayout="122c25d2b41" id="122c25d2b45_clipart"
ghash="7d8870b4" bind:graphicsource="clipart">
        <image width="100" xlink:href="Clipart__Symbols__attention2"
xlink:type="simple" xlink:actuate="onLoad" height="100"
preserveAspectRatio="xMidYMid meet" xlink:show="embed" overflow="visible"/
>
        <bind:value default="inline" property="condition"
MOS_bindtype="visibility" set="none" default.edit="inline" attr="display"
set.edit="inline" when="Unknown,Major,Minor,Info,OK,Unmanaged"/>
      </g>
    </g>
```

The following is the caption showing element name:

```
    <g gautolayout="122c25d2b53" transform="translate(56.2292,252.638)">
      <rect x="-15.7293" y="53.862" fill="white" text-anchor="middle"
width="81.7163" bind:container="true" height="18.289" stroke="gray"/>
      <bind:layout name="wrap" pad="0" id="122c25d2b52"/>
      <text x="25.0" font-size="11px" y="67.0" fill="black" text-
anchor="middle" gautolayout="122c25d2b52" stroke="none"
xml:space="preserve">ToggleIndicator<bind:value property="name"/></text>
    </g>
```

The closing tag for the new outer container:

```
  </g>
</svg>
```

15 To test that the bind rules for condition will function on the node style as expected, do the following:

15a Apply the node style to a layout for a test Service Model.

This can be done easily by dragging and dropping the node style onto the service model's Layout view.

15b Change the state of the service model using a condition algorithm or another method to verify the node style functions as necessary.

C.5 Looking at the Source Code with a Child Container Example

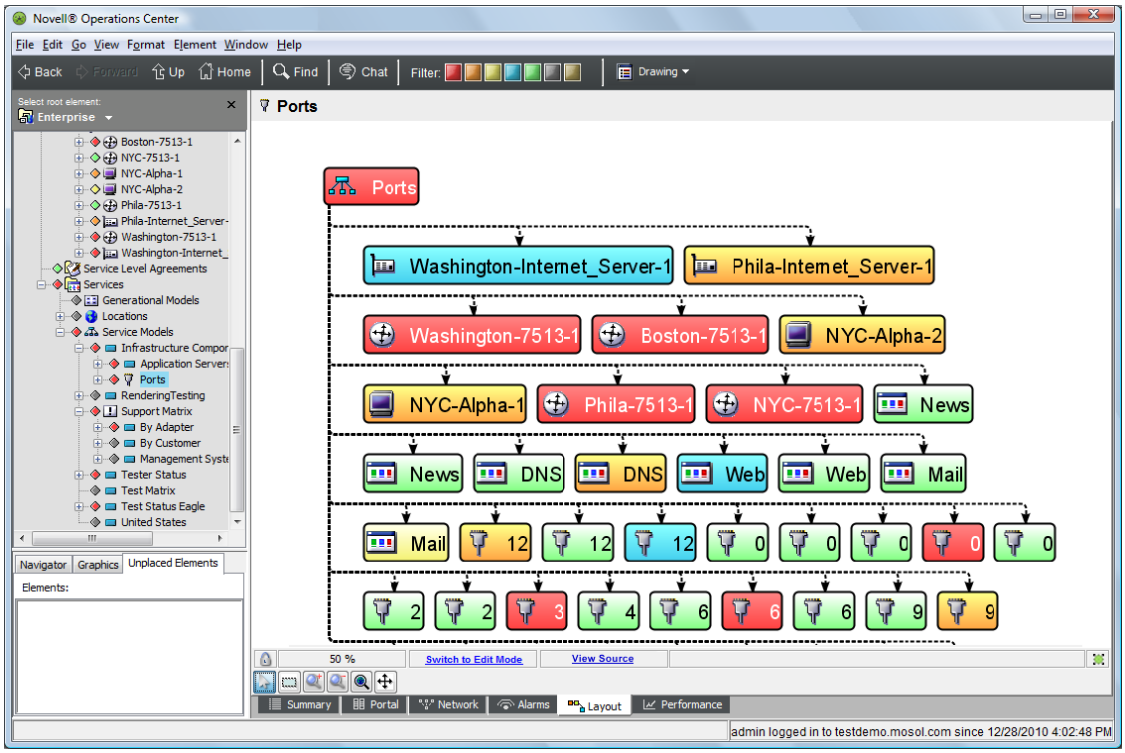
In this example, we want change the Layout view so that applies a filter and shows remaining elements by condition, using a custom node style. We'll take a look at what happens as the source code and bind commands are built in the background. Generally, here is what we'll do:

- Set up a child container
- Apply a filter that shows only ports and sort by condition
- Apply a custom node style

To build a drawing using a child container and apply a custom node:

1 In the *Explorer* pane, navigate to the desired element under *Elements* or *Service Models*, then open the Layout View.

In this example, we used a service model with an element matcher:

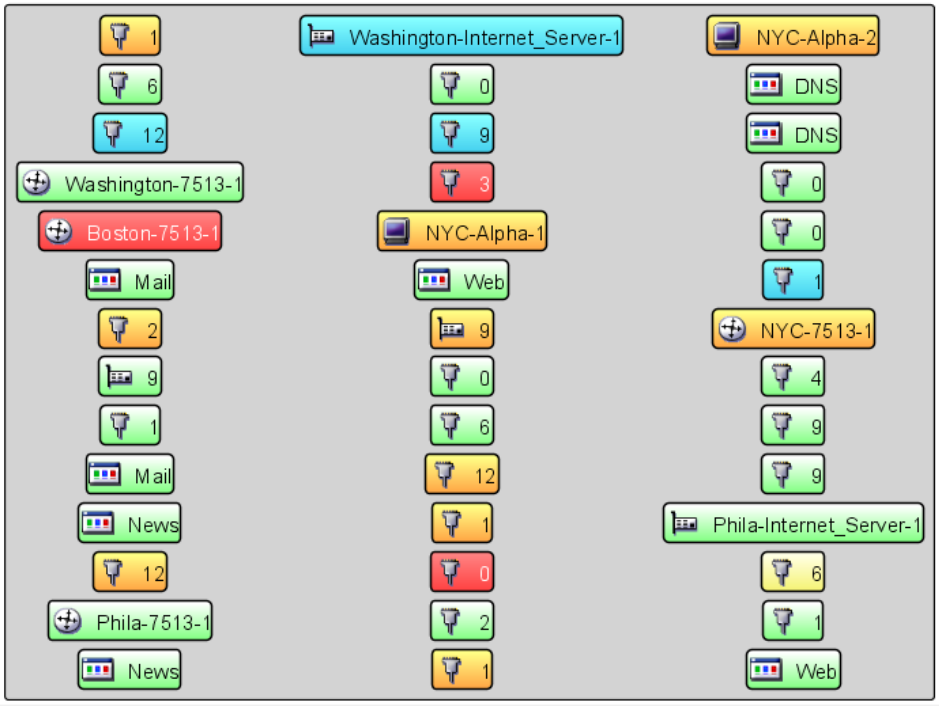


2 Right-click the background of the Layout View, then select *New Drawing*.

All elements are removed leaving the Layout view empty and in Edit Mode.

3 Right-click the Layout View, then select *Customize* > *Container* > *Add* > *Children Container*.

A child container displaying all elements displays:



The source code updates as shown here to add a container that uses a `bind:layout` tag inside the `bind:chidren` tag to layout all children. Notice that code for generated elements is removed from this code sample so that it is easier to see the basic structure of what is added.

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?><svg
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns="http://www.w3.org/2000/svg"
xmlns:bind="http://www.managedobjects.com/bind" bind:setup="1252087240607"
MOS_PanX="-217" contentScriptType="text/ecmascript" zoomAndPan="magnify"
MOS_PanY="-155" contentStyleType="text/css" dname="org=Ports/org=demo/
root=Organizations" version="1.0" preserveAspectRatio="xMidYMid meet"
MOS_World="0" MOS_Scale="0.75" overflow="visible">
  <g transform="translate(306.0,229.0)">
    <bind:layout name="wrap" pad="10" id="12385dab0e9"/>
    <rect x="40.5589" y="-6" fill="lightgray" width="909.9905" rx="4"
bind:container="true" height="684" ry="4" stroke="black" stroke-width="1.5"/>
    <bind:children self="false" id="12385dab0be">
      <bind:layout gap="10" name="grid" columns="3" id="12385dab0e8"
align="center"/>
    </bind:children>
  </g>
  <defs>
    <rect fill="none" width="32" height="32" id="emptyIcon" stroke="gray"/>
  </defs>
</svg>
```

4 Filter the elements shown.

In this step we'll create a filter that shows only ports.

Do the following:

4a Right-click the container, then select *Customize* > *Child Container* > *Filtering* > *Adjust Filtering* to open the Adjust Filtering dialog box.

4b In the *Class Name* field, enter `port.*` to find only those elements with a class beginning with port.

4c Click *OK*.

The view updates to only show port elements:



Filters in `bind` tags check the next level of elements only if the parent element passes the filter. Regular expressions can be used to define filtering matcher as in the above example.

The source code updates as shown here to add the class filter to the `bind:children` tag:

```
<bind:children classFilter="port.*"
conditionFilter="OK,INFO,MINOR,MAJOR,CRITICAL,UNKNOWN" self="false"
nameFilter="" scriptFilter="" dnameFilter="" id="12385dab0be">
      <bind:layout gap="10" name="grid" columns="3" id="12385dab0e8"
align="center"/>
    </bind:children>
```

5 Adjust the display, as desired.

In this example, we'll adjust the layout and sort by condition.

Do the following:

5a Right-click the container, then select *Customize* > *Child Container* > *Layout* > *Adjust Layout* to open the Adjust Layout dialog box.

5b Select the *4* radio button from *Columns*.

5c Click *OK* to close the Adjust Layout dialog box.

5d Right-click the container, then select *Customize* > *Child Container* > *Layout* > *Sorting* to open the Adjust Sorting dialog box.

5e Select the *Ascending* radio button from *Sort Direction*.

5f Select the *Condition* radio button from *Sort By*.

5g Click *OK*.

The Adjust Sorting dialog box closes:



The source code updates as shown here to adjust the `bind:layout` tag inside the `bind:children` tag:

```
    <bind:children classFilter="port.*"
conditionFilter="OK,INFO,MINOR,MAJOR,CRITICAL,UNKNOWN" self="false"
nameFilter="" scriptFilter="" dnameFilter="" id="12385dab0ea">
      <bind:layout id="12385dab114" columnWidth="" rowHeight="" gap="10"
name="grid" align="center" columns="4" sortascend="condition"/>
    </bind:children>
  </g>
```

6 To update with the desired node style, do the following:

6a Right-click the container, then select *Change Group*.

6b From the *Explorer* pane, expand *Administration* > *Graphics* > *Nodes*.

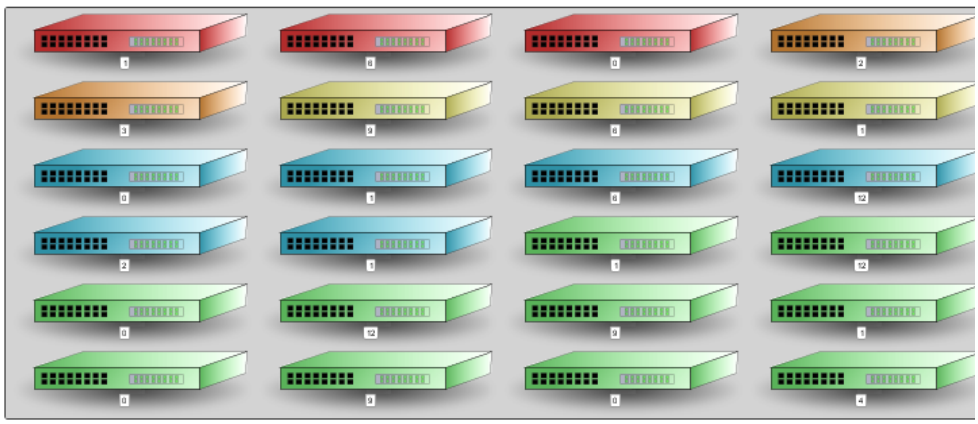
6c Navigate within the *Nodes* folder until you find the desired node style.

TIP: Be careful not to select any of the objects under *Nodes*, else the Layout view changes focus and leaves the Layout drawing you are working with.

6d Drag the selected node style over onto one of the elements inside the Layout view container.

6e Select *Change Node Style for All Nodes* from the pop-up menu.

All element nodes update with the new node style:



6f Right-click the container, then select *Save Group*.

The source code updates as shown here to add the `bind:ambient` tag inside the main `SVG` tag:

```
<?xml version="1.0" encoding="ISO-8859-1" standalone="no"?><svg
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns="http://www.w3.org/2000/
svg" xmlns:bind="http://www.managedobjects.com/bind"
bind:setup="1252088279618" MOS_PanX="-65" contentScriptType="text/
ecmascript" zoomAndPan="magnify" MOS_PanY="-14" contentStyleType="text/
css" dname="org=Ports/org=demo/root=Organizations" version="1.0"
width="1468.310546875" dname_relid="1" MOS_viewBox="120 136 1488 631"
preserveAspectRatio="xMidYMid meet"
bind:ambient="Nodes__Technology__Networking__Switch"
height="611.6928100585938" MOS_World="0" MOS_Scale="0.5"
overflow="visible">
```

Glossary of Terms

ambient node style. Changes the node style for all elements currently using the system default node style. Drop a node style on the drawing background to quickly change multiple nodes at once.

automatic binding. The process of automatically linking child elements to a Layout drawing associated with their parents.

automatic layout. The system default layout drawing, which uses the default Gradient Bubble node style and a traditional tree layout style.

binding. The process of linking an element condition or property to a component of a drawing. Changes made to the element automatically update the linked drawing component. See also color binding and content binding, and automatic binding and manual binding.

color binding. A feature that links an element condition color to the fill or line color of a drawing shape. The fill/line color updates automatically when the element condition changes.

content binding. A feature that links an element property (name, DName, attribute) to text in a drawing. If the property changes, the text updates automatically.

custom graphic. A graphic specified in the Element Graphic drawing channel is used to represent an element in a drawing.

drawing component. Any item displayed in the Layout view that was added using the drawing toolbar (shapes, arrows, text, and so on), or is an imported graphic or clipart, or was added by binding an element to the drawing.

drawing properties. Each drawing component has a set of properties displayed as a set of panes in the Palette.

editing channel. Enables switching between the Layout drawing for an element and other default drawing-related settings for the element: *Element Graphic*, *Element Node Style*, or *Element Drawing Template*.

element graphic. An element in a drawing can have a custom graphic defined. This graphic is specified in the element's Element Graphic editing channel. This graphic could have been created by a user in v4.0, or results from a conversion from v3.5.*x*.

grouping. Combines multiple elements into a single element. Grouping is useful when rearranging or managing many elements in a drawing.

manual binding. The action of dragging and dropping an element to the Layout drawing, and optionally linking, copying, or moving the element to the element associated with the drawing.

navigator. Component of the Layout view that preserves a "bird's eye" view of the drawing as you zoom and move.

node style. The graphic representation of an element in a Layout drawing. Node styles display under *Administration* > *Graphics* > *Nodes* in the *Explorer* pane.

palette. A Layout view component that displays editable properties of the item currently selected in the drawing. These property panes include line widths and colors, text fonts, and binding settings.

templates. Applies a set of node styles and layout pattern to the current drawing. Templates can save time when there is a need to create the same type of layout multiple times. Templates display under *Administration* > *Graphics* in the *Explorer* pane.

ungrouping. Separates a drawing component into the original components (before the `Group` command was used), such as countries in a continent map. Useful when you need to edit a portion of the drawing.