

Technical Document

Niagara Graphics Guide

October 4, 2021

niagara⁴

Niagara Graphics Guide

Tridium, Inc.

3951 Westerre Parkway, Suite 350
Richmond, Virginia 23233
U.S.A.

Confidentiality

The information contained in this document is confidential information of Tridium, Inc., a Delaware corporation ("Tridium"). Such information and the software described herein, is furnished under a license agreement and may be used only in accordance with that agreement.

The information contained in this document is provided solely for use by Tridium employees, licensees, and system owners; and, except as permitted under the below copyright notice, is not to be released to, or reproduced for, anyone else.

While every effort has been made to assure the accuracy of this document, Tridium is not responsible for damages of any kind, including without limitation consequential damages, arising from the application of the information contained herein. Information and specifications published here are current as of the date of this publication and are subject to change without notice. The latest product specifications can be found by contacting our corporate headquarters, Richmond, Virginia.

Trademark notice

BACnet and ASHRAE are registered trademarks of American Society of Heating, Refrigerating and Air-Conditioning Engineers. Microsoft, Excel, Internet Explorer, Windows, Windows Vista, Windows Server, and SQL Server are registered trademarks of Microsoft Corporation. Oracle and Java are registered trademarks of Oracle and/or its affiliates. Mozilla and Firefox are trademarks of the Mozilla Foundation. Echelon, LON, LonMark, LonTalk, and LonWorks are registered trademarks of Echelon Corporation. Tridium, JACE, Niagara Framework, and Sedona Framework are registered trademarks, and Workbench are trademarks of Tridium Inc. All other product names and services mentioned in this publication that are known to be trademarks, registered trademarks, or service marks are the property of their respective owners.

Copyright and patent notice

This document may be copied by parties who are authorized to distribute Tridium products in connection with distribution of those products, subject to the contracts that authorize such distribution. It may not otherwise, in whole or in part, be copied, photocopied, reproduced, translated, or reduced to any electronic medium or machine-readable form without prior written consent from Tridium, Inc.

Copyright © 2021 Tridium, Inc. All rights reserved.

The product(s) described herein may be covered by one or more U.S. or foreign patents of Tridium.

Contents

| | |
|--|-----------|
| About this guide | 9 |
| Document change log | 9 |
| Related documentation | 10 |
| Chapter 1 About Presentation XML (Px)..... | 11 |
| What's new in graphics | 12 |
| About Px views | 12 |
| About types of Px target media | 13 |
| About the Px Viewer..... | 14 |
| About Zoom capability in Px Editor | 15 |
| Zooming in on a Px Page | 15 |
| Zooming out on a Px Page..... | 16 |
| Resetting zoom on Px Page | 16 |
| About Px files..... | 16 |
| Shared Px files..... | 17 |
| About Widgets..... | 17 |
| Chapter 2 Creating a Px view canvas | 19 |
| Px views as slot properties..... | 20 |
| About Px Layers | 20 |
| Grouping Px Layers | 20 |
| Adding a Px Layer | 21 |
| Assigning/Unassigning objects to a layer..... | 21 |
| Renaming a Px Layer | 21 |
| Deleting a Px Layer | 21 |
| Chapter 3 Adding widgets using the Make Widget wizard | 23 |
| Widget painting | 24 |
| Widget commands | 25 |
| Making Bound Label widgets | 25 |
| Making Chart widgets | 29 |
| Hyperlinks with Popup Bindings (an example)..... | 32 |
| Creating a scalable image using the Picture widget..... | 32 |
| Making component Properties-based widgets..... | 33 |
| Make Palette kit-based widgets | 35 |
| Making Time Plot widgets | 39 |
| Making view-based widgets..... | 41 |
| Making Action widgets..... | 44 |
| About Widget sizing in Px Editor..... | 46 |
| Embedding a Px View in another Px View | 46 |
| Example of a PxInclude widget with a single ORD variable | 47 |
| Example of PxInclude Widgets with multiple ORD variables | 48 |
| Chapter 4 Animating graphics (data binding)..... | 49 |
| About data binding | 49 |
| Add a data binding to a widget..... | 50 |

- Animate a widget property 50
- Animate using static SVG images 51
- Relativize absolute Ords 52
- Types of data bindings..... 53
 - Types of binding properties 54
 - About bound label bindings..... 56
 - About value bindings..... 56
 - About spectrum bindings 57
 - About setpoint bindings 58
 - About increment setpoint bindings 58
 - About spectrum setpoint bindings 59
 - About action bindings 59
 - About table bindings..... 60
 - About field editor bindings..... 60
 - About Popup Bindings..... 60
 - About relative and absolute bindings 61
 - About tag-based NEQL bindings 62
- Chapter 5 Other elements to add to Px views 65**
 - Add and configure a comment box 65
 - Add a text entry field and save button..... 66
 - Embed a History Table in your Px view 67
 - Embed a history chart in a Px view 67
 - Add WeatherReports on Px Views..... 68
 - Launching a History Chart Builder in a Px view 68
 - Applying visual styles to Px views..... 69
 - Customizing the login screen 69
 - Optimizing existing Px files for mobile devices 70
 - Converting bound ORDs to tag-based NEQL ORDs..... 71
 - Using the visible property in Hx profile (an example)..... 72
- Chapter 6 Build navigation files 75**
 - Creating a new nav files 75
 - Deleting nav files 75
 - Renaming nav files..... 75
 - Editing nav files 75
 - Open nav files 76
 - Add nodes in nav files..... 76
 - Delete nodes in nav files 76
 - Edit node hierarchy 77
 - Edit node properties 77
 - Create a global navigation menu using PxInclude 77
 - Embed Px files with ord variables using PxInclude..... 78
- Chapter 7 Generate reports 81**
 - Reporting process workflow 81
 - Set up the reporting service..... 81
 - Set up report data in a ComponentGrid 82

| | |
|--|-----------|
| Creating a ReportPxFile..... | 82 |
| Configuring ExportSource component | 83 |
| Configuring the EmailRecipient..... | 83 |
| Chapter 8 About profiles | 85 |
| About Web Profiles | 85 |
| Velocity Doc Web Profile | 86 |
| Basic Hx Profile | 86 |
| Default Hx Profile..... | 87 |
| Handheld Hx Profile | 88 |
| HTML5 Hx Profile | 88 |
| Default Mobile Web Profile..... | 94 |
| Basic Wb Web Profile | 96 |
| Default Wb Web Profile | 96 |
| Handheld Wb Web Profile | 97 |
| Simple Admin Wb Web Profile | 97 |
| About Kiosk Profiles | 97 |
| Basic Kiosk Profile | 98 |
| Default Kiosk Profile..... | 98 |
| Handheld Kiosk Profile | 98 |
| Chapter 9 Px graphics reference..... | 99 |
| Px view component property sheet | 99 |
| New Px View window | 100 |
| About Px Editor | 102 |
| About Px Editor canvas..... | 103 |
| About alignment and distribution tools | 104 |
| About Px Properties | 104 |
| More About Px Layers | 107 |
| About SVG support | 108 |
| About SVG rendering | 109 |
| Browser compatible SVG images | 109 |
| About the View Source XML Window | 109 |
| Widget layout | 110 |
| Types of panes | 111 |
| About ExpandablePane | 112 |
| Widget properties..... | 112 |
| About the kitPx palette | 113 |
| About the kitPxHvac palette | 113 |
| About the kitPxGraphics palette | 114 |
| About the kitPxN4svg palette..... | 115 |
| About third-party graphics collections..... | 116 |
| About developing for portability..... | 116 |
| About the PxInclude Widget | 120 |
| PxInclude Widget concepts | 121 |
| Types of PxInclude Widget properties | 122 |
| Use PxInclude Widgets | 123 |

- About ORD Variables..... 124
- About Nav file elements 125
 - About Nav File Editor source objects 126
 - About Nav File Editor result tree..... 126
 - About Nav File Editor buttons 127
- About the ReportPxFile 127
- About the Grid Table view 128
- About the Grid Label Pane view 129
- About the Grid Editor view 131
 - Using the Grid Editor view 133
 - About editing rows/columns in a Component Grid 133
 - About the Report Edit Column/Edit Row dialog boxes..... 134
- About Mobile Px App 135
- Chapter 10 Components, views and windows 137**
 - Components in bajai module..... 137
 - bajai-BoundTable 137
 - Button 140
 - RadioButton..... 141
 - Label 143
 - HyperlinkLabel 145
 - Picture 147
 - Separator..... 147
 - PxInclude 148
 - Ellipse..... 148
 - Line 149
 - Path..... 150
 - Polygon 152
 - Rect..... 153
 - BorderPane..... 154
 - CanvasPane 155
 - ConstrainedPane..... 157
 - EdgePane 158
 - ExpandablePane 158
 - bajai-FlowPane 159
 - ResponsivePane 160
 - GridPane 162
 - ScrollPane..... 164
 - SplitPane 165
 - TabbedPane 167
 - TextEditorOptions 167
 - ValueBinding..... 169
 - Components in chart module..... 169
 - BarChart 169
 - ChartCanvas 171
 - ChartHeader 172
 - ChartPane..... 173

| | |
|--|-----|
| DefaultChartLegend..... | 175 |
| LineChart..... | 175 |
| Components in gx module..... | 176 |
| Brush..... | 176 |
| Components in kitPx module..... | 176 |
| BoundLabel, BooleanImage, NumericImage, or EnumImage..... | 176 |
| ActionButton..... | 179 |
| HyperLink Button..... | 181 |
| IncrementSetPointButton..... | 183 |
| DecrementSetPointButton..... | 183 |
| SaveButton..... | 184 |
| RefreshButton..... | 185 |
| ExportButton..... | 187 |
| BackButton..... | 189 |
| ForwardButton..... | 191 |
| LogoffButton..... | 193 |
| RebootButton..... | 195 |
| ButtonGroup..... | 198 |
| LocalizableButton..... | 199 |
| LocalizableLabel..... | 201 |
| AnalogMeter..... | 203 |
| Bargraph..... | 206 |
| SetPointSlider..... | 207 |
| SetPointToggleButton..... | 209 |
| SetPointCheckBox..... | 211 |
| SetPointFieldEditor..... | 213 |
| GenericFieldEditor..... | 214 |
| ActionBinding..... | 215 |
| BoundLabelBinding..... | 215 |
| ButtonGroupBinding..... | 216 |
| IncrementSetPointBinding..... | 216 |
| SetPointBinding..... | 217 |
| Components in pxeditor module..... | 218 |
| NewPxViewDialog..... | 218 |
| ResponsiveMigration..... | 218 |
| Components in report module..... | 219 |
| BqlGrid..... | 219 |
| ComponentGrid..... | 220 |
| EmailRecipient..... | 221 |
| ExportSource..... | 221 |
| FileRecipient..... | 222 |
| ReportPane..... | 223 |
| ReportService..... | 223 |
| SectionHeader..... | 224 |
| Plugins in pxEditor module..... | 225 |

pxEditor 225

Plugins in report module..... 225

GridTable..... 225

GridLabelPane 225

ComponentGridEditor..... 225

Index.....227

About this guide

This topic contains important information about the purpose, content, context, and intended audience for this document.

Product Documentation

This document is part of the Niagara technical documentation library. Released versions of Niagara software include a complete collection of technical information that is provided in both online help and PDF format. The information in this document is written primarily for Systems Integrators. To make the most of the information in this book, readers should have some training or previous experience with Niagara software, as well as experience working with JACE network controllers.

Document Content

This document provides information on the graphical presentation tools that are available in the Niagara 4 Workbench.

Included in the guide are basic procedures for creating and configuring graphics in the Workbench Px Editor, as well as a Px graphics reference which describes presentation concepts and architecture principles.

- To begin creating and configuring graphics in the Workbench Px Editor.

Document change log

Changes to this document are listed in this topic.

October 4, 2021

Added "Responsive Design Features" topic.

October 30, 2020

Added property table in component topics.

May 26, 2020

Added new properties of Action Bindings and Popup Bindings.

February 28, 2020

Added content on tag-based NEQL ORDS, including a procedure, "Converting bound ORDS to tag-based NEQL ORDS".

Also added a note on turning off HxPx views in the mobile profile.

August 7, 2019

Changes throughout for Niagara 4.8 release.

March 28, 2019

Updated "About Widget" topic for kitPxN4svg palette and added new topic "About the kitPxN4svg palette" .

August 21, 2018

Added new topic "ResponsivePane" in the components chapter and updated "FlowPane" topic for properties.

June 12, 2018

In the topic, "HTML5 Hx Profile", added a note regarding Chrome web browser refresh returns an error if station uses a self-signed certificate.

May 27, 2018

- Edited topics in Chapter 1. Edited “What’s new in graphics” topic, replaced content with information on enhancements in Niagara 4.6. In the section, “About profiles”, updated the “Default Mobile Web Profile” topic and added a new topic, “HTML5 Hx Profile”.
- In Chapter 2, added a procedure for “Optimizing existing Px files for mobile devices” and updated other procedures for Niagara 4.
- In Chapter 5, added topics, “Components in the pxEditor module”, and “ResponsiveMigration programObject”.

October 26, 2017

- Replaced document structure with standard entries and contents
- Added metadata tags for releases 4.3 and 4.4
- Edited “Preface” topics to use standard contents but they require additional editing for Niagara 4.

Related documentation

Additional information is available in the following documents.

- *Getting Started with Niagara*
- *Niagara Platform Guide*
- *Niagara Developer Guide*

Chapter 1 About Presentation XML (Px)

Topics covered in this chapter

- ◆ What's new in graphics
- ◆ About Px views
- ◆ About types of Px target media
- ◆ About the Px Viewer
- ◆ About Zoom capability in Px Editor
- ◆ About Px files
- ◆ Shared Px files
- ◆ About Widgets

This document describes the graphical tools, presentation concepts, architecture and basic procedures for creating and configuring graphics in the Workbench **Px Editor**.

The Px graphics environment provides the tools for you to build new applications, and you do not need to be a Java developer to create them.

To visually display the control logic that you develop in Workbench (for operations or for engineering purposes), you need to understand the basic principles, capabilities and limitations of Px target media. Obviously, graphics and text look different and have limitations when displayed on a mobile device rather than in a PC web browser. It is important to develop Px files with the primary target media types in mind and to test your Px views in the target media as you develop them.

Usually, it is easier to complete the engineering of your control logic using the **Wire Sheet** and other views before beginning to design the presentation of that logic. There are features and tools in the **Px Editor**, such as the **Make Widget** wizard, that allow you to drag components from the Nav tree to various panes in the **Px Editor**. If you have already built the logic, it should be visible in the Nav tree and available for drag-and-drop.

The general process of creating presentation views for control logic can follow many different paths. The major steps generally go as follows:

1. Create your view

When you create a view, you are creating a relationship between a Px file and a component. The Px file defines the view to associate with one or more components of various types, such as folders and points.

2. Add widgets

After creating a view, you add graphic visualizations (called widgets) to the canvas.

3. Bind your data to the widgets

To pass data to the widgets, you use data binding. The bound data from the control objects animates and updates the widgets.

4. Create a nav file

To easily find and navigate among views, you can create a customizable navigation tree using a special file type: .nav file. You edit the .nav file using the **Nav File Editor** and assign a particular nav file to a user in the user's profile (using the **User Manager** view).

5. Create and distribute a report

The reporting function helps you design, display, and deliver data to online views, printed pages, and to distribute via email.

What's new in graphics

If you are familiar with the graphics features of the prior releases of Workbench, the following summary of changes may be helpful in using the Niagara 4 version.

Enhancements

The following enhancements are implemented in Niagara 4.6 and later.

- Mobile-friendly Px improvements, which adapt the display of Px graphics on various devices (PC, cell-phone, tablet, etc.):
 - Widgets display in response to their container size, which is a function of the Responsive CanvasPane, FlowPane, and ResponsivePane components, giving you more control over how your graphics appear on a mobile device.
 - The **ResponsiveMigration programObject**, available in the **pxEditor** palette under the **Tools** folder, processes a directory full of Px files, applying responsive rules to them to optimize their display when viewed on mobile devices using HxPx.
- HxPx enhancements provide consistency of views and graphics across PC and mobile platforms. With them you to create one graphic for multiple environments.
 - Flexible graphics and legacy Mobile Px views look and function the same in an **HxPx View** as they do in Workbench. This includes the **Alarm Console**, **Scheduler** view, **Manager** views and **Property Sheet** field editors.

Deprecated items

Niagara 4.x no longer supports fox, http, and platform tunneling. This means that any hyperlinks (that is, ORDs in Px graphics) that are based on tunneling do not work.

In Niagara 4.6 and later, legacy Niagara Mobile technology is deprecated. Although it continues to be supported and works as expected, you should not use it for any new projects. Instead, use the mobile-friendly features of the HTML5 Hx Profile which provides a significantly improved mobile user experience.

About Px views

A Px view is a custom graphical view of control logic that you define in a Px file. The view provides a visualization of information in a dynamic graphical format. The purpose of a Px view is to provide a visual representation of information in a rich, dynamic format that is easy to create and to edit.

You use the Px Editor to build Px files with the desired visualization of your control logic without requiring software programming skills. When attached to a component, a Px view becomes the default view for the component.

Figure 1 Px view in a component property sheet



Looking at an object's Px view in the Property Sheet (where it may be edited) helps illustrate what a Px view is. A Px view is comprised of the following parts

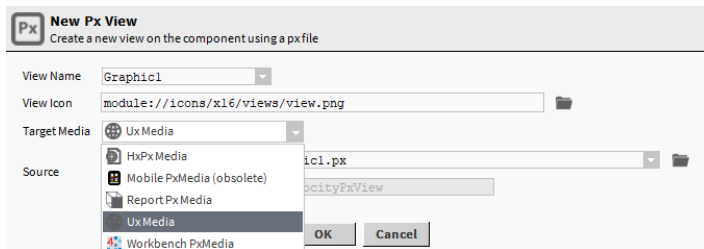
- The Px view icon appears to the left of the Px view display name, in the view selector menu, in the Nav tree side bar and in the Property Sheet view.
- The Px file contains the view's properties.

- Media identifies the type of view.
- Required permissions determine who can open the view.

About types of Px target media

PxEditor supports several client-side target media technologies. Consider the basic capabilities and limitations of these target media when you create a new graphic view.

Figure 2 New Px View



NOTE: Media values affect the initial contents of a Px file when you create it. Changing a view's Media type (on a component **Property Sheet**) after the Px file is created does not actually change the Px file.

The client-side target media technologies are:

- HxPx Media

This media suite offers a set of server-side servlets and a client-side JavaScript library. With it you can create a real-time interface without the use of the Java plug-in. It requires only web standards: HTML, CSS, and JavaScript.

If you are developing for a target media that is limited to browsers with HTML and CSS with no applet plugins available, be sure to test your Px views often in the appropriate browser as you proceed in your development. The goal of HxPx Media is to satisfactorily present your Px views in non-applet browsers, however, due to the difference in display technologies, you should verify that the Px view displays in the browser as you expect it to.

- Mobile PxMedia (obsolete)

Mobile Px pages are designed to enhance and enable widgets for presentation on a mobile device. You must select this option for your Px page if you expect to use the page on a mobile browser.

NOTE: In Niagara 4.6 and later, legacy Niagara Mobile technology is deprecated. Although it continues to be supported and works as expected, you should not use it for any new projects.

- Report Px Media

Report Px pages are simply Px pages that have a Report Pane at the root level, for convenience. Report panes include a Page number Timestamp property in a single column table.

- Ux Media

Niagara 4.10 and later provides the Ux Media type for improved graphics performance on controllers. Use the Ux Media type to achieve faster graphics load time and enable instantaneous feedback on changing values. A built-in migration tool helps you convert your existing graphics to a contemporary UI based on browser standards.

- Workbench PxMedia

The Workbench PxMedia allows the standard Workbench software to run inside a web browser using the Java Plugin. Workbench uses a small applet to download modules as needed to the client machine and to host the Workbench shell. These modules are cached locally on the browser's hard drive.

Responsive design features

A responsively designed graphic adjusts automatically to the device that displays it. The framework supports mobile-friendly Px graphics that display on various devices: PC, cellphone, tablet and so on.

- The display of a widget depends on the container size. The **ResponsiveCanvasPane**, **FlowPane**, and **ResponsivePane** components determine this size, which gives you control over how your graphics appear on a mobile device.
- The ResponsiveMigration programObject, available in the **pxEditor** palette under the **Tools** folder, processes a directory full of Px files, applying responsive rules to them to optimize their display when viewed on mobile devices using HxPx.

HxPx provides consistency of views and graphics across PC and mobile platforms. With them you can create one graphic for use in multiple environments.

Flexible graphics and legacy Mobile Px views look and function the same in an HxPx View as they do in Workbench. This includes the Alarm Console, Scheduler view, Manager views and Property Sheet field editors.

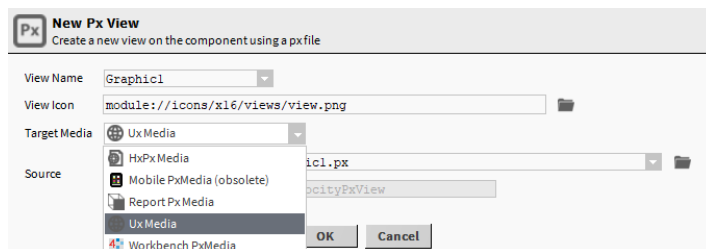
Deprecated items

- Fox, http, and platform tunneling. This means that any hyperlink (that is, ORDs in Px graphics) that is based on tunneling does not work.
- Legacy Niagara Mobile technology is deprecated. Although it continues to be supported and works as expected, you should not use it for any new projects. Instead, use the mobile-friendly features of the HTML5 Hx Profile, which provides a significantly improved mobile user experience.

What's New in Graphics

When creating a new Px page, in the **New Px View** window, you now have the option to select **UxMedia** under **Target Media**.

Figure 3 New PX View



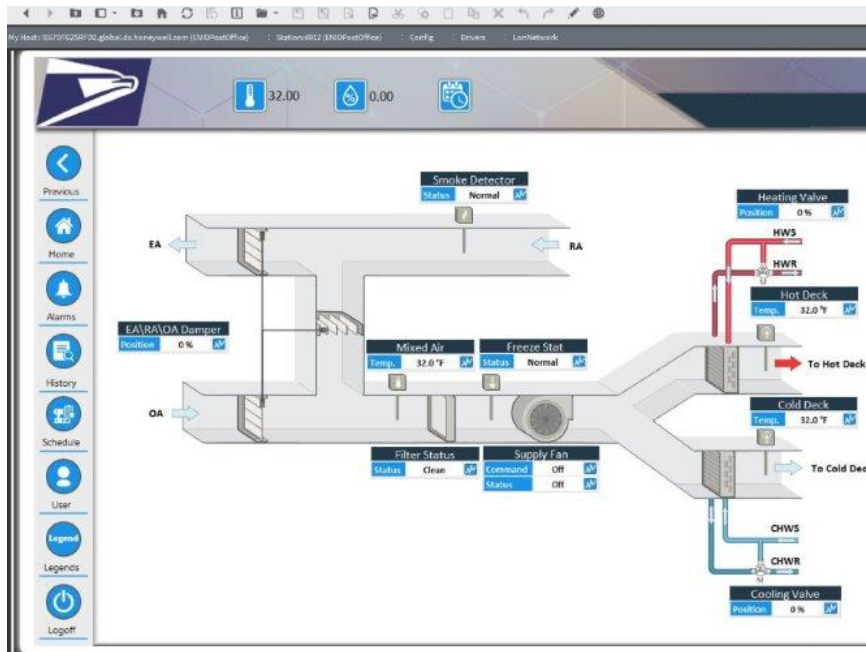
Niagara 4.10 and later versions provide **UxMedia**, which improves graphics performance on controllers. Using **UxMedia** achieves faster graphics load times and enables instantaneous feedback for changing values. A built-in migration tool helps you convert your existing graphics to a contemporary UI based on browser standards.

About the Px Viewer

The **Px Viewer** presents Px in a non-editable display. The **Px Viewer** allows the user (with required permissions) to view information and invoke actions on controls in the display.

Any time you select a Px view from the right-click menu or from the **View Selector** menu, the Px viewer displays the active component's view, as shown below. Use the Px editor to edit the Px file (as described in About Px Editor).

Figure 4 Px Viewer mode



About Zoom capability in Px Editor

In **Px Editor Edit** mode, you can zoom in and out on Px pages.

The Zoom feature uses three icons to change the magnification of the Px page. The current zoom level is visible in the status bar at the lower right corner of the window. The zoom level can range from x0.2 (maximum zoom out) to x10.0 (maximum zoom in).

Figure 5 Zoom icons



While editing a Px page, you see three zoom icons in the upper-right area of the toolbar:

- Zoom In (🔍), increases magnification for a close-up view that is helpful to achieve finer precision when positioning widgets.
- Zoom Out (🔍) reduces magnification of the page allowing you to get a better view of the whole page layout.
- Reset Zoom (🔍) returns the zoom level back to x1.0 (100%).


When navigating back and forth between Px pages, the software remembers the zoom level from page to page.

Zooming in on a Px Page

Zoom-in increases the magnification level of your Px page.

Step 1 In **Px Editor**, open an existing Px View containing at least one image.

Step 2 Click the Toggle View/Edit Mode button (🔧) to switch to Px Edit mode.

Step 3 Confirm that the current zoom level, shown in the status bar in the lower right corner, is x1.0 (100%). If not, click the Reset Zoom button ().

Step 4 Click the Zoom In button () as needed, increasing the view x0.2 (20%) with each click.


The **Px Editor** zooms in on the canvas pane providing greater precision in positioning widgets.


NOTE: The maximum magnification level is x10.0 (1000%). The system remembers the zoom level in a Px page when you navigate from page to page.


Zooming out on a Px Page

Zoom-out decreases the magnification level of your Px page.

Step 1 In the **Px Editor**, open an existing Px view containing at least one image.

Step 2 To switch to Px Edit mode, click the Toggle View/Edit Mode ().

Step 3 Confirm that the current zoom level, shown in the status bar in the lower right corner, is x1.0 (100%). If not, click the Reset Zoom button ().

Step 4 Click the Zoom Out button () as needed, decreasing magnification by x0.2 (20%) with each click.

The **Px Editor** zooms out on the canvas pane displaying more of the page at a reduced size. This is useful when working with large, complex layouts.

NOTE: The minimum magnification level is x0.2. The system remembers the zoom level in a Px page when you navigate from page to page.

Resetting zoom on Px Page

Reset Zoom returns the zoom level to x1.0 (100%).

Step 1 Click the Reset Zoom button ().

The canvas pane magnification resets to x1.0 (100%), displaying the Px page in actual size.

About Px files

A Px file defines the content and presentation of a Px view.

The Px file is a special XML file that describes the components in a database and can be any collection of components, up to a complete database. All views (wire sheet view, property sheet view, category sheet view, and so on) can be used on components in a Px file. This means that a Px view can provide a complete variety of options in the development of dynamic user interfaces.

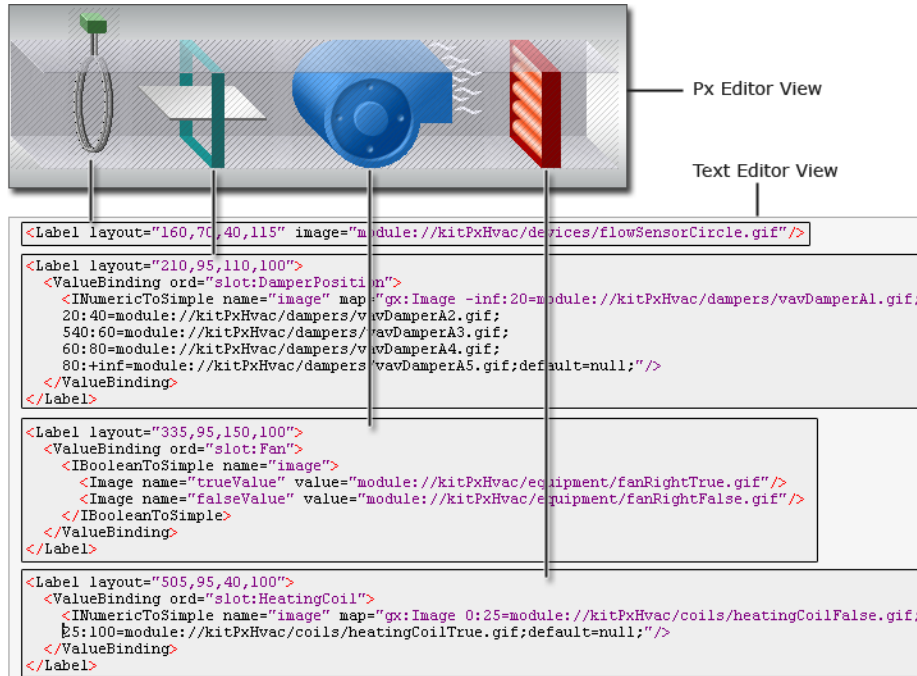
Figure 6 Basic default Px file in Text Editor View

```
<?xml version="1.0" encoding="UTF-8"?>
<px version='1.0' media='html:HtmlPxMedia'>
  <import>
    <module name='bajaul' />
  </import>
  <content>
    <ScrollPane>
      <CanvasPane name="content" viewSize="500,400"/>
    </ScrollPane>
  </content>
</px>
```

This is how a very basic Px file appears in the Text Editor. This file contains an empty canvas pane nested in a scroll pane.

As you add more objects to the file, (using the Px Editor) the file looks more like the snippet shown here. The graphic components, value bindings, and their container elements and attributes are all referenced in the Px file.

Figure 7 Px file in Text Editor View and in Px Editor View



The elements in the Px file are also graphically represented in the Widget Tree side bar pane.

NOTE: Two types of Px files are available. The standard Px file (includes a scroll pane at the root with a canvas pane) or the ReportPxFile (contains a report pane at the root).

Shared Px files

A Px file may be used as part of one or more Px views. Editing a shared Px file affects all views that use it.

Since the bindings within a Px file are always resolved relative to the current ORD, you can reuse the same Px file across multiple components by specifying bindings with relative ORDs. This allows you to create a single presentation and use it in views that are assigned to components that can use an identical graphic layout. The obvious advantage of file sharing is that you only need to create and edit one Px file for many views, thus saving time and ensuring consistency among similar applications. However, it requires that you understand the following caution.

CAUTION: Editing a Px file affects all views that use that particular Px file. When you view a component in the Px Editor, you have its Px file open for editing. If the Px file is shared with other Px views, all of those Px Views are changed.

About Widgets

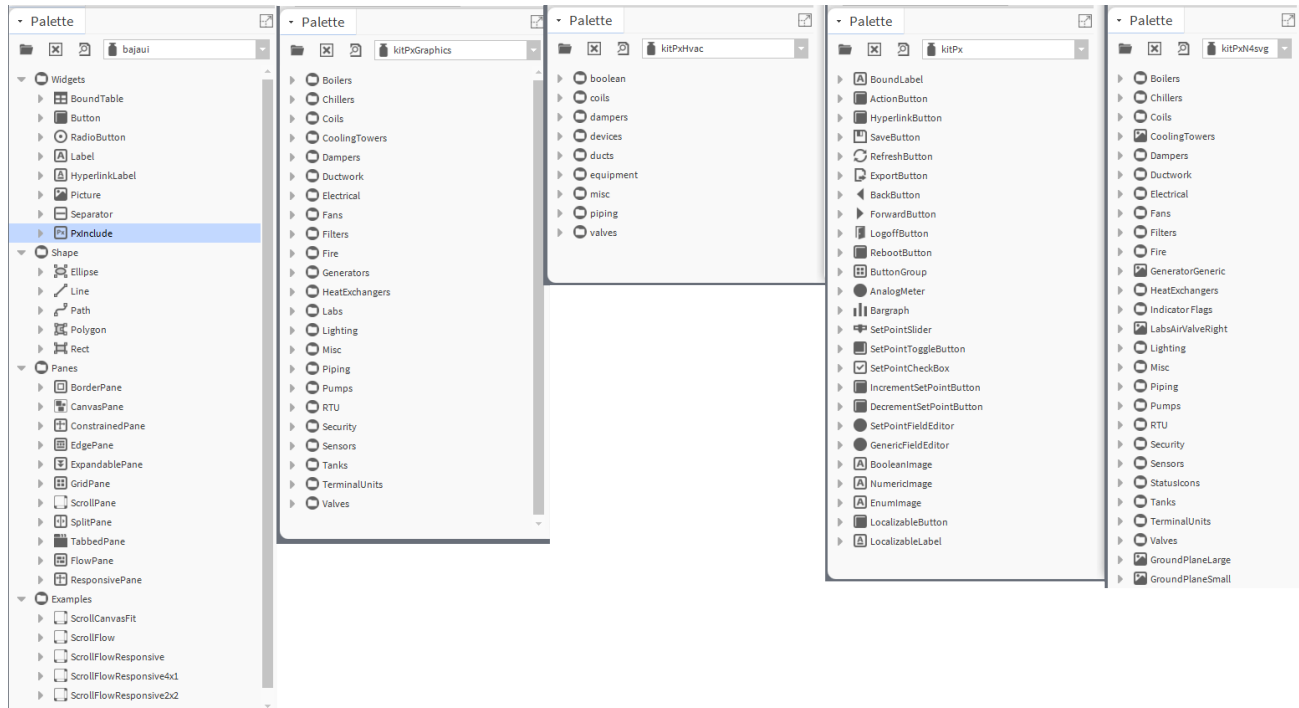
Widgets are components that provide visualization. Using **Px Editor**, you work with widget properties to define user interface functions for control and information display.

User widgets can process input in the form of mouse, keyboard, and focus events. User events are grouped into these same event categories. The **bajoui**, **kitPxGraphics**, **kitPxHvac**, **kitPx** and **kitPxN4svg** modules all contain widgets that you can use for building rich user interfaces. You find these widgets by opening them under the **Palette** side bar or by using the **Make Widget** wizard.

A number of widgets have been added to the `kitPxGraphics` module:

- high-detail images of several different actuators in both large and small sizes
- static images of a controller in three different sizes
- various animated widgets

Figure 8 Widgets in the bajauri, kitPxGraphics, kitPxHvac, kitPx and kitPxN4svg palettes



The widget tree side bar provides a good illustration of the hierarchy structure of all widgets in a Px file (refer to the *Getting Started with Niagara* for a summary).

Some complicated widgets have mini-frameworks all to their own, such as the table widget, tree widget, treeTable widget, and textEditor widget. Refer to the *Niagara Developer Guide* for more details about these types of widgets and for developer-level information about widgets.

The following sections discuss some of the primary characteristics of widgets.

Chapter 2 Creating a Px view canvas

Topics covered in this chapter

- ◆ Px views as slot properties
- ◆ About Px Layers

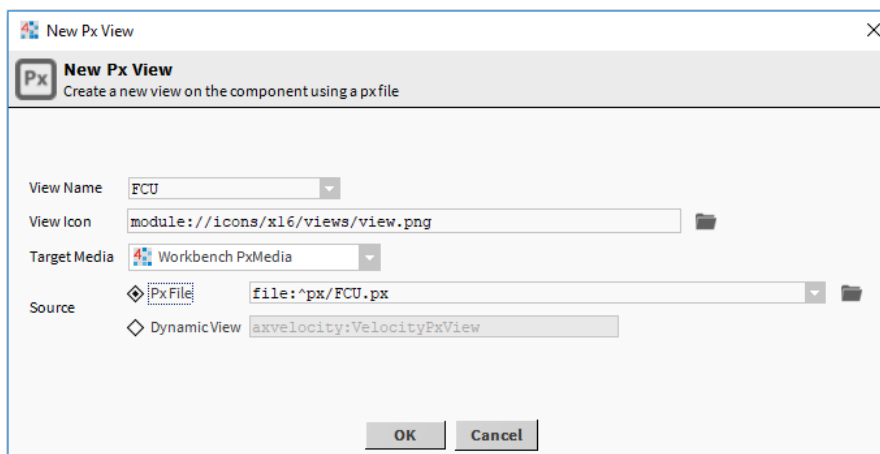
You may create and delete Px views using the **Slot Sheet** view. However, the easiest way to create a new canvas for a Px view on a component is to use the **New Px View** wizard.

When you create a view, you are creating a relationship between a Px file and a component. The Px file defines the view, which may be associated with one or more components of various types, such as folders and points.

You can assign an existing Px file to the view instead of creating a new one.

Step 1 In the Nav tree, right-click a component and click **Views**→**New View**.

The **New Px View** wizard opens.



Step 2 Enter a **View Name**.

Unless you specify a different name, the wizard assigns a default Px File name.

Step 3 Carefully select the **Target Media**.

Choosing a type of media has specific benefits when using the Px Editor to create a new view on a component for the first time.

In Niagara 4.6 and later, choosing **HxPx Media** takes full advantage of the HTML5 functionality, which provides a significantly improved mobile user experience.

NOTE: Media values affect the initial contents of a Px file when you create it. Changing a view's Media type (on a component **Property Sheet**) after the Px file is created does not actually change the Px file.

Step 4 To finish, click the **OK** button.

The wizard creates a new Px file in the station's **Files** folder and names it based on the **View Name** property. The wizard then opens the canvas for the new Px view in the **Px Editor**.

Px views as slot properties

The Px view is attached to a component as a dynamic slot property of the type `baja:PxView`. You can add new Px views directly in the slot sheet view.

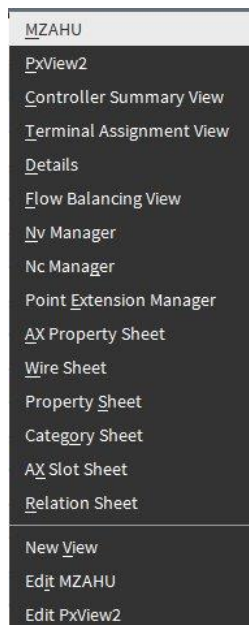
The following graphic shows three Px views in a single component slot sheet.

Figure 9 Three Px views as slot properties

| Slot Sheet | | | | | | | |
|-----------------------|--------------|----------|--------------|------------|-------|------------------|--------|
| Slot | # | Name | Display Name | Definition | Flags | Type | Facets |
| <input type="radio"/> | Property 241 | lonLink | lonLink | Dynamic | | lonworks:LonLink | |
| <input type="radio"/> | Property 242 | AlarmMap | AlarmMap | Dynamic | hN | baja:Component | |
| <input type="radio"/> | Property 243 | MZAHU | MZAHU | Dynamic | | baja:PxView | |
| <input type="radio"/> | Property 244 | PxView2 | PxView2 | Dynamic | | baja:PxView | |

The Px view highest on the slot sheet is the default view. You can reorder Px views to change the default Px view of a component. For an active component, all Px views (if any) appear in the view selector.

Figure 10 Three Px views in the view selector



About Px Layers

Px Layers are useful in terms of manipulating a number of objects in a Px Page.

Several handling options exist for Px Layers including: grouping objects in a Px page, assigning (or unassigning) objects to a layer, adding a layer, renaming a layer, and deleting a layer.

Grouping Px Layers

In a Px Editor view, you can use Px Layers to group one or more objects on the Px page for ease of manipulation.


When you assign several objects to a common layer, you can easily select all of the objects by selecting the layer. You can also lock and/or hide all of the objects in a layer.

Adding a Px Layer

You add a new Px Layer in **Px Editor** view.

Prerequisites: The **Px Layers** palette is open.

NOTE: Objects do not display a layer property until at least one layer exists.

Step 1 In **Px Editor**, on the **Px Layers** palette, click the add button ().

The **Add** window opens.

Step 2 Type a name for the new layer and click **OK**.

The new layer displays in the **Px Layers** palette and all objects that can be edited have a new layers property value option available.


Assigning/Unassigning objects to a layer

You add, change or remove an object's layer assignment in the **Px Editor** view.

Step 1 Right-click on the desired object on the **Px Editor** canvas or in the **Widget Tree** palette and select **Edit Properties** from the popup menu.

NOTE: Instead of using the popup menu, you can select an object and change the layers value using the **Properties** palette.

The **Properties** window opens.

Step 2 Click the value select button () of the layer property.

A list of options opens.

Step 3 Choose one of the following:

- To add or change the object's layer assignment, choose a named layer value option.
- To remove any object-layer assignment, choose the empty (top) option.

Step 4 To close the **Properties** window, click **OK**.

The **Px Editor** adds, changes or removes the object-layer assignment depending on your selection.

Renaming a Px Layer

This procedure describes how to change the name of a Px Layer.

NOTE: Renaming a layer does not affect the layer assignments or the objects assigned to that layer. For example, if Object A is assigned to Layer 1 when Layer 1 is renamed to Layer 2, Object A remains assigned to the layer (now named Layer 2).

Step 1 In the **Px Editor's Px Layers** palette, right-click on the layer to rename and select **Rename** from the popup menu.

Step 2 In the **Rename** window, type the new name and click **OK**.

The **Px Editor** changes the layer name.

Deleting a Px Layer

This procedure describes how to remove a Px Layer.

NOTE: Removing a layer deletes the layer from the **Px Layers** palette but does not delete any objects or affect any object properties other than the layer property for any assigned objects. The **Px Editor** removes the layer property value from any objects assigned to the deleted layer.

Step 1 In the **Px Editor's Px Layer** palette, right-click on the layer that you want to remove and select **Remove** command from the popup menu.

The layer is removed and no longer appears in the **Px Layer** palette.

Chapter 3 Adding widgets using the Make Widget wizard

Topics covered in this chapter

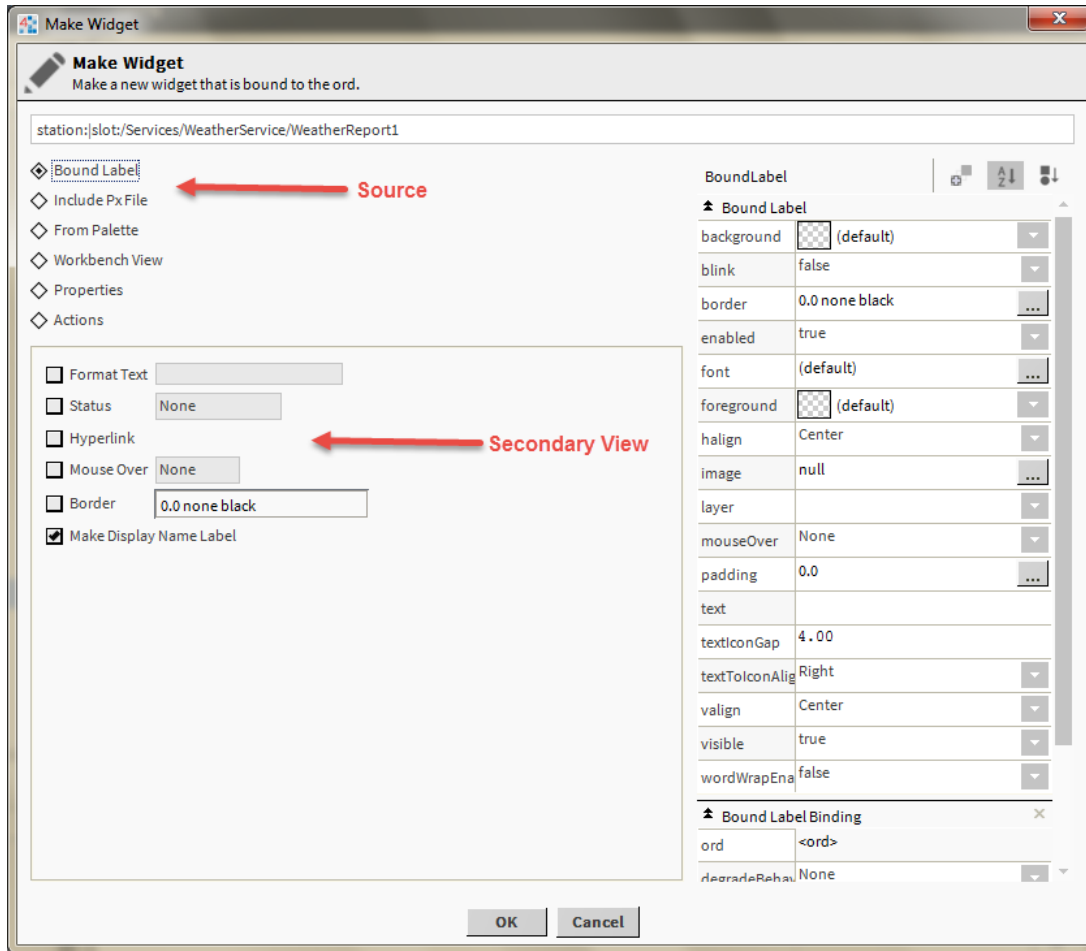
- ◆ Widget painting
- ◆ Widget commands
- ◆ Making Bound Label widgets
- ◆ Making Chart widgets
- ◆ Hyperlinks with Popup Bindings (an example)
- ◆ Creating a scalable image using the Picture widget
- ◆ Making component Properties-based widgets
- ◆ Make Palette kit-based widgets
- ◆ Making Time Plot widgets
- ◆ Making view-based widgets
- ◆ Making Action widgets
- ◆ About Widget sizing in Px Editor
- ◆ Embedding a Px View in another Px View

The **Make Widget** wizard provides properties for creating widget bindings.

You can add graphic visualizations to your Px page by dragging pre-made widgets from a palette or by making new widgets. When you make a widget using the **Make Widget** wizard, the window provides properties for creating the widget bindings.

Step 1 From the Nav side bar, drag a widget onto the **Px Editor** canvas.

The **Make Widget** wizard opens, with the ord for the selected component displaying in the ord area.



Step 2 To change the ord value, double-click it.

Step 3 From the Source options list, select the type of binding (choose from: Bound Label, Properties, Actions, and others).

NOTE: Binding options that are dimmed indicate invalid options for the selected component.


The Secondary view area displays properties and options that are related to the selected Source option.

Step 4 In the Secondary view area, choose a widget template, formatting options for display labels, or views, as appropriate for your Source option.

Step 5 In the right-hand list of properties, edit the properties or actions, as desired.

Step 6 To complete the configuration, click **OK**.

The wizard closes and your new widget displays on the **Px Editor** canvas.

Step 7 To toggle the view, click the view/edit mode tool bar icon ().

This icon alternates between displaying the widget in the **Px Viewer** and **Px Editor**.

Widget painting

Painting defines how widgets present themselves graphically (using: colors, transparencies, and so on), as well as clipping.

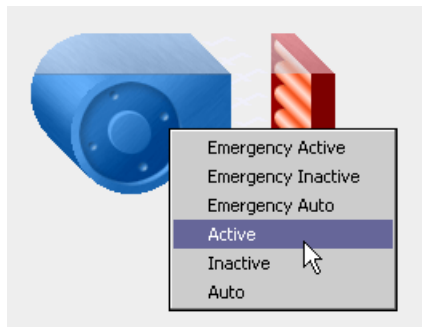
The **Px Editor** always positions the origin (0, 0) of a graphic at the top, left corner of the widget. It sets the graphic's clip (visible area) to the widget's size. Alpha and transparent pixels blend with the pixels already drawn. The **Px Editor** draws widgets in property order beginning with the widget at the bottom and ending with the widget at the top. Effective z-order is the reverse of property order.

Widget commands

Widgets can have user commands that appear as buttons and menu actions.

Toggle commands are commonly used with a check box, radio button, and other items. Menu commands are available and used on fans, coils, and other widgets.

Figure 11 Widget commands



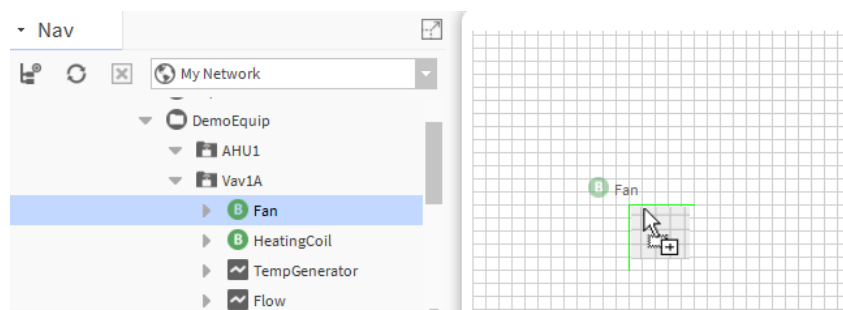
Making Bound Label widgets

This procedure creates a new Bound Label widget using the **Make Widget** wizard. When you make a widget, the wizard window provides properties for creating the widget bindings.

Prerequisites: A Px View canvas is open. No open palette is required.

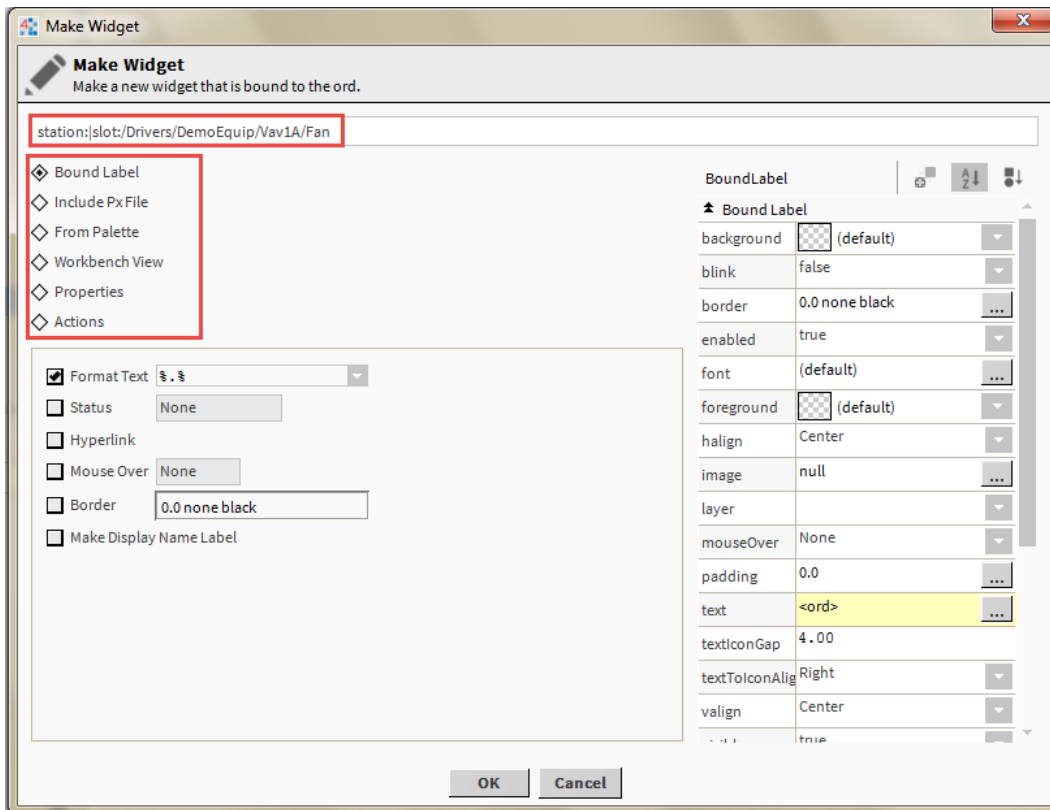
Step 1 In the Nav side bar, expand the tree to locate the component to add to the **Px Editor** canvas.

The canvas opens.



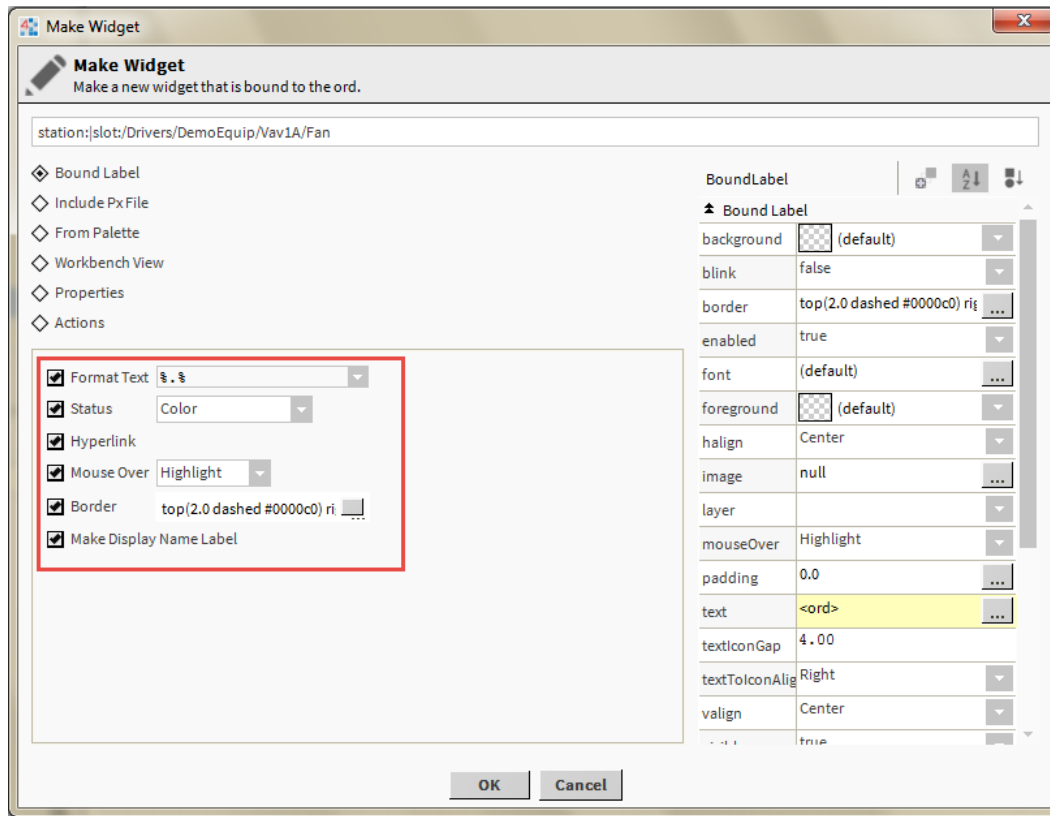
Step 2 Drag the component to the canvas.

The **Make Widget** wizard opens.



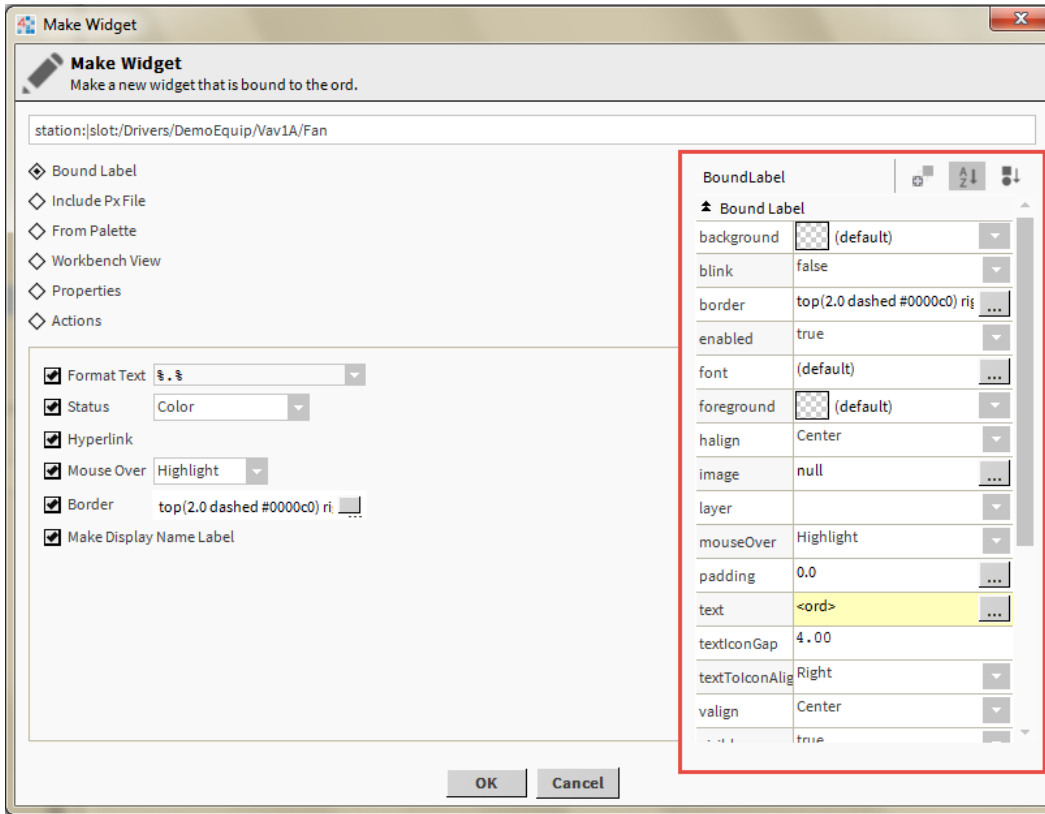
The ord for the selected component displays in the ord area (first red box).

- Step 3 For a different slot in the component (for example: out or value) or to bind to a different component, double click the ord and browse to the new location.
- Step 4 Select the **Bound Label** from the source options list (second red box).
The source properties display below the source options list.



Step 5 Configure the source properties (red box).

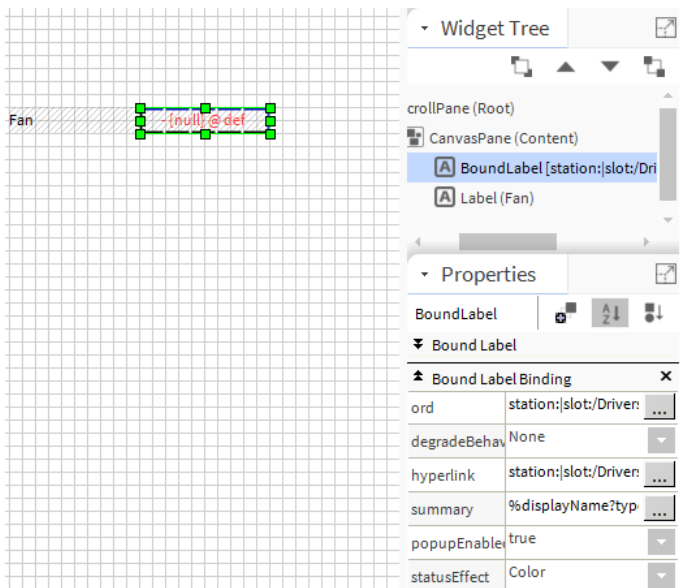
The **Make Display Name Label** property creates a separate unbound label that appears directly beside the bound label when the widget displays on the canvas.



Step 6 On the right side, edit the Bound Label properties.


Step 7 To override an option, change it in the properties area and click **OK**.

The **Make Widget** wizard closes and the new bound label opens on the **Px Editor** canvas.



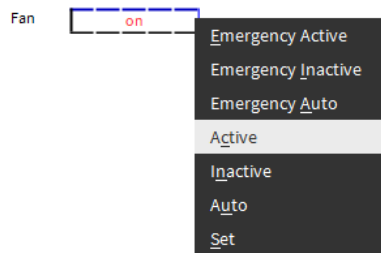
Step 8 Edit the display label properties using the Properties side bar on the right.

These properties are available because you selected **Make Display Name Label** in the source options section of the wizard.

Step 9 To toggle between the viewer and editor, click the view/edit mode icon () in the tool bar.

Step 10 Confirm that the configuration works.

For example, to activate the Fan above, right-click the graphic and click **Active**.

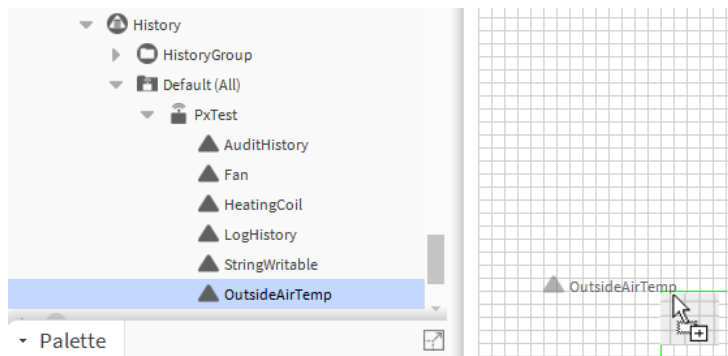


Making Chart widgets

This procedure describes how to create a chart widget using the **Make Widget** wizard.

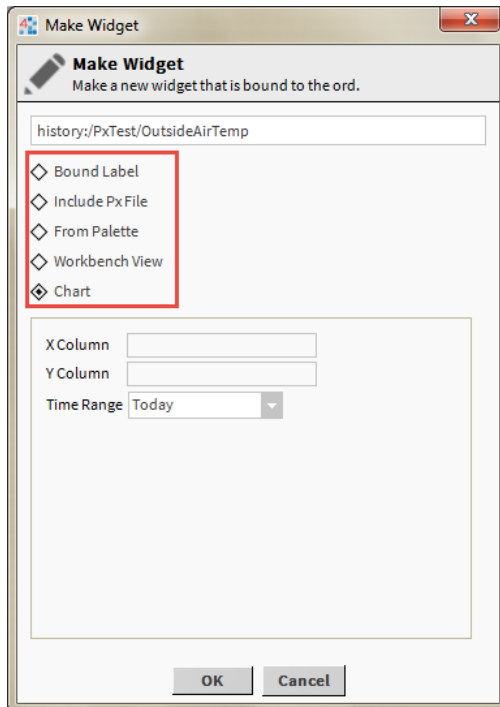
Prerequisites: A Px View canvas is open. No open palette is required.

Step 1 Open the Nav side bar and expand the tree to locate the component to add to the **Px Editor** canvas.



Step 2 Drag the component to the canvas.

The **Make Widget** wizard opens.

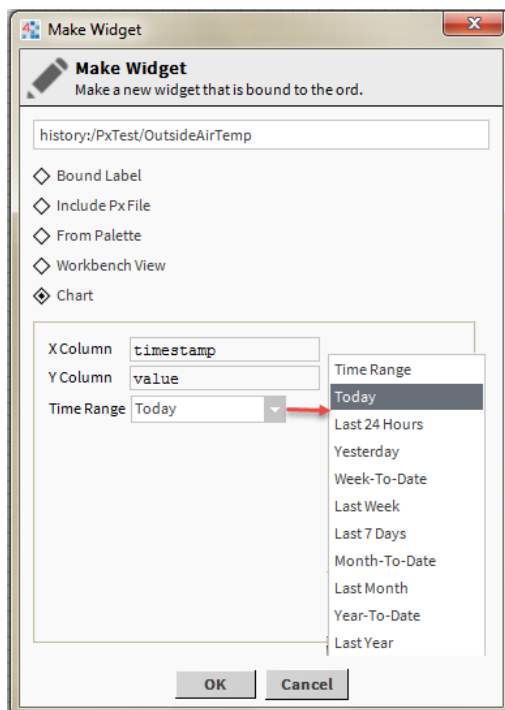


The ord for the selected component displays in the ord area.

Step 3 To browse for a different slot in the component (for example: out, value) or to bind to a different component, double click the ord and select a different ord.

Step 4 Select **Chart** in the source options list.

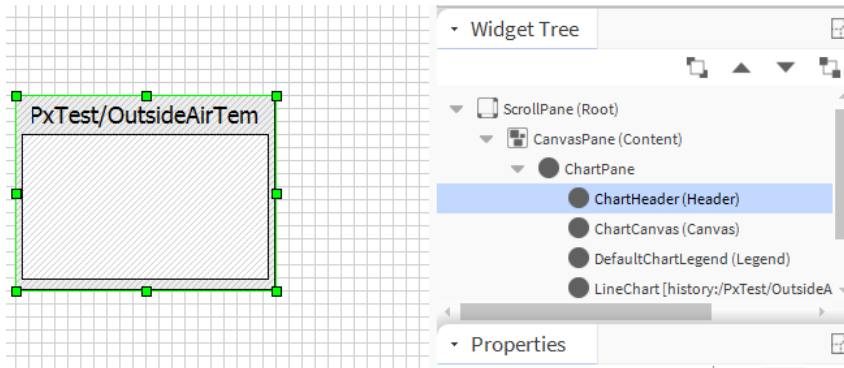
The source properties display.



Step 5 Configure the **X Column** and **Y Column** (x and y axes) of the chart and the **Time Range**.

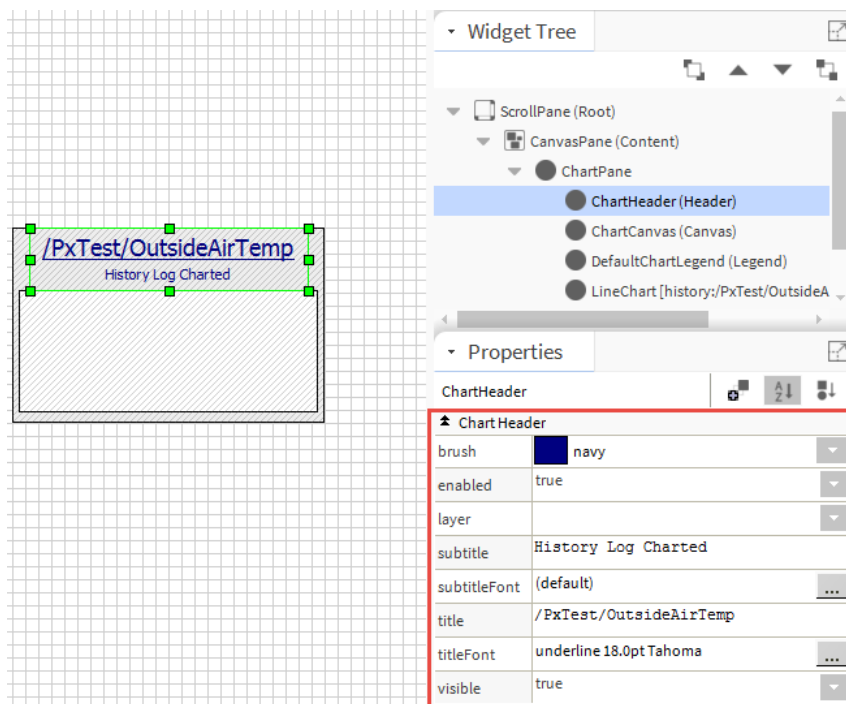
Step 6 To complete the wizard, click **OK**.


The **Make Widget** wizard closes and the new Chart widget opens on the **Px Editor** canvas.

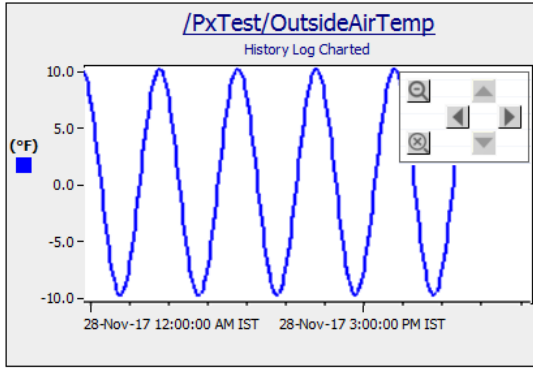


Step 7 Edit the widget by doing one or both of the following:

- To allow the plot to display completely, resize the widget.
- To edit the chart's widget properties (header, lines, fonts, and so on), use the **Widget Tree** and **Properties** tree on the right.



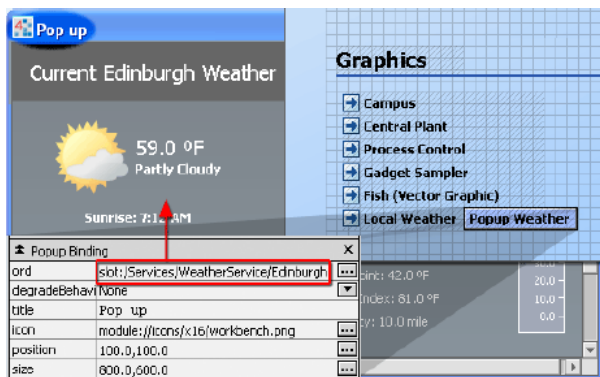
Step 8 To display the Chart in the **Px Viewer**, click Toggle View/Edit Mode () in the tool bar. The viewer opens.



Hyperlinks with Popup Bindings (an example)

You add a Popup Binding to an object that serves as a hyperlink. After adding the Popup Binding, you configure Px and **Popup** Window properties. This topic describes an added hyperlink using a Popup Binding on a **Px Button** widget.

Figure 12 Example of the options to configure a popup hyperlink



In the **Px Editor** view, a Popup Binding is added to a **Px** button widget (Popup Weather). This is the default view of the component.

The Popup Binding properties (displayed in the bottom left corner) specify the following:

| Property | Description |
|------------------------|--|
| ord | Binds the component to: <code>slot:/Services/WeatherService/Edinburgh</code> . |
| degradebehavior | No degrade behavior is assigned. |
| title | The title Pop up appears in the top left corner of the Popup Window. |
| icon | The <code>workbench.png</code> icon is assigned to the top left corner of the Popup Window. |
| position | Window X and Y coordinate screen position is <code>x=100</code> and <code>y=100</code> pixels. |
| size | The Popup Window size is 800 pixels wide and 600 pixels high. |

Creating a scalable image using the Picture widget

This procedure is only applicable for the external SVG image (which is not available in the `kitPxN4.svg` palette). You can easily create a scalable image using an SVG image in a **Picture** widget with the `scale` property set to `Fit`.

Prerequisites:

- A recent HTML5 compliant browser , for example: FireFox, Chrome, Opera, IE and Safari.
- The svgBatik module (svgBatik.jar) must be in the !modules folder.
- The SVG images must be created outside of Workbench, using vector graphic software, and copied to the station file system

SVG images are vector graphic images. You can certainly save BMP, GIF, PNG, etc. images as SVG images. However, if other images are not vector graphics to begin with, their quality degrades when scaled.

NOTE: Interactive features of SVG images, such as embedded Javascript-based animations, are not supported.

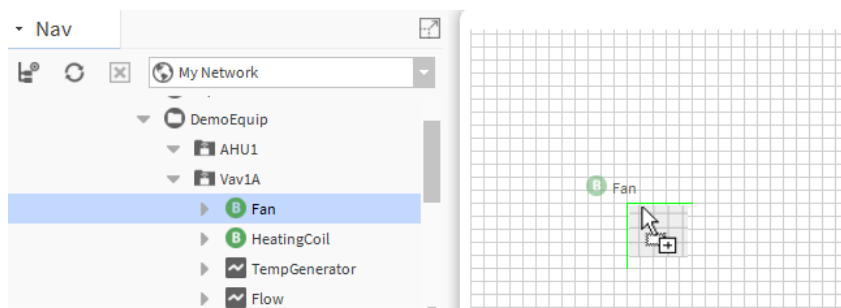
- Step 1 To create a new Px view, right-click a component in the Nav tree and click **Views→New View** .
- Step 2 In the **Palette** sidebar, open the **ba7au1** palette and expand the **Widgets** folder.
- Step 3 Drag a **Picture** widget to the **Px Editor** canvas.
- Step 4 Double-click the **Picture** widget to display its **Properties** pane (right side of the canvas).
- Step 5 Click the File Chooser icon (...) to the right of the **image** property, browse to select an SVG image and click **OK**.
- Step 6 Set the **scale** property to **Fit** and click **OK**.
- Step 7 In the Px view, click and drag a corner of the widget to resize it smaller and/or larger.
- The **Picture** widget scales perfectly without degrading the graphic quality.

Making component Properties-based widgets

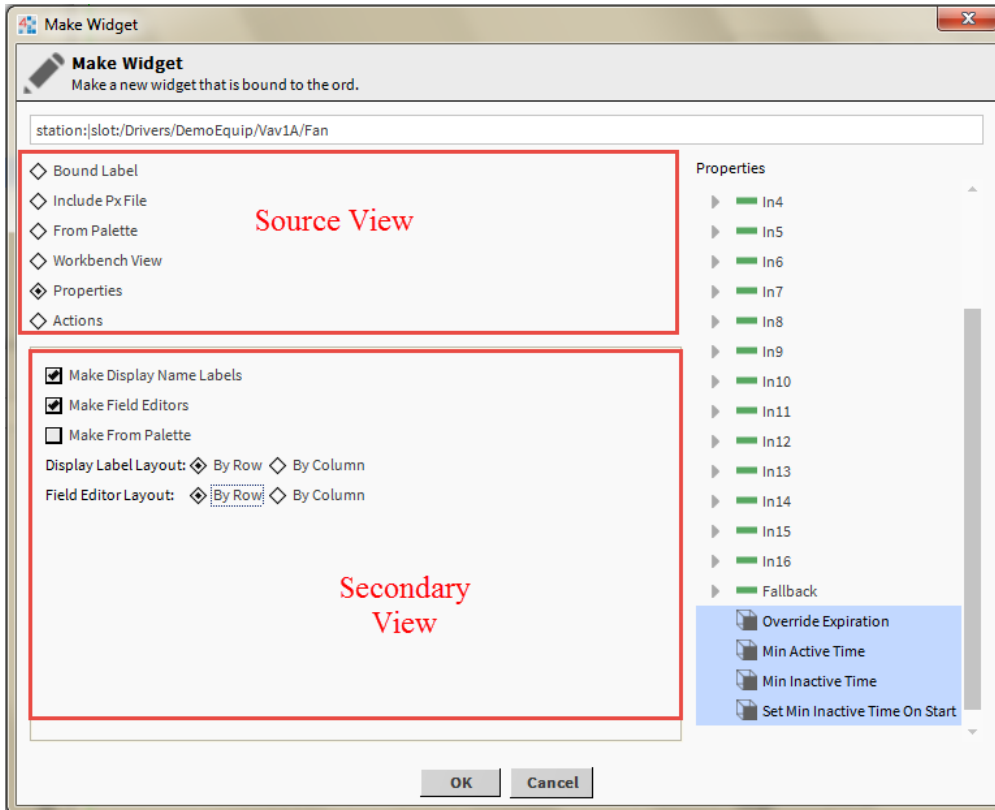
This procedure describes how to create a component Properties-based widget using the **Make Widget** wizard.

Prerequisites: A Px View canvas is open. No open palette is required.

- Step 1 In the Nav side bar, expand the tree to locate the component for which to create a Px View.



- Step 2 Drag the desired component to the canvas.
The **Make Widget** wizard opens.



The ord for the selected component displays in the ord area.

NOTE:

Step 3 To bind to a different slot or component, double click the ord and browse to the different slot in the component (for example: out or value) or to a different component.

Step 4 Select the `Properties` from the Source options list.

The Secondary View area displays options that are available for creating the new widget.

Step 5 In the Secondary view area, select one or more options from the list.

- **Make Display Name Labels**

This option displays a separate unbound label that appears directly beside the bound label when the widget displays on the canvas. Edit the display label properties using the properties side bar after completing the Make Widget wizard.

- **Make Field Editors**

This option displays an area for editing the property in the Px viewer (or in a browser).

- **Make From Palette**

This option opens the palette side bar view in the wizard with which to select a suitable widget template from any available palette.

- **Display Label and Field Editor Layout options (By Row or By Column)**

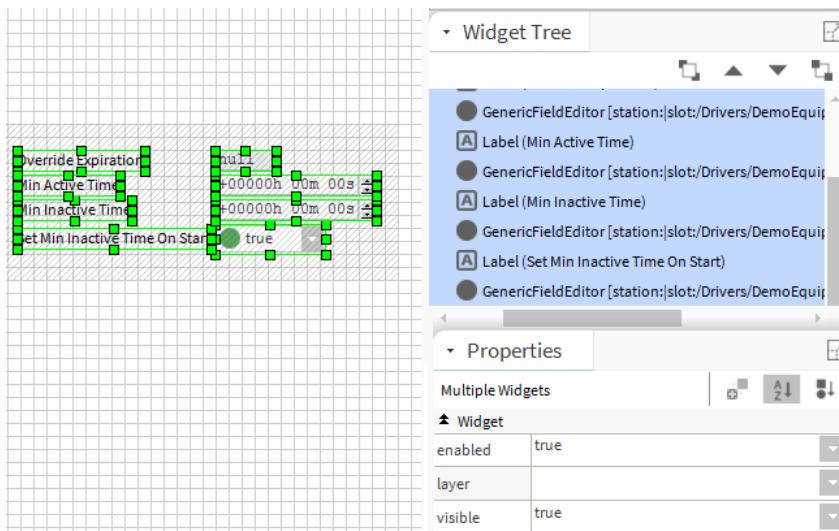
These options affect the arrangement of the properties on the canvas when you complete the wizard. Your choice of layout is optional and you may rearrange the layout after you place the items on the canvas.


Step 6 In the Properties sheet area, to select the properties to bind to the widget, hold the **Ctrl** key and select more than one property.

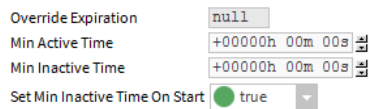


Step 7 To complete the wizard, click **OK**.

The **Make Widget** wizard closes and the new property widgets appear on the **Px Editor** canvas.



Step 8 To display the result in the **Px Viewer**, click Toggle View/Edit Mode () in the tool bar.

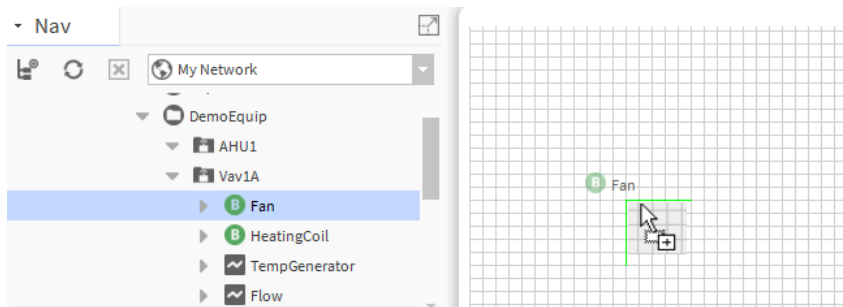


Make Palette kit-based widgets

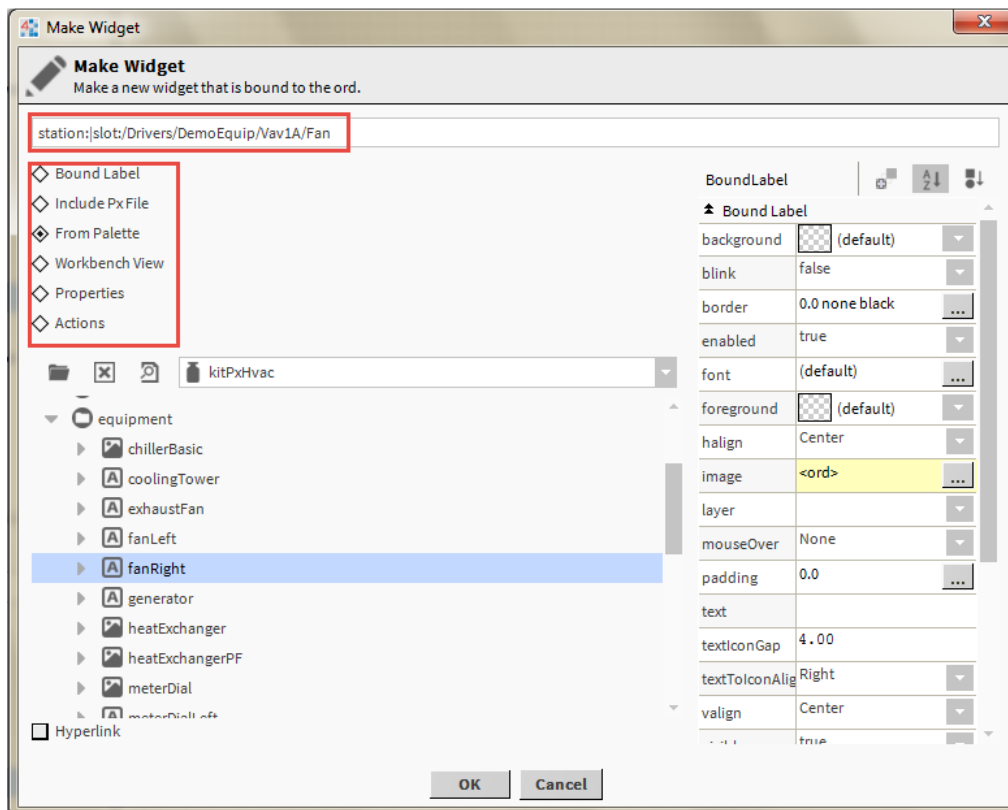
This procedure describes how to create a Palette kit-based widget using the **Make Widget** wizard.

Prerequisites: A Px View canvas is open. No open palette is required.

Step 1 In the Nav side bar and expand the tree to locate the component that you want to add to the Px Editor canvas.

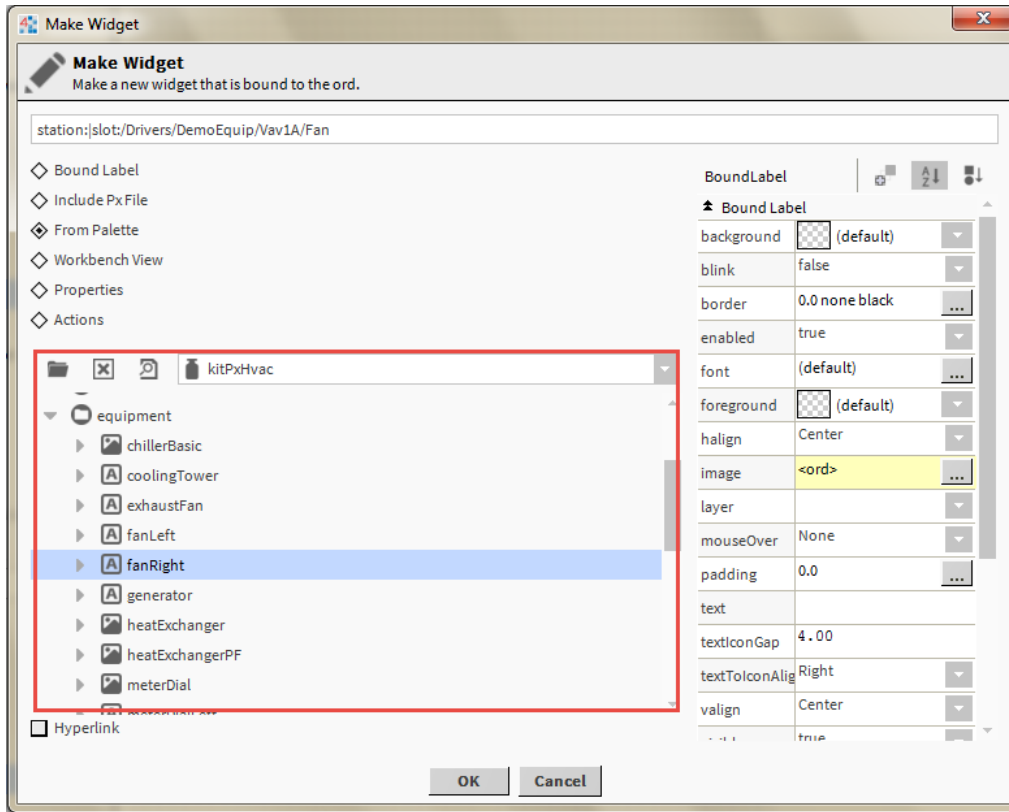


- Step 2 Drag the component onto the canvas.
The **Make Widget** wizard opens.



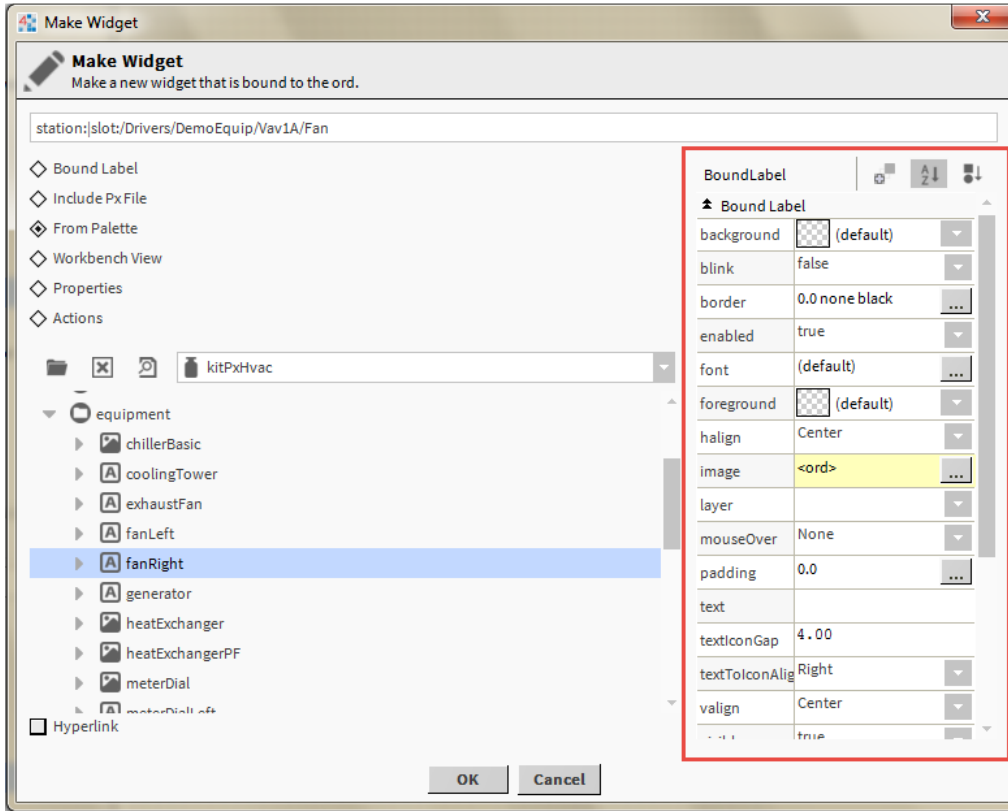
The ord for the selected component displays in the ord area.

- Step 3 To bind to a different slot or component, double click the ord and browse to the different slot in the component (for example: out or value) or to a different component.
- Step 4 Select the **From Palette** from the Source options list.
The Secondary view area opens the palette side bar view.

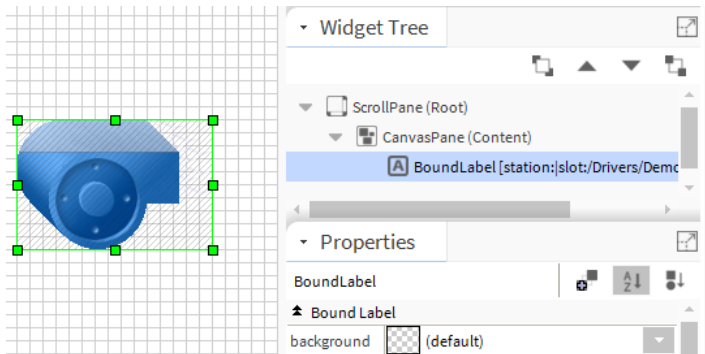



All palettes that are currently open in the palette side bar are available under the module option selector arrow of the Secondary view area (red box).

- Step 5 To open other palettes, click on the folder icon (📁).
- Step 6 Expand the module tree to find and select the desired widget template.



Step 7 In the Properties area (to the right) edit the widget template properties and click **OK**. The **Make Widget** wizard closes and the new widget opens on the **Px Editor** canvas.



Step 8 To display the result in the **Px Viewer**, click Toggle View/Edit Mode () in the tool bar.



Making Time Plot widgets

This procedure creates a Time Plot widget using the **Make Widget** wizard.

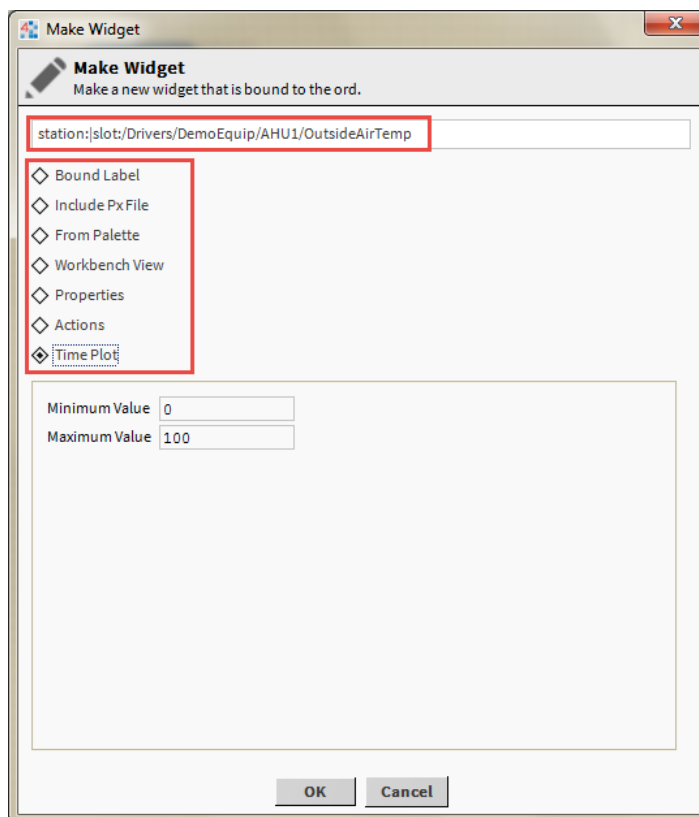
Prerequisites: A Px View canvas is open. No open palette is required.

Step 1 Open the Nav side bar and expand the tree to locate the component that you want to add to the Px Editor canvas.



Step 2 Drag the desired component to the canvas.

The **Make Widget** wizard opens.

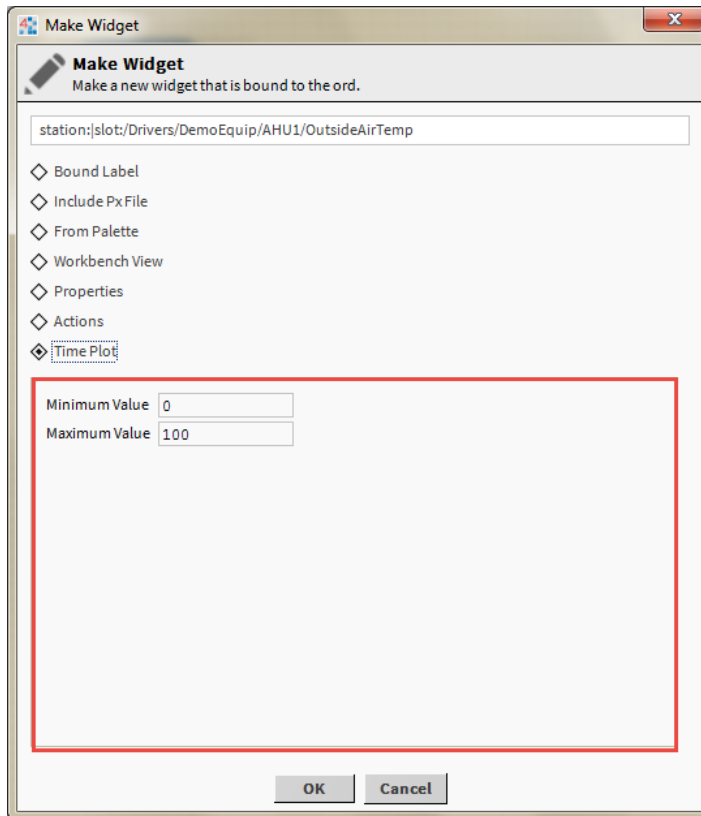


The ord for the selected component displays in the ord area.

Step 3 To bind to a different slot or component, double-click the ord and browse to the different slot in the component (for example: out or value) or to a different component.

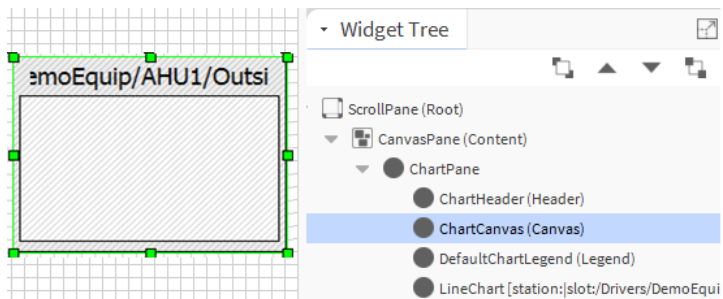
Step 4 Select the `Time Plot` from the Source options list.

The Secondary view area displays **Minimum Value** and **Maximum Value** properties.



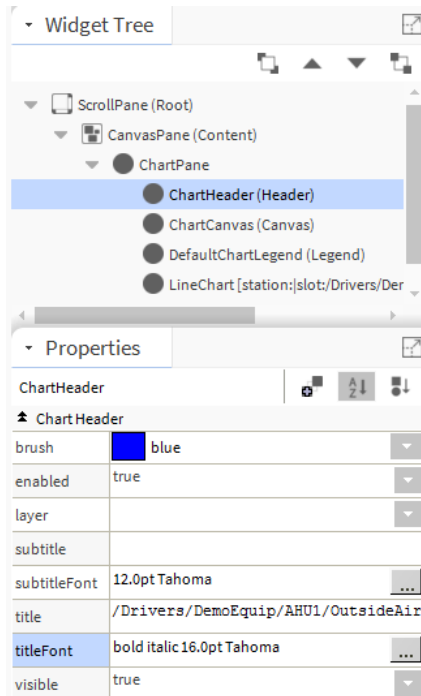
Step 5 In the Secondary view area (red box) enter the desired minimum and maximum values to display in the **Time Plot** widget and click **OK**.


The **Make Widget** wizard closes and the new Time Plot widget opens on the **Px Editor** canvas.

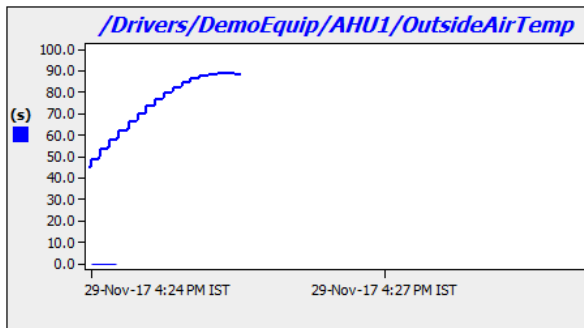


Step 6 Edit the widget, if desired, by doing one or both of the following:

- To allow the plot to display completely, resize the widget.
- Edit the Time Plot widget properties (header, lines, fonts, and so on) using the **Widget Tree** and **Properties** palettes.



Step 7 To display the result in the **Px Viewer**, click Toggle View/Edit Mode () in the tool bar.

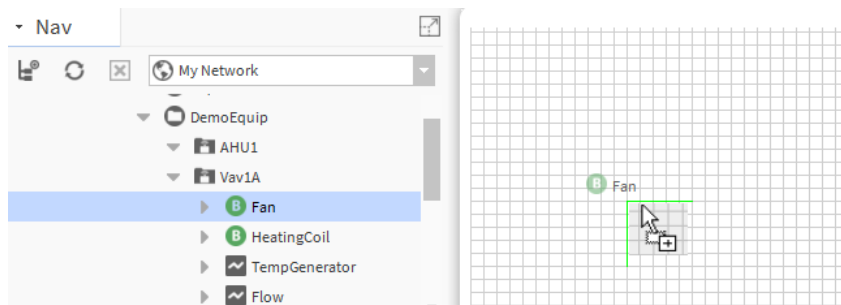


Making view-based widgets

This procedure creates a view-based widget using the **Make Widget** wizard.

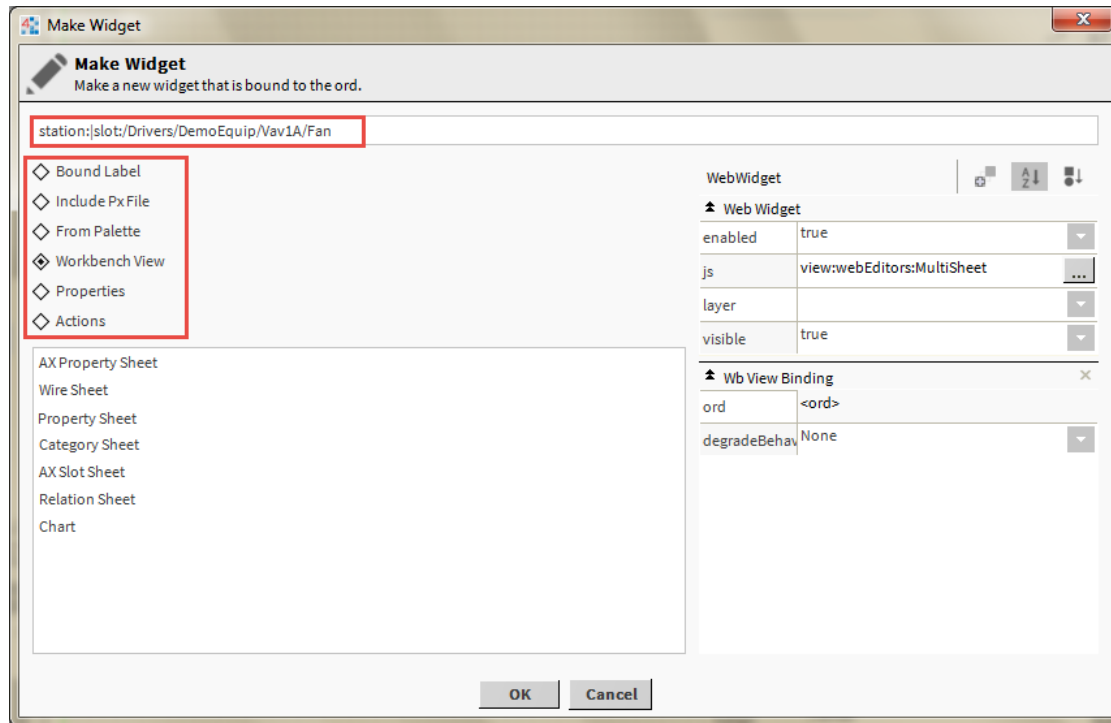
Prerequisites: A Px View canvas is open. No open palette is required.

Step 1 In the Nav Tree, expand the tree to locate the component to add to the **Px Editor** canvas.



Step 2 Drag the component to the canvas.

The **Make Widget** wizard opens.

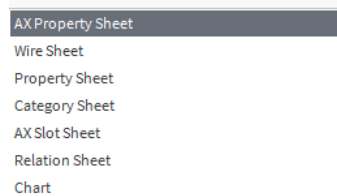


The ord for the selected component displays in the ord area.

Step 3 To bind to a different slot or component, double-click the ord and browse to the different slot in the component (for example: out or value) or to a different component.

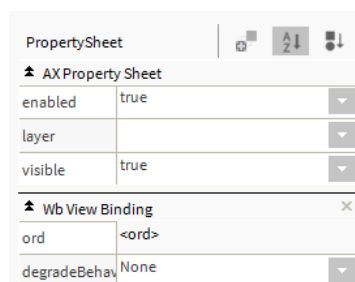
Step 4 Select *Workbench View* from the Source options list.

The Secondary view area displays a list of the types of view options (Property Sheet, Chart, etc.) that are available for the selected component.



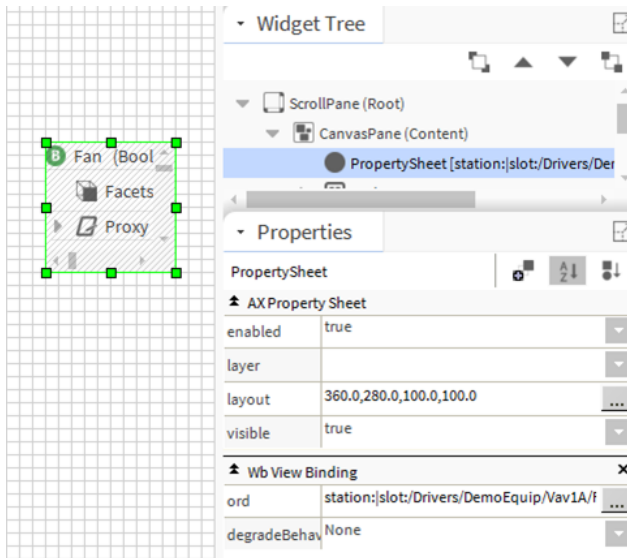
Step 5 In the secondary view area, select the desired component view to add.

The AX Property Sheet and Property Sheet views display the component properties.




Step 6 In the Properties area, edit the widget template properties, as desired and click **OK**.

The **Make Widget** wizard closes and the new widget view opens on the **Px Editor** canvas.

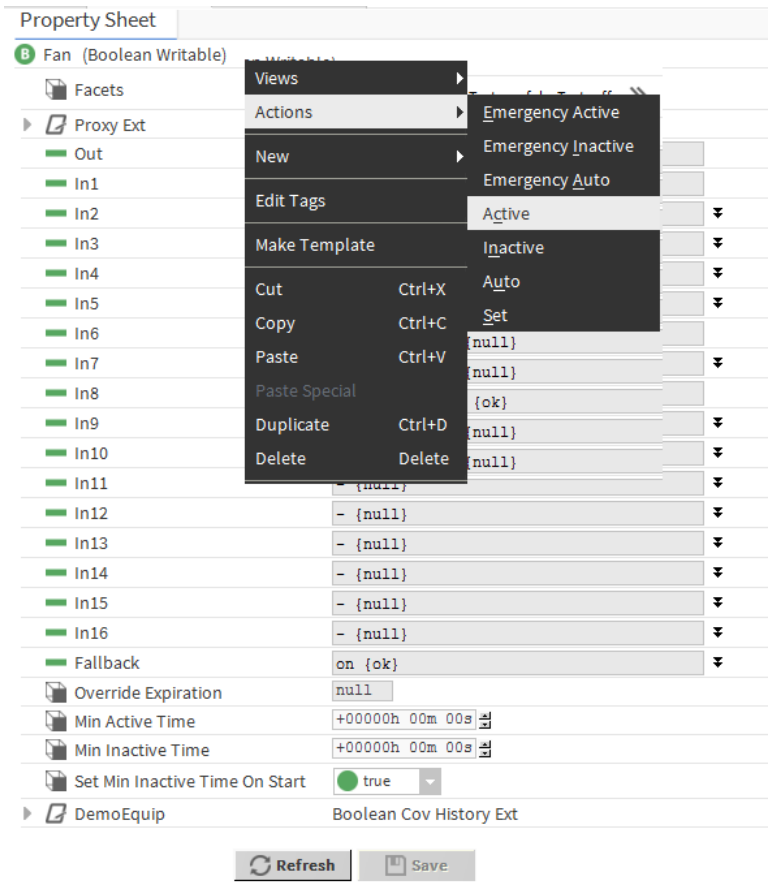


Step 7 Use the handles on the widget view to expand the view to the desired size or edit the **Layout** property in the properties side bar.

Scroll bars appear if the widget view is larger than the canvas view area.

Step 8 To display the result in the **Px Viewer**, click Toggle View/Edit Mode () in the tool bar.

The view is fully functional in the Px viewer, where you can edit properties, invoke commands, or make other edits to a component, just as if you are editing in Workbench.

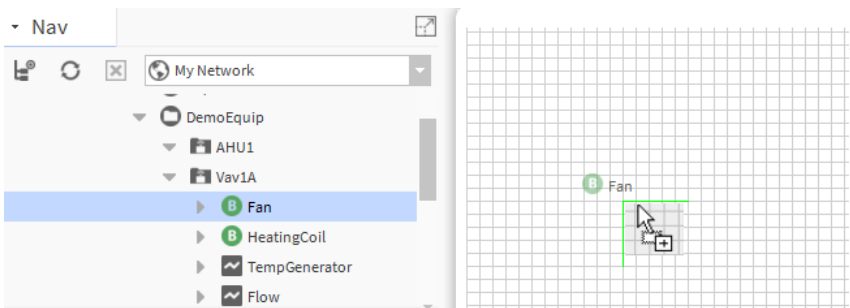


Making Action widgets

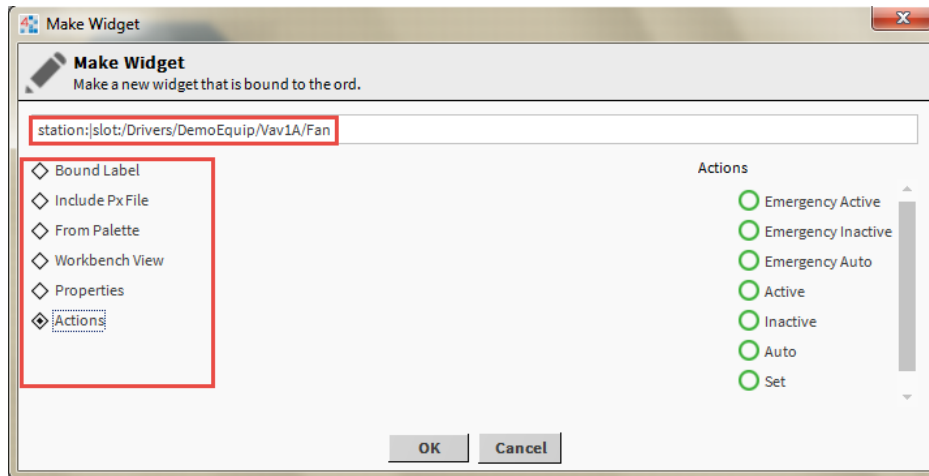
This procedure creates an Action widget using the **Make Widget** wizard.

Prerequisites: A Px View canvas is open. No open palette is required.

Step 1 Open the Nav side bar and expand the tree to locate the component to add to the **Px Editor** canvas.



Step 2 Drag the desired component to the canvas.
The **Make Widget** wizard opens.

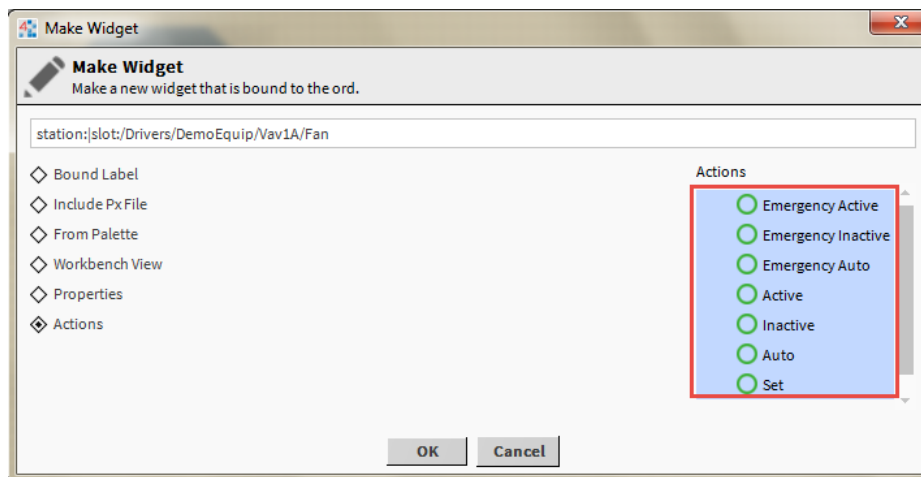


The ord for the selected component displays in the ord area.

Step 3 To bind to a different slot or component, double-click the ord and browse to the different slot in the component (for example: out or value) or to a different component.

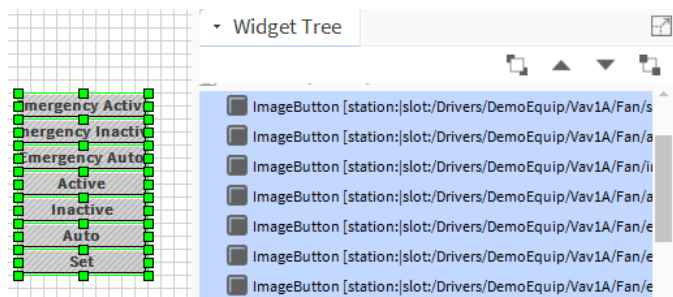
Step 4 Select the **Action** from the Source options list.


The Properties view area displays the actions that are available for creating the new widget.

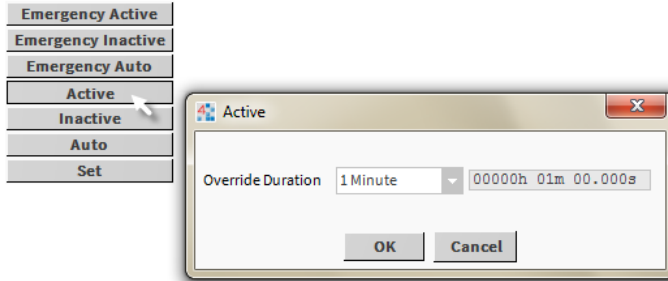


Step 5 In the Secondary side view area, select one or more actions from the list and click **OK**.

The Make Widget wizard closes and the new property widgets appear on the **Px Editor** canvas.



Step 6 To display the result in the **Px Viewer**, click Toggle View/Edit Mode () in the tool bar.

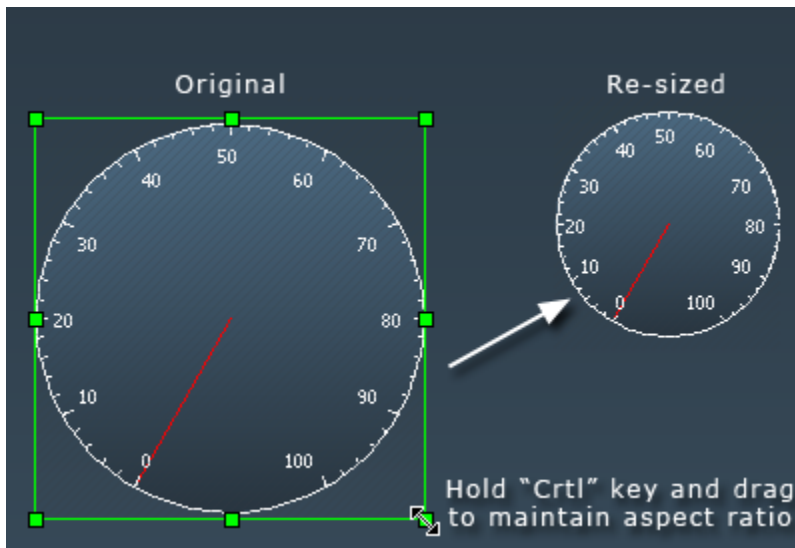


About Widget sizing in Px Editor

Sizing a widget is part of customizing the view.

To resize a widget on the canvas pane you must be in **Edit** mode. Click and drag one of the widget bounding box handles. To maintain the aspect ratio of the widget (maintain a proportionate relationship between the widget's height and width) press and hold the Ctrl key before you select and while you drag the widget bounding box.

Figure 13 Resizing a widget while maintaining aspect ratio



NOTE: Workbench supports the use of SVG images, which retain image quality when scaled up or down in size.

Embedding a Px View in another Px View

Using the PxInclude Widget and ORD Variables you can embed a single Px file in another Px file. The following is a general workflow to do this.

- Step 1 Design and create (or open an existing) Px file to use as an included or child Px file.
This is the file to embed in a parent Px file using the **PxInclude** widget.
- Step 2 If you are using any bound variables, consider whether or not to use an ord variable for the ord property value of any bindings to assign only at the parent Px file level, then create the variables using the `$(var)` syntax.
- Step 3 Save the child Px file in the desired location.
- Step 4 Create (or open an existing) parent Px file.

This is the file that uses the **PxInclude** widget to embed the child Px file.

Step 5 Add a **PxInclude** widget to the parent Px file using one of the following methods:

- Drag (cut and paste) the widget from a palette and edit the **ord** property to add the child Px file.
- Drag (cut and paste) the child Px file from the Nav tree to add the widget to the parent Px file with the child Px file already defined in the **PxInclude** **ord** property.
- Drag (cut and paste) a component from the Nav tree to initiate the **Make Widget** wizard. Use the **Make Widget** wizard to select the child Px file and choose which variables to bind.

Step 6 Select the **PxInclude** widget and edit the following properties, as required:

- **Enabled**, **Layout**, and **Visible**—leave set to default values.
- **ord**—click the value and define or browser to select the child Px file. If you used the **Make Widget** wizard, this property may already be defined.
- **variables**—to open the **Variables** window, click the value property. Use the editor property to define an **ord** value for each variable that appears. Notice the **after** column to verify the correct **ord** path for each variable.

Step 7 To close the **Variables** window, click **OK**.

Step 8 Save the parent Px file and attach it as a view to a component, if desired.

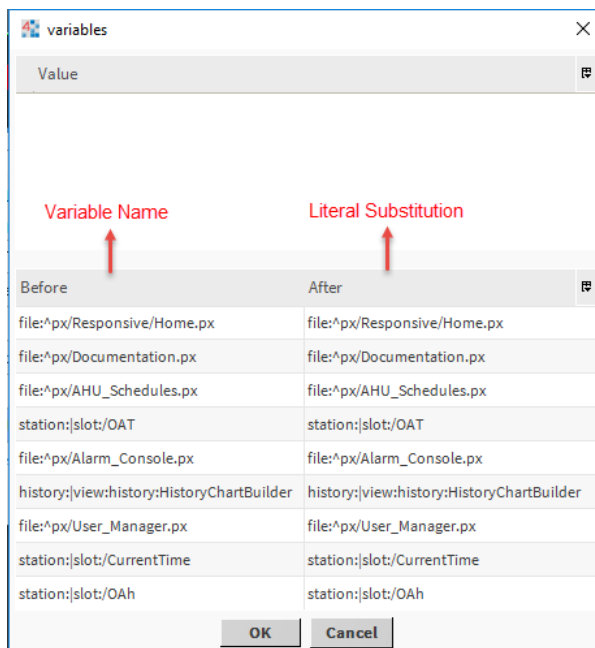
Step 9 To verify that the variable **ord** paths are set correctly, view the parent Px file from the component view (if used).

Example of a PxInclude widget with a single ORD variable

The following example illustrates using a **PxInclude** widget with a single **ORD** variable.

The figure shows an example of the **variables** window with one variable assigned.

Figure 14 Example: PxInclude Widget with a Single ORD variable



This window opens when you double-click on the **variables** property in a **PxInclude** widget.

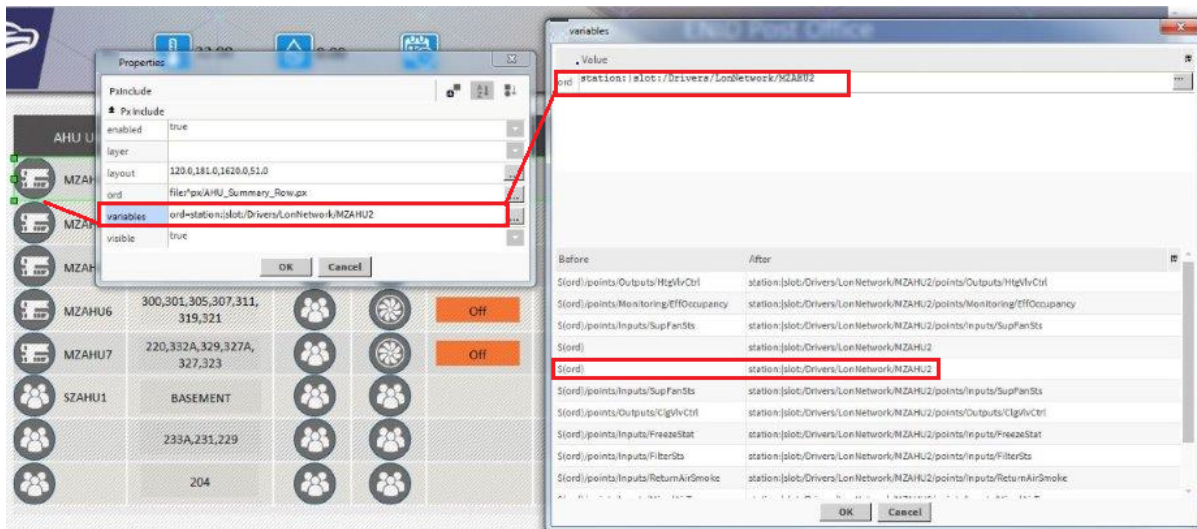
- The top half of the window is an editor. The text string you type a text string appears in the **Value** column as a substitute for the variable. Each literal text string replaces each variable that has been set up in the child Px file.
- This example shows a single variable that is used in four instances. Each instance of the variable `ord` has a different point appended to it.
- The **Before** column (bottom-left area of the window) displays each instance of the variable as defined in the child Px file.
- The **After** column (bottom-right area of the window) displays the results of any edits entered in the editor. This is the literal `ord` value that the PX file uses.

Example of PxInclude Widgets with multiple ORD variables

This example illustrates how PxIncludes can use more than one ORD variable.

The figure shows a **Px Editor** view of a Px file. This Px file has a **PxInclude Widget** with multiple variables defined, as shown in the widget's **Properties**, and **variables** windows.

Figure 15 Example: PxInclude Widgets with Multiple ORD Variables



This window opens when you double-click on the **variables** property in a PxInclude widget.

- The **PxInclude widget** is placed in the bottom left corner of the **Px Editor** view of a parent Px file (displaying three of the four included ORD variables).
- You define the ORD variables by typing in the editor. What you type appears in the **Value** column of the **variables** window. The variable values display in the **After** column in the lower half of the window.
- Defined variables display in the **PxInclude Properties** window (or **Properties** palette) with multiple variables displaying in the **variables** property value field separated by commas.
- Values display in the **PxInclude** widget area only after the variable is defined (using the **Variables** window).

Chapter 4 Animating graphics (data binding)

Topics covered in this chapter

- ◆ About data binding
- ◆ Add a data binding to a widget
- ◆ Animate a widget property
- ◆ Animate using static SVG images
- ◆ Relativize absolute Ords
- ◆ Types of data bindings

Animated graphics are comprised of one or more widgets assembled in a Px file, available for display using the Workbench display media types. Animated graphics change, or update, based on data values that come from object sources that are connected (or bound) to them.

A graphic can be as simple as a single word of text ON or a number 72, or it can be an animated image such as a rotating fan. Widgets provide the graphic visualization of data in Workbench. Animated graphics are comprised of one or more widgets assembled in a Px file, available for display using the Workbench display media types.

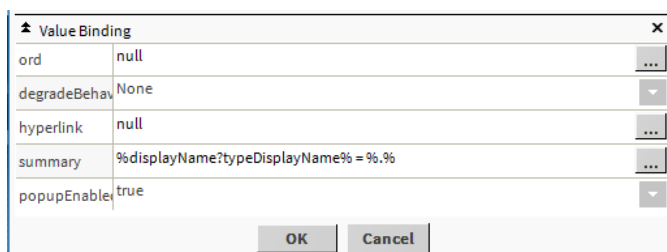
Widgets are animated by binding any widget properties to a legitimate data source. This means that you can connect numeric values to widget properties that use numeric values and you can connect binary values to objects that can use binary values. By animating the properties of a widget, you can control text and image appearance as well as a change a widget's location on the page and even its visibility.

About data binding

Bindings are established between a widget and an object. Binding provides real-time information for presentation.

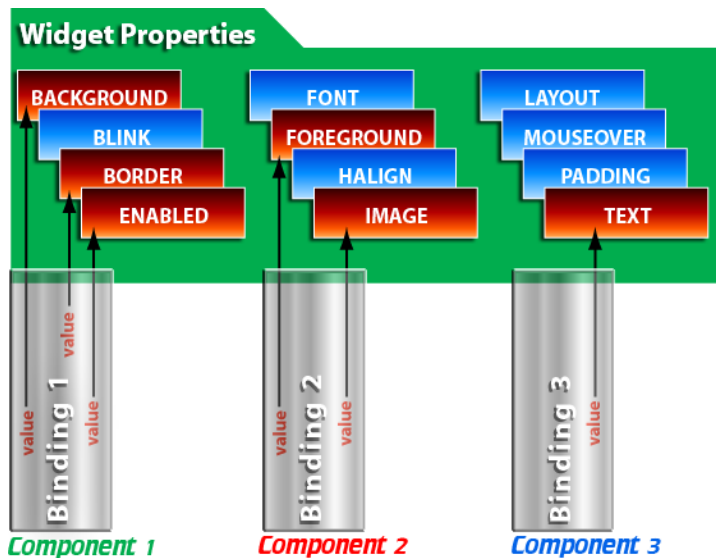
All widgets may be bound to data sources using data binding. An ord links a bindings to a widget. A single binding consists of a single widget-object relationship. A binding's ord property identifies the location of the object that updates and animates the widget.

For example, the most common type of binding, the value binding, provides some of the typical functions that are associated with building real-time information for presentation as both text and graphics. This includes support for mouse-over status and right-click actions. Additionally it provides a mechanism to animate any property of its parent widget using converters that convert the target object into property values. The following figure shows a value binding.



The following figure illustrates the object-to-widget property binding concept. In this example, a widget has three separate data bindings. This means that each binding is coming from a different object and therefore each binding has a different ord that defines its binding. Each binding provides access to an object's values so that they may be used, as required, to animate the widget properties.

Figure 16 Widget with three bindings




Add a data binding to a widget

Add a binding to a widget that is already on the Px Editor canvas by editing properties.

There are different ways to add a binding to a widget. You can add a binding to a widget using the **Make Widget** wizard or you can edit the widget properties as described here.

Step 1 In the PxEditor canvas, select the desired widget.

Step 2 In the **Properties** side bar, click the **Add Binding** button ().

Step 3 In the **Add Binding** window, select a binding type from the options list and click the **OK** button. The binding type properties are added to the binding area at the bottom of the widget property sheet.

Step 4 Under the binding type properties, click the **ord** property. The **ord** window opens.

Step 5 In the **ord** window, type or browse to the value to bind to the widget.

Step 6 Click the **OK** button.

The ord is added to the binding area of the widget property sheet.

Animate a widget property

This topic describes to animate a widget property.

Animating a property means to link a widget property to a bound data component value, so that the widget can display any change in value as it occurs.

NOTE: Before you can animate a widget property, you must add a binding from the desired component to the target widget. The binding must exist before the property can be animated.

Step 1 In the Px Editor canvas, select (or double-click) the widget to animate.

The widget properties display in the **Properties** side bar (or window, if you double-clicked).

Step 2 In the **Properties** side bar (or window) right-click on the property to animate and select **Animate** from the popup menu.

The **Animate** window opens. If you have more than one binding associated with the widget, all bindings are available in the **Binding** options list.

Step 3 From the **Binding** options list, select the binding that is associated with the component value to which to link.

Step 4 From the **Converter Type** options list, select the binding that is associated with the component value to link to.

NOTE: The default converter type is based on the component that you have selected and most of the time this is the converter type to use.

The **Animate** window displays different properties, based on the type of value (and converter) that you are animating.

Step 5 Complete the binding to the value using the controls in the **Animate** window and click **OK**.

The **Animate** window disappears and the binding opens in the properties side bar (or window).

Step 6 If using the **Properties** window, click **OK** to close it.

The property value is animated.

Animate using static SVG images

This type of animation requires at least two SVG images, of identical size and shape. The animation is caused by alternately displaying one image and then the next. You can apply this animation to a **Picture** widget or **Label**.

Prerequisites:

- A recent HTML5 compliant browser, for example: FireFox, Chrome, Opera, IE (v.10 or later) and Safari.
- The `svgBatik` module (`svgBatik.jar`) must be in the `Modules` folder.

The following example uses the **Picture** widget and two arrow images, in one the arrow points up, in the other it points down.

Step 1 In the station, create a new `EnumWritable` point to use as the data source for the animation.

Step 2 Set the range values for the **Facets** as follows:


- 0 = Up
- 1 = Down

Step 3 Right-click the component and select **Views**→**New View** to open a new Px view.

Step 4 Select **PxViewer**→**Toggle View/Edit mode** to switch to Edit mode.

Step 5 Open the **bajoui** palette, expand the `Widgets` folder and drag the **Picture** widget to the Px Editor canvas.

Step 6 Double-click the widget to open the **Edit Properties** window and click the **Add Binding** button (in the upper right corner), to add a value binding to the widget.

Step 7 In the Value Binding section of the **Properties** window, click on the `ord` property of the binding and click the selection icon () to access the Component Chooser.

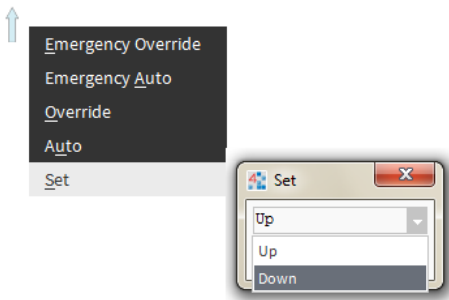
- In the Component Chooser, locate and select the `out` value of the `EnumWritable` data source created in step one, for example: `station:|slot:/Drivers/Points/EnumWritable/out/value`, and click **OK**.

Step 8 Open the `kitPxN4svg` palette, expand `Piping` folder and drag the `DirectionArrowDown` and `DirectionArrowUp` image files to the Px Editor canvas.

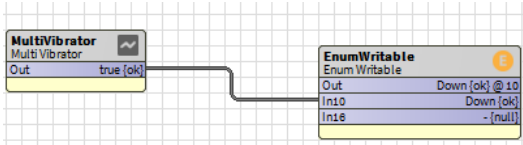
NOTE: The `kitPxN4svg` and `Piping` folders are automatically generated under the station's `Files` folder.

- Step 9** Double click on the **Picture** widget, **Properties** window opens.
- Step 10** In the **Properties** window, click the selection icon to access the File Chooser.
- In the File Chooser, locate the added **DirectionArrowUp** image file (for example, `file:^px/kitPxN4svg/Piping/Direction_Arrows/DirectionArrow_Up.svg`)
- Step 11** In the **Picture** widget section of the **Properties** window, right-click the **image** property and click **Animate**.
- In the **Animate** window, select the first facet of the **EnumWritable** and click **Set** and use the File Chooser to locate and select the first image file (for example, `file:^px/kitPxN4svg/Piping/Direction_Arrows/DirectionArrow_Up.svg`) to be displayed for facet value 0 and click **OK**.
 - Select the second facet of the **EnumWritable** and click **Set** and use the File Chooser to locate and select the second image file (for example, `file:^px/kitPxN4svg/Piping/Direction_Arrows/DirectionArrow_Down.svg`) to be displayed for facet value 1 and click **OK**.
- Step 12** Click **Save** to save these changes to your Px View.

The graphic image changes automatically when you set the **Actions**. The image displayed depends on whether the **EnumWritable** Action is set to **Up** or **Down**. For example, in View mode, right-click the displayed arrow image and select **Actions**→**Set** and change to **Down**:



Additionally, providing an input to the **EnumWritable** using a control component, such as **MultiVibrator** in the **kitControl** palette, causes the animation to occur automatically. For example, drag a **MultiVibrator** component onto the wire sheet view for the **EnumWritable** and set the **MultiVibrator** **Period** property to 5 seconds (so that the **out** value changes from **true** to **false** every five seconds). In the wire sheet view, connect the **MultiVibrator** **out** value to the **EnumWritable** **In**:



Switch to the Px Editor View mode to observe the animation.


Relativize absolute Ords

Open the Relativize ords window from the **Bound Ords** palette. This Px Editor tool provides a convenient way to change absolute ords to relative ords.

When you use the Px Editor tools to bind data, the ord is usually supplied in an absolute format, by default. For example, `station:|slot:/Logic/HousingUnit/AirHandler/DamperPosition`. If you change the data binding to make it relative, then the path resolves relative to its current parent ord. This relative path makes the Px file resolve data bindings correctly to identically named components that reside in different locations, making this one Px file reusable in many views.

- Open an existing Px view that contains one or more widgets with bindings that have absolute ords.
- In Edit mode, in the **Bound Ords** area confirm that there are absolute ords.

TIP: An absolute ord is relative to the station and shows the entire file path from the station's **Config** node.

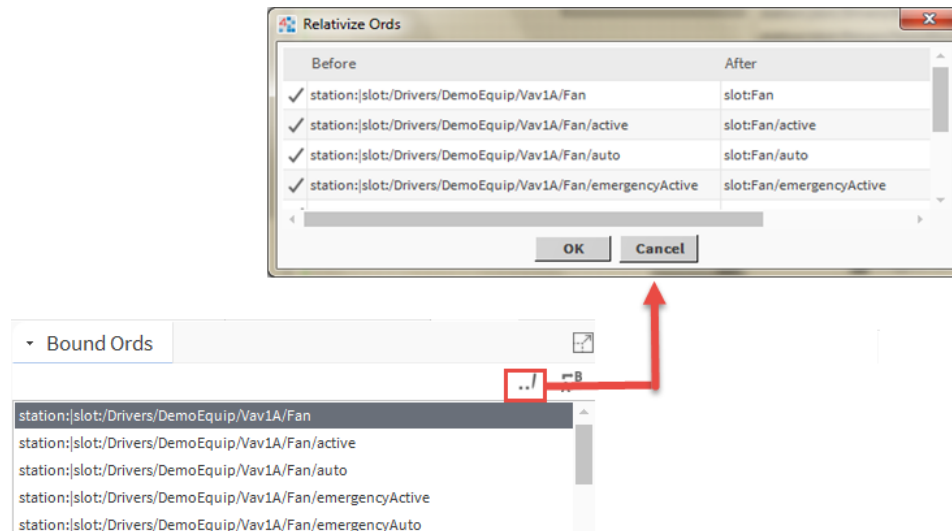
Step 3 In the **Bound Ords** area, click the **Relativize Ords** () button.

The **Bound Ords** window opens listing all of the ords that can be relativized.

Step 4 Click the **OK** button.

The **Bound Ords** area, you can see the shortened file path of each relativized ord. The ords are relative to the current parent ord.

The following image shows the relativizing absolute ords:



Types of data bindings

There are different types of bindings that may be used with widgets.

Some bindings work with only a certain type of widget (for example, a bound label binding) and other binding types may be used with several types of widgets (for example, a value binding). Shown here are bindings as they appear in an options list.

The following list describes each binding type.

- **Action binding**
Invokes an action on the binding target component when an event is fired by the parent widget.
- **Bound Label binding**
Connects a value to a bound label widget.
- **Field Editor binding**
Used to bind field editor components to an object.
- **Popup binding**
Used to display a Px view in an additional popup window that you can specify and configure.
- **Setpoint binding**
Used to display the current value of a setpoint and also to provide the ability to modify it.
- **Spectrum binding**
Used to animate a widget's brush (color) property.

- **Spectrum setpoint binding**
Used in conjunction with a spectrum binding.
- **Table binding**
Used to bind table data in a bound table.
- **Value binding**
Used to bind to values that are typically under a component.

Types of binding properties

One or more of the following properties may be included with various binding types.

- **actionArgument**
This property works with certain widgetEvents to specify the action to take when the widgetEvent is triggered. For example, a mouseEvent, such as a click on a widget can change a Boolean status from true to false or prompt the user to select a setting. Argument options, like the widgetEvents, themselves, are context sensitive, depending on widget and binding type.
- **degradeBehavior**
This property specifies how the object behaves when binding communications are not available. If a binding is not usable, this property allows the designer to choose how to degrade the UI gracefully. For example, if the user does not have permission to invoke a specific action, a button that is bound to that action can be dimmed or hidden entirely. To preserve backward compatibility, the default degradeBehavior is `none`.
- **extent**
This spectrum binding property represents the total range of the bound value which maps from low to high.
- **highColor**
This spectrum binding property specifies the color of the highest value assignment for a spectrum color binding. The high color displays when the bound target is greater than $\text{setpoint} + \text{extent} / 2$. As the bound value decreases below the maximum value specified, the color approaches the color set by the `Mid Color` property.
- **hyperlink**
This property provides a link to another object. When used, the hyperlink is active in the browser or in the Px viewer.
- **icon**
This property specifies an icon to appear in the top left corner of a popup window. The icon property is not available with the popup binding.
- **increment**
This property is a value that can be set to increment (increase) or decrement (decrease) a current value by the assigned amount. A positive number in this property field increments a value; a negative number in this field decrements a value.
- **lowColor**
This spectrum binding property specifies the color that is used for the lowest value assignment for a spectrum color binding. The low color displays when the bound target value is less than $\text{setpoint} - \text{extent} / 2$. As the value bound to this property increases above the minimum value specified, the color approaches the color set by the `Mid Color` property.
- **midColor**

This spectrum binding property specifies the color that is used for the middle value assignment for a spectrum color binding. This color is displayed when the bound value is exactly at the setpoint. As the binding value increases above this point, the displayed color approaches the color set by the High Color property. As the bound value decreases below this point, the displayed color approaches the color set by the `Low Color` property.

- **modal**

This property applies to Popup Bindings. When set to `true`, the popup window associated with the Popup Binding is a modal window.

NOTE: A modal window is a child (or secondary) window that appears in front of a parent window and, typically, does not allow interaction with the parent window. The modal window created using this property is always on top but does not prevent a user from interacting with the parent window.

- **ord**

This property specifies the ORD to bind to the widget. This property must have a value for the widget to be bound. In a Popup binding this is the path that designates the component view that displays in the popup window.

- **popupEnabled**

This property allows (`true`) or prohibits (`false`) a secondary popup (right-click) menu to appear when a user clicks on this label from a browser or Px viewer.

`false` prevents actions from being executed at a control point from a specific Px graphic while still allowing access to the action from a different Px graphic or Property Sheet.

NOTE: Other ways to make actions unavailable include:

- Hiding the action slot on the component makes the action unavailable but it would apply to every point of access in the station.
- Actions can also be assigned operator or admin permission requirements to limit the access.

position

This property designates an initial screen position for a popup window.

- **setpoint**

This binding property specifies the value that is used for displaying the Mid Color. For example, if you set this value to `70`, the color that you define for Mid Color displays when the bound value is `70`.

- **size**

This property designates an initial screen size for a popup window.

- **statusEffect**

this property provides three options for enabling or disabling status effects.

- **Color**

This option enables a background color change when the bound value status changes.

- **Color and blink**

This option to enables a background color change and a blinking effect when the bound value status changes.

- **None**

This option disables any effects based on bound value status changes.

- **summary**

This property specifies a display name for the widget in text or by means of a script.

- **title**

This property specifies the text that appears in a popup window title bar.

- **widgetEvent**

This property specifies one of several possible events: `actionPerformed`, `focusEvent`, `invokeAction`, `keyEvent`, `mouseEvent`. Possible actions depend on the type of widget and action. Some widgetEvents specify an `actionArgument`, as described, above.

- **widgetProperty**

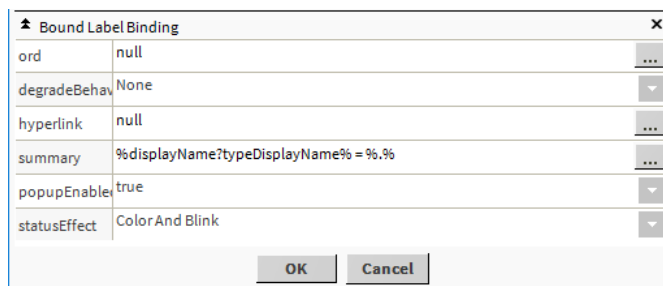
This property allows you to specify the property that you are targeting in the binding's parent widget. For example, you can use a spectrum binding that has a bound label parent to change the background property of that parent label by selecting `background` in the Widget property. Or, you can change the foreground color by choosing `foreground`. You can target only one value per binding but you may add additional bindings if you want to target more than one property in the parent widget.

About bound label bindings

Bound labels connect a value to a bound label widget.

Bound label bindings are used exclusively for connecting a value to a bound label widget. Bound labels, which are located in the kitPx module, have properties that you can edit and access from the Px Editor properties side bar. The properties pane for the bound label binding is shown below.

Figure 17 Bound label binding properties



To access these properties after dragging a **BoundLabel** from the **kitPx** palette to the Px Editor, double-click the bound label.

Bound label binding properties include the following:

- ORD
- Hyperlink
- Summary
- Popup Enabled
- Status Effect

About value bindings

Value bindings support real-time graphics, mouse-over effects, and right-click actions.

Value bindings bind to values that are located, typically, under a component. Widget properties may be overridden by creating dynamic slots of the same name with a converter that maps the bound target object into a property value. This capability is what animates any widget property. Value bindings support features, such as real-time graphics, mouse-over, and right-click actions.

Value binding properties include the following:

- ORD
- Hyperlink

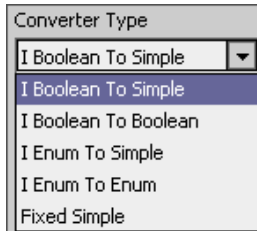
- Summary
- Popup Enabled

Types of Converters

Converters are used to change data from one type to another so that the data can be used to communicate with a particular widget property value.

In most cases, when you animate a property, the correct data converter appears by default at the top of the list, as shown.

Figure 18 Types of converters



The following types of converters are available when using a value binding:




- Fixed Simple
- I Numeric to I Simple
- I Status to Simple
- Object to String

About spectrum bindings

Use this binding to animate a widget's brush (color) property by mapping a numeric value into a color range defined by lowColor, midColor, and highColor properties.

The properties pane for the spectrum binding properties is shown below.

Figure 19 Spectrum binding properties

| Spectrum Binding | | x |
|------------------|---|-----|
| ord | null | ... |
| degradeBehavior | None | ▼ |
| widgetProperty | | ▼ |
| lowColor |  | ... |
| midColor |  | ... |
| highColor |  | ... |
| setpoint | 50.00 | |
| extent | 100.00 | |

Spectrum properties include the following:

- ORD
- degardeBehaviour
- Widget Property
- Low Color
- Mid Color
- High Color
- Set Point
- Extent

About setpoint bindings

Use this binding to display the current value of a setpoint and also to provide the ability to modify it.

A setpoint is a status value property such as `fallback`.

The setpoint binding ORD must resolve down to the specific property that is being manipulated. If it is bound to a component or to a read-only property, then the binding attempts to use a set action to save.

The properties pane for the spectrum setpoint binding is shown in

Figure 20 Setpoint binding properties

| Set Point Binding | | x |
|-------------------|-------------------------------------|-----|
| ord | null | ... |
| degradeBehav | None | ▼ |
| hyperlink | null | ... |
| summary | %displayName?typeDisplayName% = %.% | ... |
| popupEnabled | true | ▼ |
| widgetEvent | | ▼ |
| widgetPropert | | ▼ |

Setpoint properties include the following:

- ord
- degradeBehavior
- Hyperlink
- Summary
- Popup Enabled
- Widget Event
- Widget Property

About increment setpoint bindings

Use this type of setpoint binding to increment or decrement a numeric value.

The properties pane for the increment setpoint binding is shown below.

Figure 21 Increment setpoint binding properties

| Increment Set Point Binding | | x |
|-----------------------------|-------------------------------------|-----|
| ord | null | ... |
| degradeBehav | None | ▼ |
| hyperlink | null | ... |
| summary | %displayName?typeDisplayName% = %.% | ... |
| popupEnabled | true | ▼ |
| widgetEvent | actionPerformed | ▼ |
| increment | 1.00 | |

Increment setpoint setpoint properties include the following:

- ORD
- degradeBehaviour
- Hyperlink
- Summary
- Popup Enabled

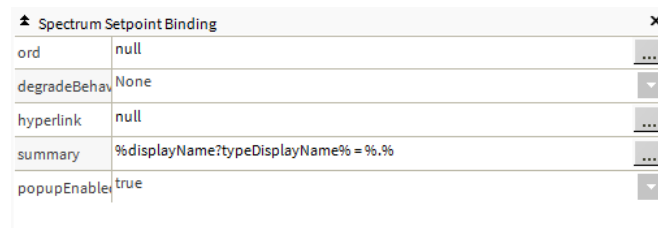
- Widget
- Increment

About spectrum setpoint bindings

This binding is used in conjunction with a spectrum binding to animate the midColor properties.

The properties pane for the spectrum setpoint binding is shown below:

Figure 22 Spectrum setpoint binding properties



Spectrum setpoint properties include the following:

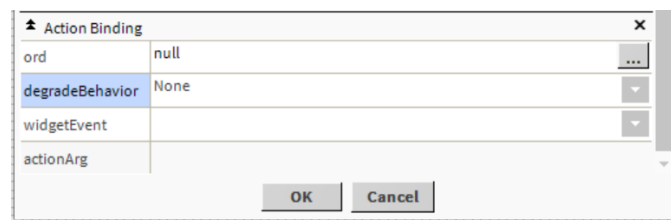
- Ord
- degradeBehaviour
- hyperlink
- summary
- popupEnable

About action bindings

This type of binding invokes an action on the binding target component when an event is fired by the parent widget.

The ORD of an action binding must resolve down to a specific action within a component. Examples of actions include: active, inactive, override, and other commands. The properties pane for the action binding is shown here.

Figure 23 Action binding properties



Action binding properties include the following:

| Property | Value | Description |
|------------------|-----------|--|
| ord | path | You can use ord property to choose the location of the data value to bind the widget. |
| degrade behavior | drop-down | Widget disappears when you set <code>Hide</code> , security checks shows permission not allowed. Widget disables when you set <code>Disable</code> and security checks shows permission not allowed. |

| | | |
|-------------|-----------|--|
| widgetEvent | drop-down | Widget Event drop-down list Defines the action to perform. |
| actionArg | read-only | |

About table bindings

This type of binding is used to bind table data in a bound table.

Table bindings can bind to any collection using a BQL, SQL, or History ORD binding. For example, a table binding ORD might look like the following:

```
station:"slot:/"bql:select toPathString,toString from control:BooleanPoint
```

The properties pane for the table binding is shown below:

Figure 24 Table binding properties



Table binding properties include the following:

- Ord
- degradeBehaviour

About field editor bindings

Field editor bindings are used to bind field editor components to an object.

A field editor is a component that is designed to view and edit an object property. Field editors are designed to be laid out on panes and are built with standard widgets like buttons, text fields, check boxes, and so on.

The properties pane for the field editor binding is shown here.

Figure 25 Field editor binding properties



Field editor binding properties include the following:

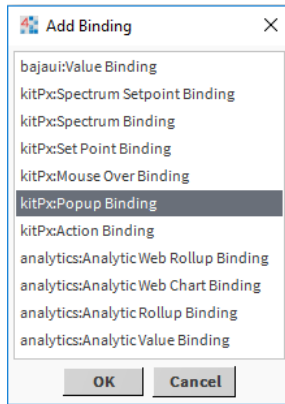
- Ord

About Popup Bindings

This type of binding may be used to open a Px view in an additional popup window that you can specify and configure in terms of size, location, and content.

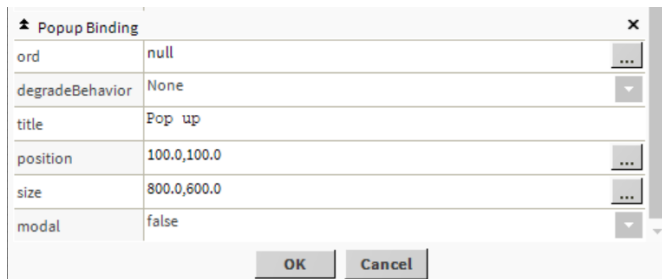
The Popup binding is attached to a Px object (such as a button) and configured using the Popup Binding properties. The figure below shows the Popup Binding listed in the **Add Binding** window.

Figure 26 Popup Binding in the Add Binding Window



The Popup Binding has the following properties:

Figure 27 Popup Binding properties



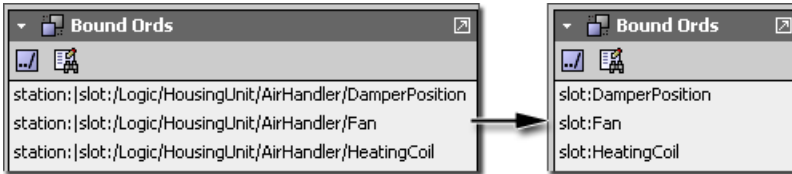
| Property | Value | Description |
|------------------|-------------------------|--|
| ord | path | You can use ord property to choose the location of the data value to bind the widget. |
| degrade behavior | None or Disable or Hide | Widget disappears when you set <code>Hide</code> , security checks shows permission not allowed. Widget disables when you set <code>Disable</code> and security checks shows permission not allowed. |
| title | text | By default title Pop up appears but User can add any title in web (customize title). <code>%lexicon(kitPx:popupBinding.title)%</code> format also available. It is use to add a title for multilingual stations. |
| position | number | User can set the position of Popup Binding as per requirement. |
| size | number | User can set the size of Popup Binding as per requirement. |
| modal | true or false | User get access to original dialog when set the new dialog with modal <code>true</code> and user need to complete the user dialog before they switch the window when set the modal <code>false</code> . |

About relative and absolute bindings

Data bindings, like ORDs, can be relative or absolute.

Since ORDs can be relative or absolute and widgets are bound to data sources using an ORD, data bindings can be relative or absolute, as shown here.

Figure 28 Absolutely bound ORDs and relatively bound ORDs



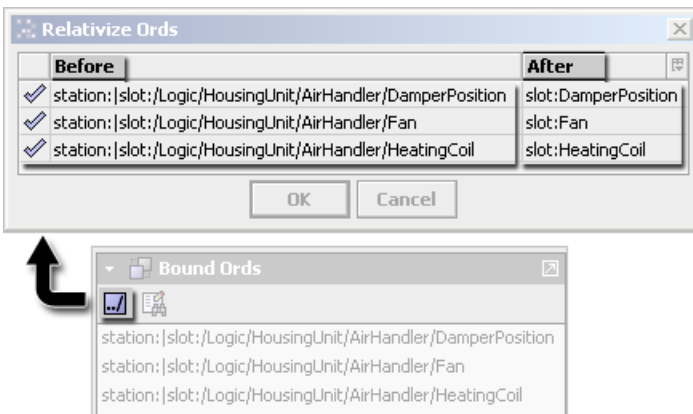
This is a particularly important point to remember if you want to design Px files that are used in multiple views on a local station or even possibly for use across different stations. When you use the Px Editor tools to bind data, the ORD is usually supplied in an absolute format, by default. The ORD is relative to the station, such as:

```
station:|slot:/Logic/HousingUnit/AirHandler/DamperPosition
```

This absolute path ensures that the data always resolves to a single unique component (DamperPosition) in the location that is specified by the ORD, regardless of where the Px file or the parent component is located. If the same Px file is attached to a view that belongs to a different component, the ORD will still resolve to the original DamperPosition component because of the absolute path. However, if you make data binding relative, then the path will resolve relative to its current parent ORD. This relative path makes the Px file resolve data bindings correctly to identically named components that reside in different locations, thus making one Px file usable in many views.

You can open the **Relativize ORDs** window from the Bound ORDs palette, as shown. This Px Editor tool provides a convenient way to change absolute ORDs to relative ORDs.

Figure 29 Relativizing bound ORDs



About tag-based NEQL bindings

Niagara 4.9 and later has added support for tag-based bindings. Tag-based bindings use NEQL queries and resolve NEQL Ords instead of using more traditional Ord types, such as slot Path Ords.


In these descriptions, the component on which a Px view is placed is referred to as the base component. A Px view with relativized slot path Ords is more reusable than one with absolute slot path Ords but it still requires components to be positioned the same and named exactly the same on other base components. Tag-based Px bindings use NEQL queries instead of slot paths for the bound ORDs. The result is that components can have different names and be placed anywhere in the station as long as there is a set of tags to identify the component and a (optional) relation to get from the base component to that bound component.

If there is not a relation between the base component and the bound component but the bound component is a descendant of the base component, a Select NEQL query can be used for the Ord binding. If the bound

component is not a descendant of the base component, there must be a relation and a Traverse NEQL query will be used.

Figure 30 Neqlize Ords (Convert to tag-based) window

| Before | After | Path |
|---|--|---------------------------------|
| <input type="checkbox"/> neql:tag1 single: | neql:tag1 single: | Failed to resolve Before Ord |
| <input type="checkbox"/> neql:tag3 single: | neql:tag3 single: | slot:/Points/BooleanWritable3 |
| <input type="checkbox"/> neql:traverse hs:equipRef<- where a:tag1 single: | neql:traverse hs:equipRef<- where a:tag1 single: | No identifying tags found |
| <input type="checkbox"/> neql:traverse hs:equipRef<- where a:tag3 single: | neql:traverse hs:equipRef<- where a:tag3 single: | slot:/External/NumericWritable3 |
| <input checked="" type="checkbox"/> slot:BooleanWritable2 | neql:tag2 single: | slot:/Points/BooleanWritable2 |
| <input checked="" type="checkbox"/> station:slot:/External/NumericWritable2 | neql:traverse hs:equipRef<- where a:tag2 single: | slot:/External/NumericWritable2 |
| <input type="checkbox"/> station:slot:/External/NumericWritable4 | station:slot:/External/NumericWritable4 | No identifying tags found |
| <input type="checkbox"/> station:slot:/External/NumericWritable5 | station:slot:/External/NumericWritable5 | Not an endpoint or descendant |
| <input type="checkbox"/> station:slot:/Points/BooleanWritable4 | station:slot:/Points/BooleanWritable4 | No identifying tags found |

Invoked in a **Px Editor** view, from the **Bound Ords** pane by clicking the  (Neqlize Ords) button. The **Neqlize Ords** window attempts to find a set of tags and optionally a relation to distinguish a component in the bound ORDs list from a search set. The search set depends on what type of query you want.

The **Neqlize Ords** window shows the **Before** conversion value of the slot path Ords, the **After** conversion value of the tag-based NEQL Ords with tags and an optional relation, and the **Path** value shows the absolute slot path ord to that component because any previously converted NEQL queries may resolve to anywhere in the station.

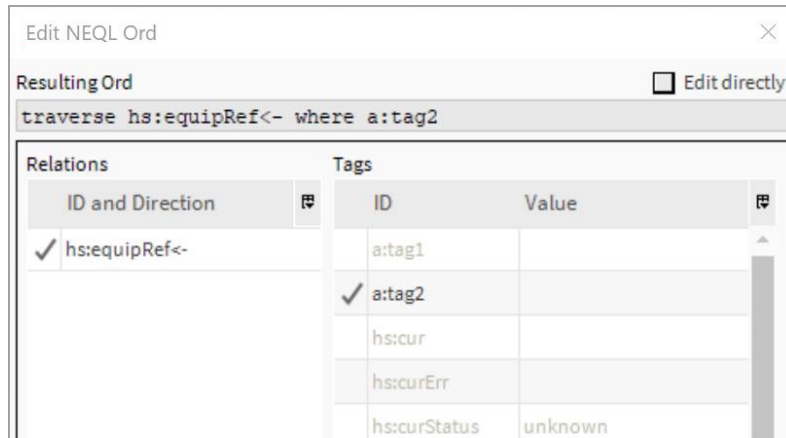
Additional buttons

- **Options** opens the **Workbench Neqlize Options** window.

- **Refresh All** refreshes all selected rows in the **Neqlize Ords** window.

Right-click menu options

- **Edit NEQL Ord** opens the **Edit Neql Ord** window. When selected, the **Edit directly** checkbox gives you full control to make changes to the Resulting Ord value of your NEQL query. When not selected, however, you can select different tags and relations to use. The query is generated based on your selections.



- **Refresh Row** which is invoked by right-clicking on a single row.

Types of NEQL queries

There are two types of queries in NEQL:

- A **Traverse** query uses a single relation ID and direction to get from a base component to a set of endpoints. If a component in the bound ORDs list is one of those endpoints, an algorithm compares the tags on that component to the tags on the other endpoints. If there is a set of tags that distinguishes the component from the other endpoints, that set of tags, the relation ID, and relation direction are returned. If there is not a distinguishing set of tags, another search is made with another relation on the base component.
- A **Select** query operates on the descendants of a base component. If a component in the bound ORDs list is one of those descendants, an algorithm compares the tags on that component to tags on the other descendants. If there is a set of tags that distinguishes the component from the other descendants, that set of tags, but no relation ID or relation direction, is returned.

Within the Workbench **Tools**→**Options** for **Px Editor**, there are three query modes for finding a set of tags and optional relation:

- **Traverse If Possible** - attempts to find a traverse query but falls back to a select query.
- **Traverse Only**- attempts to find a traverse query only; an error is returned if the bound component is not an endpoint of one of the base component's relations or there is not a set of tags that distinguishes the bound component from other endpoints.
- **Select Only**- attempts to find a select query only; an error is returned if the bound component is not a descendant of the base component or there is not a set of tags that distinguished the bound component from other descendants.

In addition to the query mode, you can specify certain tags and relations to exclude from the query to keep the graphic as reusable as possible. A few examples of tags and relations that limit reusability, and that are already listed in default exclusions, are shown here:

- **Tags:** n:name, n:displayName, n:ordInSession, hs:id
- **Relations:** n:child, n:parent

A set of Default Excluded Tags and Relations is collected from each installed tag dictionary. A set of Custom Excluded Tags and Relations can also be specified in the station's TagDictionaryService. Additionally, a set of User Excluded Tags and Relations can be specified in the Px Editor Workbench Options.

Chapter 5 Other elements to add to Px views

Topics covered in this chapter

- ◆ Add and configure a comment box
- ◆ Add a text entry field and save button
- ◆ Embed a History Table in your Px view
- ◆ Embed a history chart in a Px view
- ◆ Add WeatherReports on Px Views
- ◆ Launching a History Chart Builder in a Px view
- ◆ Applying visual styles to Px views
- ◆ Customizing the login screen
- ◆ Optimizing existing Px files for mobile devices
- ◆ Converting bound ORDs to tag-based NEQL ORDs
- ◆ Using the visible property in Hx profile (an example)

There are several ways to add a comments box to a Px view. This section explains one way.

There are a number of ways to accomplish this. This section describes one method.


You can provide a comments box on a Px view for users to enter comments or notes. All saved entries are displayed in the Px view as well. The basic workflow for this is as follows:

Add and configure a comment box

Part one of adding a comment box to a Px view is to add and configure components in your station.

Prerequisites:

- You are connected to the station with the PX Editor view open.

- Step 1 In the Px Editor palette side bar, open the **control** palette, and drag a **StringWritable** point to a location in the **Config** node on your station.
- Step 2 In the palette side bar, open the **history** palette, expand the Extensions folder and drag a **StringCOV** history extension onto the **StringWritable** point added earlier.
- Step 3 In the Nav tree, double-click on the added **StringCov** component to open its **Property Sheet** and set the **Enabled** property value to `true` and click **Save**.
- Step 4 In the Nav tree, right-click on the **StringWritable** point and select **Actions**→**Set** from the popup menu.
- Step 5 In the **Set** window, type some text to generate the first history record, for example: `First text entry` and click **OK**.
- Step 6 In the Nav tree, double-click on the **StringWritable** point to open its **Property Sheet**.
- Step 7 In the **Facets** slot, click on the configuration icon (»).
- The **Config Facets** window opens.
- Step 8 In the **Config Facets** window, click the **Add** button () and configure the facet key as follows, and click **OK**.
 - Key: `multiLine`
 - Type: `Boolean`
 - Value: `true`

Step 9 To complete the comment box, click **Save**.

Continue with the next procedure.

Add a text entry field and save button

This topic describes how to add necessary widgets to to the comment box to display a text entry field and a save button.

Step 1 In the Nav tree, drag your **StringWritable** point onto the canvas in your **Px** page.

Step 2 In the **Make Widget wizard**, do the following:

- Select **From Palette** as the source of the widget.
- In the drop-down list, select the `kitPx` palette.
- In the `kitPx` palette select a `SetPointFieldEditor` widget.
- Click the **OK** button.

Step 3 In the `PxEditor` canvas, double-click on the `SetPointFieldEditor` widget to open the **Properties** window.

Step 4 Configure the **Set Point Binding ord** property to the `in16` slot and click **OK**.

The text entry field displays on the canvas.

Step 5 In the **Palette** side bar, open the `kitPx` palette.

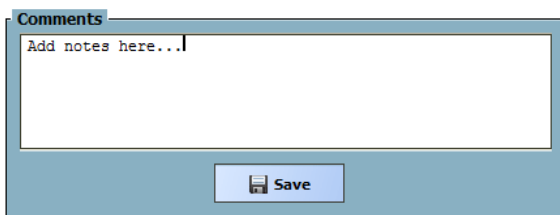
Step 6 Drag a **Save** button widget to a position near the text entry field on the canvas.

Step 7 Design your comments box as desired, for example, you could enclose the comments box and **Save** button in a Border pane and add a Label as shown below.

Step 8 To save your changes, click the **Save** icon (📁) in the toolbar.

Step 9 To test the comment box, switch to **View** mode by clicking the Toggle View/Edit Mode button (🔍), type some text in the text entry property and click the **Save** button.

Entering text activates the **Save** button. Your comments box should resemble the one shown here:



Step 10 In the Nav tree, expand the station's **History** node, locate the **StringWritable** history and double-click to open the **History Table** view.

| PxTest/StringWritable | | 5 records | |
|--------------------------|-------------|-----------|--------------------------|
| Timestamp | Trend Flags | Status | Value |
| 31-Mar-17 3:47:00 PM IST | {start} | {null} | |
| 31-Mar-17 3:47:40 PM IST | {} | {ok} | First text entry. |
| 31-Mar-17 4:13:52 PM IST | {} | {ok} | note added here... |
| 31-Mar-17 4:27:49 PM IST | {start} | {ok} | my latest entry |
| 31-Mar-17 4:28:36 PM IST | {} | {ok} | This is my latest entry. |

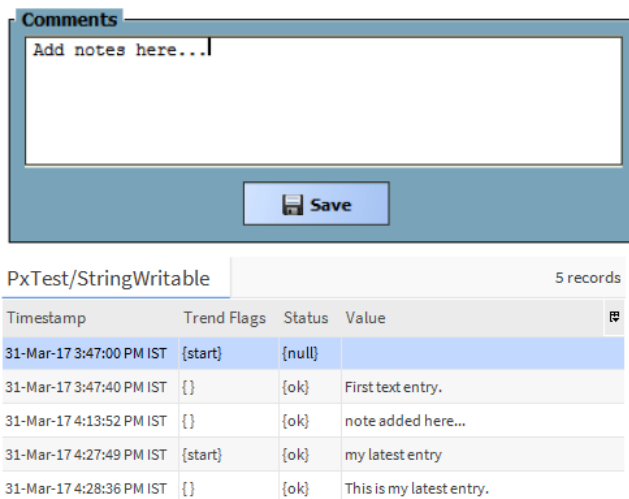
The **History Table** view contains a record of each saved text entry.

Embed a History Table in your Px view

This topic describes how to embed the **History Table** view in your **Px** page to display the saved comments.

- Step 1 Drag the **StringWritable** history component to the **Px** view.
- Step 2 In the **Make Widget** wizard, select **Workbench View** and **History Table**.
- Step 3 Double click on the added **StringWritable** history component, the **Properties** window opens.
- Step 4 In the **Properties** window, set the following values:
 - **defaultLiveUpdates** to `true`
 - **show time range editor** to `false`
 - **show delta editor** to `false`
 - **show live updates buttons** to `false`

After saving your changes, the **History Table** containing all the text entries displays in the **Px** view. When a new comment is entered and saved in the **Px** view, the **History Table** automatically updates with the latest entry. Your comments box and **History Table** should look like this:



The screenshot shows a 'Comments' widget with a text input field containing 'Add notes here...!' and a 'Save' button. Below it is a 'History Table' for 'PxTest/StringWritable' showing 5 records.

| Timestamp | Trend Flags | Status | Value |
|--------------------------|-------------|--------|--------------------------|
| 31-Mar-17 3:47:00 PM IST | {start} | {null} | |
| 31-Mar-17 3:47:40 PM IST | {} | {ok} | First text entry. |
| 31-Mar-17 4:13:52 PM IST | {} | {ok} | note added here... |
| 31-Mar-17 4:27:49 PM IST | {start} | {ok} | my latest entry |
| 31-Mar-17 4:28:36 PM IST | {} | {ok} | This is my latest entry. |

Embed a history chart in a Px view

Embedding a pre-configured, individual history chart in a **Px** view.

Prerequisites:

- A pre-configured history in the station's **History** node.

- Step 1 Drag the pre-configured history or history extension component onto your **Px** page.
- Step 2 In the **Make Widget** wizard, select **Workbench View** and **History Table**.
- Step 3 Click **OK** and click **Save**.

The individual history is embedded in the **Px** page.

You can use the export icon () or the **File**→**Export** menu option to export a history chart or history table. The history name and time range of the data are included at the top of the exported file (CSV, TXT, PDF), however, it shows value as the column identifier.

If you also want to provide access to the **History Chart Builder** in your **Px** view, you can add a popup binding on a widget that launches it.

Add WeatherReports on Px Views

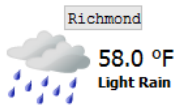
You can include one or more iconic displays of weather data on Px views.

Prerequisites:

- The **weather** module must be installed.
- Station has Internet connectivity, or is on a NiagaraNetwork with a Supervisor using the **WeatherService**.

- Step 1 In the **Palette** side bar, open the **weather** palette.
- Step 2 Drag the **WeatherService** component to the **Services** node.
- Step 3 Double-click on the **WeatherService** to open the **Weather Manager**.
- Step 4 Click the **New** button.
The **New** window opens.
- Step 5 Click **OK** button to add a new **WeatherReport** component in the **WeatherService** node.
The **WeatherReport** is configured with the default provider, `Nws Weather Provider`.
- Step 6 Drag the **WeatherReport** component to your Px page.
The **Make Widget** wizard opens.
- Step 7 Select the **Workbench View** and click the **OK** button to add **WeatherReport** to the Px page.
- Step 8 Select **PxEditor**→**Toggle View/Edit Mode** to switch to PxViewer.

The Px view includes a display of current weather data:



Launching a History Chart Builder in a Px view

You can launch the History Chart Builder in a Px view by adding a Pop Up binding on a widget in your Px page. The ord of the Popup binding is linked to the station's **History**. When finished with the chart builder, simply close the Popup to return to your Px view.

Prerequisites:

- A pre-configured history in the station's **History** node.
- Step 1 Drag an **ActionButton (ImageButton)** widget from the **kitPx** palette to your Px page.
 - Step 2 Double click on the **ImageButton** in the **Widget Tree** palette. **Properties** window pops up.
 - Step 3 In the **Image button** property sheet view, configure the following:
 - In the **text** slot, type the desired button text (for example, Popup Chart Builder)
 - Step 4 Add a Popup Binding by Clicking **Add Binding** button.
 - In the **ord** slot, paste the following value: `history:|view:history:HistoryChartBuilder`.
 - Step 5 Click **OK** and click the **Save** icon in the menu bar.
 - Step 6 In View mode, click the **Popup Chart Builder** button to access the History Chart Builder in the **Popup** window.

NOTE: You can control the size, position and title of the **Popup** window.

Step 7 When finished looking at the histories, close the **Popup** window, and the primary Px view is still open.


Applying visual styles to Px views

There are two ways to apply visual styles to a Px view: use Workbench and configure Px properties.


Using Workbench themes to apply styles globally means all Px files reference a single properties file, the NSS (Niagara Style Sheet).

Configuring Px view properties applies the visual style to the current Px view. You must define and configure these properties on every Px page:


- **Create Px Properties**

To open the **Add** window, click the **Add New**  button. In the **Add** window, name the property and choose property types from drop-down lists.

- **Edit Px Properties**

In the Properties palette **edit**  buttons to edit a selected existing property values.

- **Delete Px Properties**

Click the **Delete**  button to delete a selected property.

- **Link Px Properties**

With a Px object selected, right-click on the desired object property and select the **Link** menu item from the popup menu. Under the **Link** submenu, select the property item to apply.

NOTE: The popup menu is context sensitive. Px Properties that are not appropriate for the selected object property are not available (dimmed).

- **Unlink Px Properties**

With a Px object selected, right-click on the linked property and select the **Unlink** menu item from the popup menu.

Customizing the login screen

Instructions for customizing the login screen with your own logo and text.

Prerequisites:

- The logo image that you plan to use must be placed in the file system of the station
- Open connection to your station

Create a customized login screen

Step 1 Drag your logo image file from My File System to the Files folder of the Station.

Step 2 Expand **Config**→**Services**, right-click on the **WebService** node and select **Views**→**Slot Sheet**.

Step 3 Right-click under the current slots and select **Add Slot** from the popup menu.

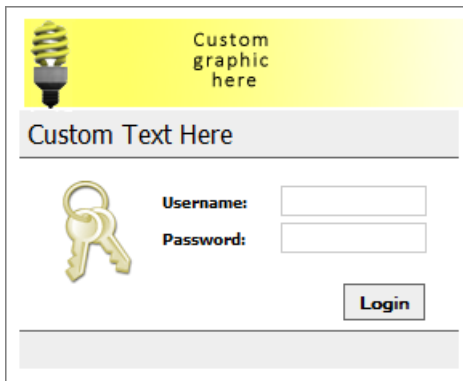
Step 4 Enter `logo` as the name and `baja Ord` as the type and click **OK**.

Step 5 Right-click on the **WebService** node and select **Views**→**Property Sheet**.

Step 6 Scroll down to the **logo** slot and click the down arrow next to the folder icon and select File Ord Chooser.

Step 7 Select the image that you want to use as the logo on the login screen and click **Save**.

- Step 8 Right-click on the **Config** node of the station and select **Views→Slot Sheet**.
- Step 9 Right-click under the slots and select **Add Slot** from the popup menu.
- Step 10 Enter `displayName` as the name and `baja Format` as the type and click **OK**.
- Step 11 Right-click on the **Config** node and select **Views→Property Sheet**.
- Step 12 In the **Display Name** field, enter the text that you want to display on the login screen, for example: `Custom Text Here`.
- For example, Custom Text Here
- Step 13 Click the **Save** icon in the toolbar.
- Step 14 Restart the station and log in through a browser window (`http://localhost/login`) to see your customized login screen:



Optimizing existing Px files for mobile devices

This procedure describes how to process a folder full of existing Px files (or a single existing Px file), applying responsive rules to the graphics to optimize their display when viewed on mobile devices using HxPx.

Prerequisites:

- You are connected to a running station with a folder containing one or more Px files to be optimized.
- Your station is running Niagara 4.6 or later.
- The pxEditor palette is open.

CAUTION: It is strongly recommended that you back up all Px files in a safe place before running the program object.

- Step 1 Click to expand the station **Services→ProgramService**.
- Step 2 In the pxEditor palette expand the Tools folder and drag the **ResponsiveMigration programObject** to the ProgramService in the station.
- Step 3 In a **Property Sheet** view ResponsiveMigration programObject, in the filePath field, click the Browse icon to locate the folder of Px files (or a single Px file) to be optimized and click **Open** to close the window.
- Step 4 In Services, click on DebugService to open the **Logger Configuration** window, and set the `com.tridium.px.editor.util` log to the preferred log level so that you can view the data, and click **Save**.
- Step 5 Click to disable/enable any of the processing features.
- If unsure, leave set to true.
- Step 6 In the Property Sheet (or Nav Tree) right-click on the ResponsiveMigration program, and click **Actions→Dry Run**

The programObject lists all the matching files in the target directory and describes exactly what changes to make.

NOTE: It is recommended that you execute Dry Run at least once. If you have done so previously, an alternative is to click on **Actions**→**Execute** to run the programObject a single time in this step, omitting the remaining steps.

Step 7 Click **Dry Run** to deselect it.

The Dry Run option is turned off.

Step 8 Right-click on the ResponsiveMigration program a second time, and click **Actions**→**Dry Run**.

Executing Dry Run a second time commits the changes to the file system.

The selected graphics have been processed to optimize their display when viewed on mobile devices using HxPx.

More configuration details are available in the [ResponsiveMigration-programObject, page 218](#) component topic.


Converting bound ORDs to tag-based NEQL ORDs

Improve the reusability of a Px view that contains bound ORDs by converting the slot path ORDs to tag-based NEQL ORDs.

Prerequisites:

- Workbench is installed and running.
- You have a component that has an existing Px view with bound ORDs assigned to it. The component on which a Px view is placed is referred to as the base component.
- You have more than one component configured with tags and relations.

Step 1 Open the **Px Editor** view for an existing Px view.

Step 2 In the **Bound Ords** pane, click  (Neqlize Ords) to convert the Slot Path Ords to tag-based NEQL Ords and open the **Neqlize Ords** window.

NOTE: When editing a Px file, the **Neqlize Ords** button is disabled as is the **Relativize Ords** button. Both of these buttons become enabled when editing the graphic on a component running in a station. The Neqlize Ords button will also be disabled if connected to an older station or a station that does not have the tagdictionary module installed (the automatic conversion cannot be done in that case and the single scheme will be missing which will prevent the Neqlize Ords from working).

In the example below, the highlighted row shows that the conversion process found the Numeric-Writable2 point has the `hs:equipRef` relation that traverses from the base component to that point, and the point has the `a:tag2` tag applied. It is this tag/relation combination that distinguishes that component from all the other components. The conversion process then pipes on the `single:scheme` to get the single point from the query result.

Step 3 Click **OK** to save the selected converted Ords and close the window.

After saving the converted Ords, the Px view can be reused on a different base component and the bound component can be named differently and be positioned anywhere in the station, as long as there is a set of tags to identify the component, and a relation to get from the base component to the bound component.

Example

Neqlize Ords (Convert to tag-based)

Options... Refresh All

| Before | After | Path |
|---|--|---------------------------------|
| <input type="checkbox"/> neql:tag1 single: | neql:tag1 single: | Failed to resolve Before Ord |
| <input type="checkbox"/> neql:tag3 single: | neql:tag3 single: | slot:/Points/BooleanWritable3 |
| <input type="checkbox"/> neql:traverse hs:equipRef<- where a:tag1 single: | neql:traverse hs:equipRef<- where a:tag1 single: | No identifying tags found |
| <input type="checkbox"/> neql:traverse hs:equipRef<- where a:tag3 single: | neql:traverse hs:equipRef<- where a:tag3 single: | slot:/External/NumericWritable3 |
| <input checked="" type="checkbox"/> slot:BooleanWritable2 | neql:tag2 single: | slot:/Points/BooleanWritable2 |
| <input checked="" type="checkbox"/> station:slot:/External/NumericWritable2 | neql:traverse hs:equipRef<- where a:tag2 single: | slot:/External/NumericWritable2 |
| <input type="checkbox"/> station:slot:/External/NumericWritable4 | station:slot:/External/NumericWritable4 | No identifying tags found |
| <input type="checkbox"/> station:slot:/External/NumericWritable5 | station:slot:/External/NumericWritable5 | Not an endpoint or descendent |
| <input type="checkbox"/> station:slot:/Points/BooleanWritable4 | station:slot:/Points/BooleanWritable4 | No identifying tags found |

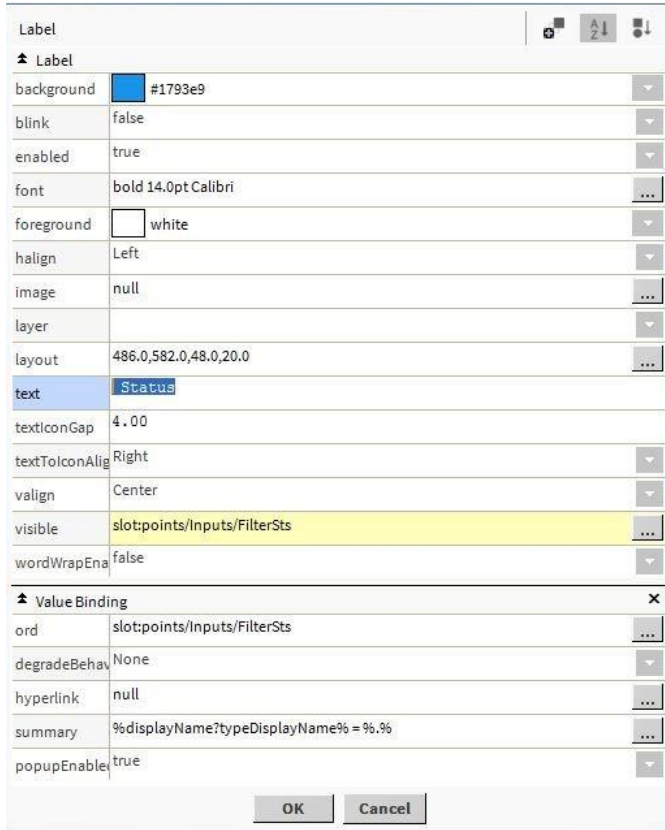
OK Cancel

Additional details on this example:

- The first four rows were previously converted to NEQL query ords. The Before Ords in the second and fourth rows still resolve. They are not selected (the After Ord is the same as the Before Ord) because they are already NEQL query Ords, but the Path column still shows the component the query resolves to.
- The first row does not resolve to any component because the tags have changed since that NEQL query Ord was converted. Note that Path column displays any errors encountered during conversion.
- The 3rd row no longer resolves to a single component only and more tagging is required to distinguish the component from other components.
- The fifth and sixth rows are slot paths that are able to be converted to NEQL query ords. The fifth row is a descendant only- there are no relations between the base and that bound component. The sixth row is for a component that is not a descendant of the base component but is the endpoint of a relation on the base component. Any rows where the Before Ord does not start with `neql` that can be successfully converted are automatically selected (indicated with a checkmark).
- The seventh through ninth rows are for slot path bound ords that failed to be converted to NEQL query ords. The seventh row is for a component that is the endpoint of a relation on the base component but does not have a set of tags that distinguishes it from other endpoints. The eighth row is for a component that is not a descendant of nor an endpoint of a relation on the base component.

Using the visible property in Hx profile (an example)

An example of binding and using the `visible` property is shown in the following illustration.

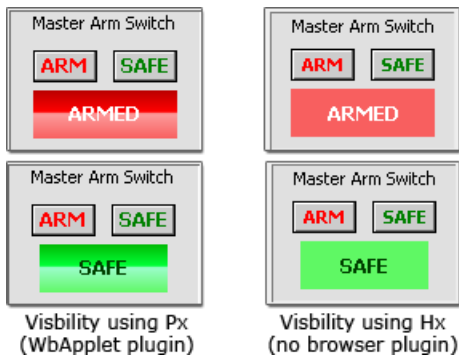


The example shows the `visible` property animated using a value binding. The value binding is linked to a writable point that passes its `true` or `false` value to the `visible` property so that it is visible (`true`) or hidden (`false`).

Note the following visibility characteristics:

- The `visible` property functions in Hx exactly as it functions in Px.
- Direct children of Tabbed panes cannot have a hidden visibility (`visible = false`).
- Direct children of Scroll panes cannot have a hidden visibility (`visible = false`).
- The `visible` property updates every 5 seconds.

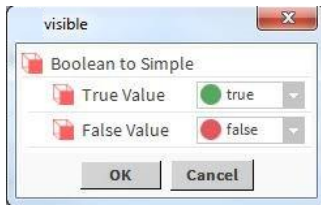
The following illustration gives a comparison of how a simple use of the `visible` property displays in a browser using the `WbApplet` and using Hx (no applet).



Note the following about this example:

- The `ARM` and `SAFE` buttons write to a Boolean point.

- The **ARMED** and **SAFE** labels have a `visible` property that is bound to and animated by the value of the Boolean point.
- Clicking the **ARMED** button sets the Boolean point value to `true`.
- Clicking the **SAFE** button sets the Boolean point value to `false`.
- The `true` Boolean point value sets the **ARMED** label `visible` property to `true` and the **SAFE** label `visible` property to `false` as defined by their respective `visible` properties using the visible window.
- The `false` Boolean point value sets the **ARMED** label `visible` property to `false` and the **SAFE** label `visible` property to `true` as defined by their respective `visible` properties using the visible window.
- The behavior of the `visible` property is the same in both Px (WbApplet) and Hx rendered views in the browser.



Chapter 6 Build navigation files

Topics covered in this chapter

- ◆ Creating a new nav files
- ◆ Deleting nav files
- ◆ Renaming nav files
- ◆ Editing nav files
- ◆ Create a global navigation menu using PxlInclude
- ◆ Embed Px files with ord variables using PxlInclude

The topics in this section are commonly associated with setting up and working with the Nav files.

Creating a new nav files

You can have more than one nav file associated with a station. Nav files must reside in the **File** node, not in the station database.

NOTE: An efficient method for managing nav files is to create a `nav` folder on the file system to hold all your nav files.

Step 1 In the Nav tree, right-click on the **File** node and select **New→NavFile.nav**.

The **File Name** window opens with a default file name

Step 2 In the **File Name** window, type a name for the new nav file and click **OK**.

The **File Name** window disappears and the new nav file opens in the Nav tree.

Deleting nav files

This procedure describes how to delete the nav file.

Step 1 In the Nav tree, expand the **Files** node and navigate to the nav file to delete.

Step 2 Right-click on the nav file and select **Delete** from the options list.

NOTE: An alternative method is to select the nav file in the Nav tree and press the **Delete** key on the keyboard.

The nav file is deleted.

Renaming nav files

This procedure describes how to rename the nav file.

Step 1 In the Nav tree, expand the **Files** node and navigate to the nav file to rename.

Step 2 Right-click on the nav file and select **Rename** option from the list.

The **Rename** window opens with a default file name.

Step 3 In the **Rename** window, type a new name for the nav file as desired and click **OK**.

The nav file is renamed.

Editing nav files

You can edit nav files in the Text Editor or in the Nav File Editor. The procedures in this section cover opening nav files, adding/deleting nodes, editing node hierarchy and editing nodes in a nav file.

Open nav files

The nav file resides under the **Files** node, not in the station database.

NOTE: Keeping all of your station nav files in a `nav` folder simplifies file management.

- Step 1 In the Nav tree under the **Files** node, find the nav file to open.
- Step 2 Right-click on the nav file and select **View→NavFileEditor** or double click on the nav file to open the **Nav File Editor** view.
- The **Nav File Editor** opens displaying the file in the **Result Tree** pane.
- Step 3 Edit the nav file, as desired.
- Step 4 Click the **Save** icon on the toolbar or select **File→Save** to save your changes.

Add nodes in nav files

This procedure describes how to add node in the nav file.

- Step 1 Open the desired nav file in the **Nav File Editor**.
- Step 2 Add a node to the **Result** pane using any of the following methods:
- **Copy and paste**
Copy a component or Px file from the Nav side bar and paste it into the **Result Tree** pane. The new node opens in the **Result Tree** as a child node of the node that you target when pasting.
 - **Drag (from the Nav side bar)**
Drag a component or Px file from the Nav side bar and drop it onto the **Result Tree** pane. A new node opens in the **Result Tree** as a child node of the node that you target when dropping it.
 - **Popup menu**
Right-click on the component to add and select Copy from the popup menu. Right-click the desired target area in the **Result Tree**, and select Paste from the popup menu. A new node opens in the **Result Tree** as a child node of the node that you target when pasting.
 - **New button**
Click the **New** button on the bottom of the **Nav File Editor** view. The **New Node** window opens.
Edit the **Display Name**, **Target Ord**, and **Icon** properties, and click **OK**. A new node opens in the **Result Tree** as a child node of the node that is selected.
 - **Drag (from the Source Tree)**
Toggle the **Show Components** button and the **Show Files** button to display the desired objects in the **Source Objects** pane.
Drag the component or Px file from the **Source Objects** pane and drop it onto the **Result Tree** pane at the desired location. A new node opens in the **Result Tree** as a child node of the node that you target.
- Step 3 Click the **Save** icon on the toolbar or select **File→Save** to save your changes.

Delete nodes in nav files

This procedure describes how to delete node in the nav file.

- Step 1 Open the desired nav file in the **Nav File Editor**.
- Step 2 In the **Result Tree** pane, locate the node to delete and do one of the following:
- Select the node to delete and click **Delete**

- Select the node to delete and press **Delete** on the keyboard
- Right-click on the node to delete and select Delete from the option.

Edit node hierarchy

Node hierarchy in a nav file is the position of parent and child nodes in a tree structure.

Step 1 Open the desired nav file in the **Nav File Editor**.

Step 2 In the Nav tree, locate the nav file to edit and do one of the following:

- Drag the tree node from one location to another in the tree.
- Select one or more child nodes in the tree and use the **Move Up** or **Move Down** buttons to raise or lower the node in its parent node.

NOTE: The **Move Up** or **Move Down** buttons move nodes only within their parent node. To move a node to a different parent, you must cut and paste the node.

The **Nav File Editor** opens the new node location in the **Result Tree** pane.

Step 3 Click the **Save** icon on the toolbar or select **File**→**Save** to save your changes.

Edit node properties

This procedure describes how to edit the node properties.

Step 1 Open the desired nav file in the **Nav File Editor**.

Step 2 In the Nav tree, locate the nav file to edit and do one of the following:

- Double-click on the desired node.
- Select the desired node and click the **Edit** button.
- Right-click on the node file and select **Edit** from the list.

Step 3 In the **Edit** window, edit the node **Display Name**, **Target Ord**, and **Icon** properties, as desired and click **OK**.

Step 4 Click the **Save** icon on the toolbar or select **File**→**Save** to save your changes.

Create a global navigation menu using PxInclude

Create a global navigation menu that you can use on multiple Px pages in a station to achieve a standardized, uniform look.

You can have a consistent navigation menu available on your home page or on all pages and you need only edit the navigation menu page any time a change is required. Since the navigation menu Px file is a PxInclude on the other Px views, they are automatically updated with any change that you make.

Step 1 Create a new Px view for the menu.

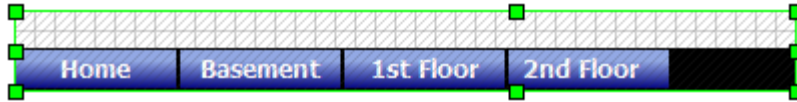
NOTE: Remove the ScrollPane at the root in this view otherwise scroll bars display on the menu once it is embedded in other Px views.

Step 2 Drag **Action Button** widgets from the **kitPx** palette to the canvas pane, one for each Px view that includes the menu.

Step 3 For each image button widget, add a value binding setting the **ord** property to the appropriate Px file. For example, `Home.px`, `Basement.px`, `1st_Floor.px`, `2nd_Floor.px`.

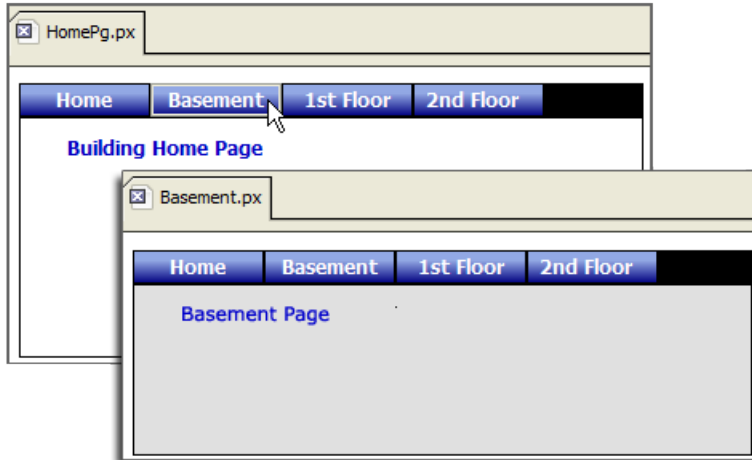
Step 4 Click the **Save** icon in the toolbar to save your changes.

This Px file is the global, navigation menu that you can include in other Px views.



Step 5 To embed the navigation menu, drag the **PxInclude** widget from the **bajauri** palette to each of the other Px pages on the station.

The result is a standardized menu displayed on each page, which you maintain by editing the navigation menu page any time a change is required.



Embed Px files with ord variables using PxInclude

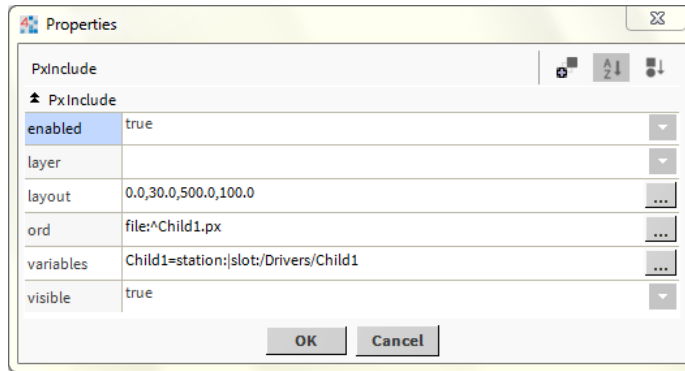
This example illustrates embedding Px files with ord variables using a PxInclude widget.

Prerequisites:

- Have two relativized Px views.

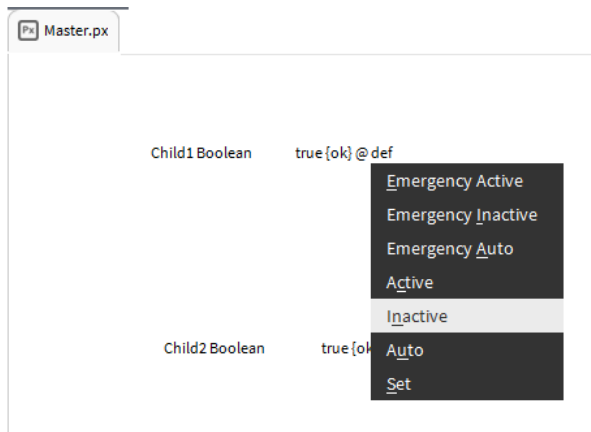
In this example, there are two relativized views, Child 1 and Child 2 to include in the master Px file.

- Step 1 Edit the relativized ords in your Px views to use the `$(variablename)` syntax in place of the actual ords, for example: `$(Child1)/Variable1`.
- Step 2 Save the Child 1 and Child 2 Px files.
- Step 3 Create another folder named, Master and add a Px view.
- Step 4 Drag the Child 1 folder onto the Master Px view.
- Step 5 In the **Make Widget** wizard, select the **Include Px File** option, and use the File ord Chooser to select the `Child1Px.px` file and click **OK**.
- Step 6 Select variable to bind, `Child 1`, and click **OK**. The PxInclude widget's properties.



Step 7 Repeat the steps 4–6 above using the Child 2 folder.

Step 8 Switch to View mode to see the embedded Px files with ord variables. Use the **Actions** menu to affect the changes:



Chapter 7 Generate reports

Topics covered in this chapter

- ◆ Reporting process workflow
- ◆ Set up the reporting service
- ◆ Set up report data in a ComponentGrid
- ◆ Creating a ReportPxFile
- ◆ Configuring ExportSource component
- ◆ Configuring the EmailRecipient

The Reporting function helps you to design, display, and deliver data to online views and to printed pages. The process of creating and sending a report can be performed in a variety ways. The following example illustrates the basic workflow for the reporting process.

Reporting process workflow

The following list is a basic workflow for the reporting process.

- **Setup the reporting service**
Add the ReportService and child components (ExportSource and EmailRecipient) from the **Report** palette **Services** container.
- **Layout report data in a ComponentGrid.**
Add a ComponentGrid component (from the **Report** palette) onto the desired location in your Nav tree. Using the **Grid Editor** view, drag points from the Nav tree or use the popup menu or main menu icons to add rows and columns to the ComponentGrid.
- **Create a Px page display of the report**
Using the Px editor, add the ComponentGrid to a ReportPxFile page and design the report layout using ReportWidgets available from the **Report** palette.
- **Configure ExportSource component**
In the ExportSource property sheet, identify the report Px file and schedule the report delivery.
- **Configure EmailRecipient component**
In the EmailRecipient property sheet, identify the desired recipients and email account to use for delivery of the report.

Set up the reporting service

Add the reporting service and components to the station.

Step 1 In the Palette side bar, open the **report** palette.

Step 2 Expand the `Reporting` folder.

Step 3 Drag the following components **ReportService** and child components (**ExportSource** and **EmailRecipient**) into your station **Services** node.

ReportingService is setup and ready to use.

Set up report data in a ComponentGrid

Add a **ComponentGrid** component (from the `Reporting` folder) onto the desired location in your Nav tree. Using the **Grid Editor** view, drag points from the Nav tree or use the popup menu or main menu icons to add rows and columns to the **ComponentGrid**.

- Step 1 In the Palette side bar, open the **report** palette.
- Step 2 Expand the `Reporting` folder from the **report** palette and drag the **ComponentGrid** component into the **Config** node in your station.
- Step 3 Right-click on the new **ComponentGrid** component and select **Views→Grid Editor**.
The **Grid Editor** view opens.
- Step 4 Expand the Nav tree to locate the desired network and expand the network so that you can see the devices in the tree.
- Step 5 Click and drag a single device from the Nav tree onto the middle pane in the **Grid Editor**.
NOTE: You must define this template row before adding columns.
This also adds the same device to the lower pane as one of the defined rows.
- Step 6 In the Nav tree, click and drag the same device to the top pane of the **Grid Editor** (no columns defined).
This adds a column definition which defaults to the `displayName` of the device.
- Step 7 Again, click and drag the same device in the Nav tree to the top pane of the **Grid Editor**.
This adds a second column definition.
- Step 8 Double-click the second column definition in the top pane of the **Grid Editor** to open the **Edit** window. Change the **Name** property to `Status` and change the **format** property to `%status%`, and click the **OK** button.
- Step 9 In the Nav tree, select all of the other devices that you wish to include in the table, then click and drag them onto the lower pane of the **Grid Editor**.
This adds all of those devices as separate rows of the grid table.
- Step 10 Save the **Grid Editor** and change to the **Grid Table** view to see the completed table.

Creating a ReportPxFile

This procedure describes how to create a ReportPxFile.

- Step 1 Right-click on a **Files** container for a Px file and select **New→ReportPxFile.px**. A **Name for New File** window opens.
- Step 2 In the **Name for New File** window, edit the default file name, if desired, and click **OK**.
The new Px file, pre-loaded with a **ReportPane** widget, appears in the Nav tree.
- Step 3 Double-click on the new **ReportPxFile** to open it in PxViewer.
- Step 4 Switch to Edit mode.
- Step 5 Open the **report** palette, expand the `Reporting` folder and drag a **ComponentGrid** widget to the canvas pane.
- Step 6 In the **Make Widget** wizard, click **Workbench View** and in the secondary view area select **Grid Editor** and click **OK**.
- Step 7 Design the report layout using **ReportWidgets**.

Configuring ExportSource component

This procedure describes how to configure ExportSource component.

- Step 1 In the station, expand **Config→Services→ReportService** and double-click on the **ExportSource** component to open the **AX Property Sheet** view.
- Step 2 In the **Source** slot, click on the icon to launch the ReportPxFile using the **Export Source wizard**.
- Step 3 In the **Export Source** wizard, click the folder icon and browse to your **ReportPxFile** and select it click **Open** to close the window.
- Step 4 In the **Export Source** wizard, click **Next** to continue and **Finish**.
- Step 5 In the Property Sheet view, expand the **Schedule** slot to schedule the report delivery. Select from the following options:
 - Trigger Mode (Manual, Daily, or Interval)
 - Time Of Day
 - Randomization
 - Days of Week
 - Last Trigger
- Step 6 Click **Save**.

You have configured the report source and delivery schedule.

Configuring the EmailRecipient

Two **EmailRecipient** components send email from the system. One is located in the **email** palette. This component manages alarms that are configured to be sent via email. The second is located in the **report** palette. It manages the sending of reports to one or more specific email addresses.

Prerequisites: The **EmailService** and **ReportService** are available in the station's **Config→Services** folder.

- Step 1 Right-click the **EmailRecipient** under **ReportServices** or **EmailServices** nodes and click **Views→Property Sheet**.
- Step 2 Enter the name and email address of recipient(s) and click **Save**.
- Step 3 Do the same for the alarm **EmailRecipients**.

Chapter 8 About profiles

Topics covered in this chapter

- ◆ About Web Profiles
- ◆ About Kiosk Profiles

Profiles provide Niagara software engineers with the ability to customize both the desktop Workbench and the Web Workbench interface.

Profiles control how Workbench appears while using either the Java plugin, Hx, or mobile technology. In this context, it does not refer directly to security settings or personal preferences.

The type of profile used to log in to the station can result in certain views being inaccessible even if the user has permissions to access the view.

NOTE: If you include a view-based widget as part of a Px page, then, from a user-profile perspective, it is just a Px page which is allowed. The user permissions to that included view (component) in the station determine if it displays in the Px page or not.

The software engineer can create customized Workbench applications that provide different functionality.

Workbench comes with the following two categories of profiles that have specific profile types available. These profiles are described in the following sections.

- Web Profiles
- Kiosk Profiles

About Web Profiles

Custom user interface design, at the browser level, allows for different views that may include or exclude features such as sidebars, navigation trees and other tools that are provided through the interface. Web profiles identify these different web interfaces.

You can assign a default Web profile to each user that is listed in the User Manager. In addition, if you have Mobile licensed, you can assign a Default Mobile Web Profile as well.

Following are some of the standard Web profile options:

- Default Wb Web Profile includes all Workbench functions
- Simple Admin Wb Web Profile includes a subset of Workbench interface features, but with admin access available.
- Basic Wb Web Profile includes a subset of Workbench interface features.
- Mobile Profile (default) provides a way for mobile devices (cell phones and tablets) to effectively display station views.
- Default Hx Profile provides full Hx interface features with no Java plugin required.
- Basic Hx Profile provides a subset of interface features with no Java plugin required.
- Default Touch Profile uses Hx technology to provide a real-time user interface without the Java plugin download.
- Basic Touch Profile uses Hx technology to provide a real-time user interface without the Java plugin download.
- Handheld Touch Profile uses Hx technology to provide a set of features that are optimized for viewing on a handheld touchscreen device.

Velocity Doc Web Profile

This profile supports VelocityDoc components. If the component has its `useProfile` property set to `true`, the system uses this profile to generate an outer HTML page while the VelocityDoc generates the inner HTML content.

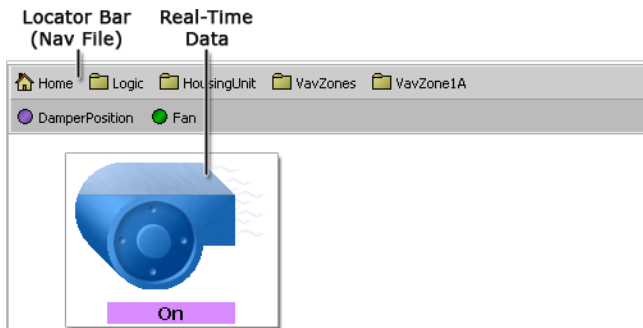
Apache Velocity is an open source web template engine that is integrated into the Niagara Framework. It provides users with the ability to script an HTML page together using a very simple scripting language.

Developers and advanced users can take advantage of the Velocity API through station components that one can configure from the `axvelocity` palette.

Basic Hx Profile

This profile uses Hx technology. It provides a reduced set of features but include real-time user interface without the Java plugin download.

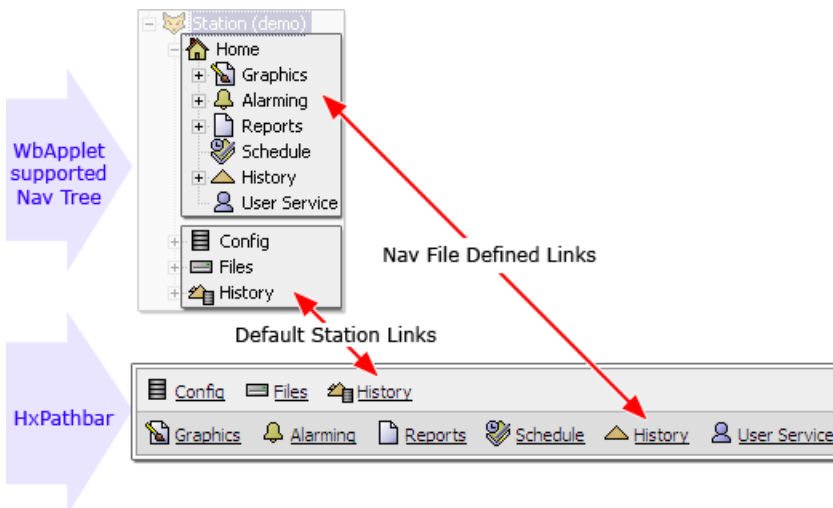
Figure 31 Basic Hx Profile example



Major features of the Basic Hx Profile include:

- A locator bar appears across the top of the view area and displays links defined by the nav file.
- HxPathbar provides a default set of links between the three root spaces: Station (Config), Files, and History. Users can fully navigate a station in Hx without having to manually type in URLs.

Figure 32 Comparing the HxPathbar and the WbApplet Nav Tree



- Real-time data appear without using Hx technology instead of the Java plugin.

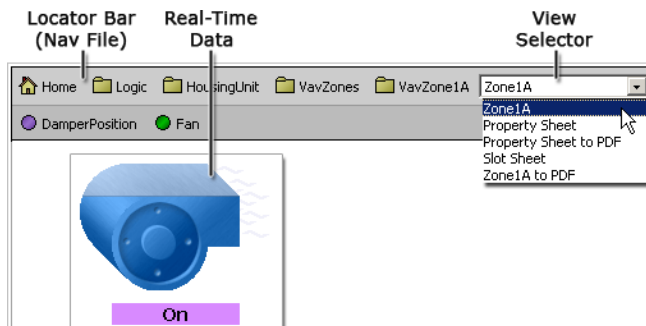
- The **visible** property, which displays and behaves the same in Hx profiles and WbWeb profiles (using the WbApplet). For example, you may bind and control a **visible** property in some widgets using true/false values to show or hide a widget.

Default Hx Profile

This profile uses Hx technology to provide a real-time user interface without the Java plugin download.

Here is an example of the Default Hx Web Profile:

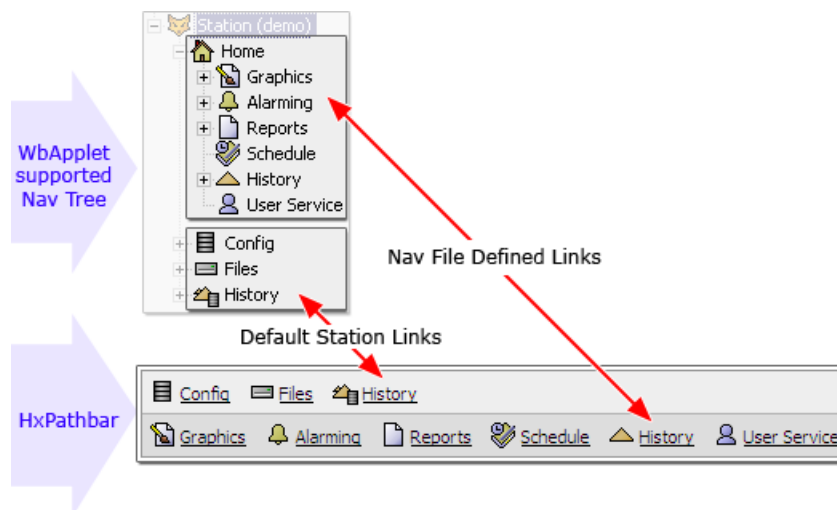
Figure 33 Default Hx Profile example



Major features of the Default Hx Profile include:

- An interactive Path Bar (locator bar (just below the toolbar) that contains the path or ord for the current view, and functions as a browser address permitting a user to enter an ord or a URL.)
- A HxPathbar that provides a default set of links between the three root spaces: Station (Config), Files, and History. Users can fully navigate a station in Hx without having to manually type in URLs.

Figure 34 Comparing the HxPathbar and the WbApplet Nav Tree



- Real-time data displayed using Hx technology instead of the Java plugin.
- View Selector drop-down list that provides a set of alternative views for the active object.
- The **visible** property, which displays and behaves the same in Hx profiles and WbWeb profiles (using the WbApplet). For example, you may bind and control the **visible** property in some widgets with true and false values set to show or hide a widget.

Handheld Hx Profile

This is an Hx profile without any profile chrome.

Use this profile to limit user navigation to what is provided in Px pages.

HTML5 Hx Profile

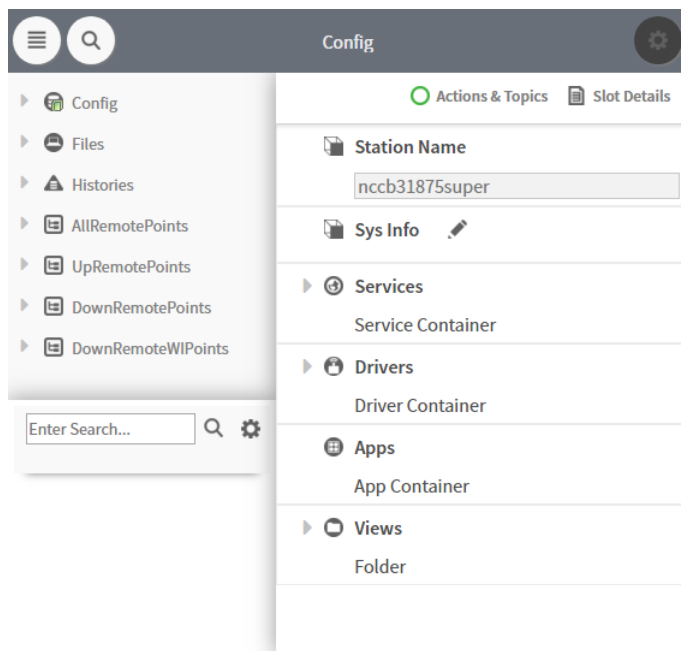
In Niagara 4.6 and later, the HTML5 Hx Profile defines the Default Web Profile and the default Mobile Web Profile.

IMPORTANT:

In Niagara 4.6 and later, legacy Niagara Mobile technology is deprecated. Although it continues to be supported and works as expected, you should not use it for any new projects. Instead, use the mobile-friendly features of the HTML5 Hx Profile which provides a significantly improved mobile user experience.

The HTML5 Hx Profile features a responsive layout that automatically scales views to display accurately on a mobile device, as well as on a PC using either Workbench or a browser. When logging in to a mobile device, the profile responsively adjusts the layout to a more simplified, mobile-friendly user experience. The station display is more visually streamlined, making best use of the limited space on mobile devices. These changes make it easier for operators to use.

Figure 35 Station connection in HTML5 Hx Profile (shows expanded Nav and Search sidebars)

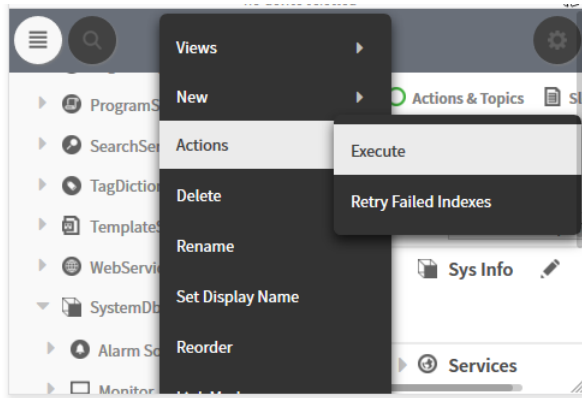


For example, on logging in to a station, the Nav tree lists standard station folders (Config, Files, Histories, etc.) and any hierarchies at the root of the station. This eliminates having to navigate to the station and expand folders.

Following are some examples of the responsive layout.

- The side bar is repositioned at the top of the view.
- The palette button is hidden.
- The actions command button becomes visible, enabling users to invoke Widget commands, select an alternate view, log off, and navigate to home.

Figure 36 Rotated (landscape) view shows right-click menu with actions



There are many options available for the HTML5 Hx Profile in a user's web profile. These options are respected. For example, on a mobile device, if a user cannot select a view based on role and permissions, the profile hides the command rather than show it in a dimmed state.

Similarly, the profile supports many user selections even in a view adjusted for the mobile device. For example, if the user selects the palette, it remains visible as the browser window is made smaller.

The mobile-friendly features of the HTML5 Hx Profile provide a consistent user experience that supports the functionality available in a mobile device (cellphone or tablet). The profile features a responsive layout that is capable of scaling views for display on a mobile device, including touch-screen features, such as pinch-zoom functionality and touch-activated command buttons.

Existing HxPx views display well in the small screen environments. This includes truly mobile-friendly **Alarm Console**, **Scheduler**, **Web Chart**, **Property Sheet**, and manager views that feature compact editors for configuration purposes.

Apple devices and operating systems

| Apple iPhone type | Operating system |
|-------------------|------------------|
| iPhone 6s | iOS 11 |
| iPhone 6 Plus | iOS 11 |
| iPhone 7 | iOS 11 |
| iPhone 7 Plus | iOS 11 |
| iPhone 8 | iOS 11 |
| iPhone 8 Plus | iOS 11 |
| iPhone X | iOS 11 |

Android devices and operating systems

| Android Mobile Device | Operating System |
|-----------------------|---------------------------------------|
| Samsung Galaxy S8 | 7.0 Nougat, 8.0 Oreo |
| Samsung Galaxy S7 | 6.0 Marshmallow, 7.0 Nougat, 8.0 Oreo |
| Google Pixel | 7.0 Nougat, 8.0 Oreo |
| Google Nexus (TBD) | (TBD) |

Tablet devices and operating systems

| Tablet device | Operating system |
|-----------------------|--|
| Apple iPad | iOS 11 |
| Samsung Galaxy Tablet | 5.1 Lollipop 6.0 Marshmallow, 7.0 Nougat, 8.0 Oreo |
| Amazon Fire | 5.1 Lollipop 6.0 Marshmallow, 7.0 Nougat, 8.0 Oreo |

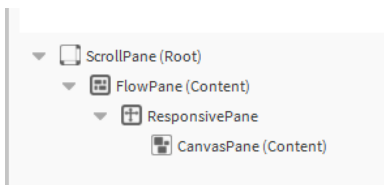
NOTE: The Chrome web browser does not cache any web resources when connected over https to a station that has a self-signed certificate or a certificate that is not trusted. This can hurt the performance of various web-based pages served up from Niagara, such as the HTML5 Hx Profile and Hx views. For best results, install and maintain a fully-signed certificate for an optimal web viewing experience.

Create an HTML5 graphic

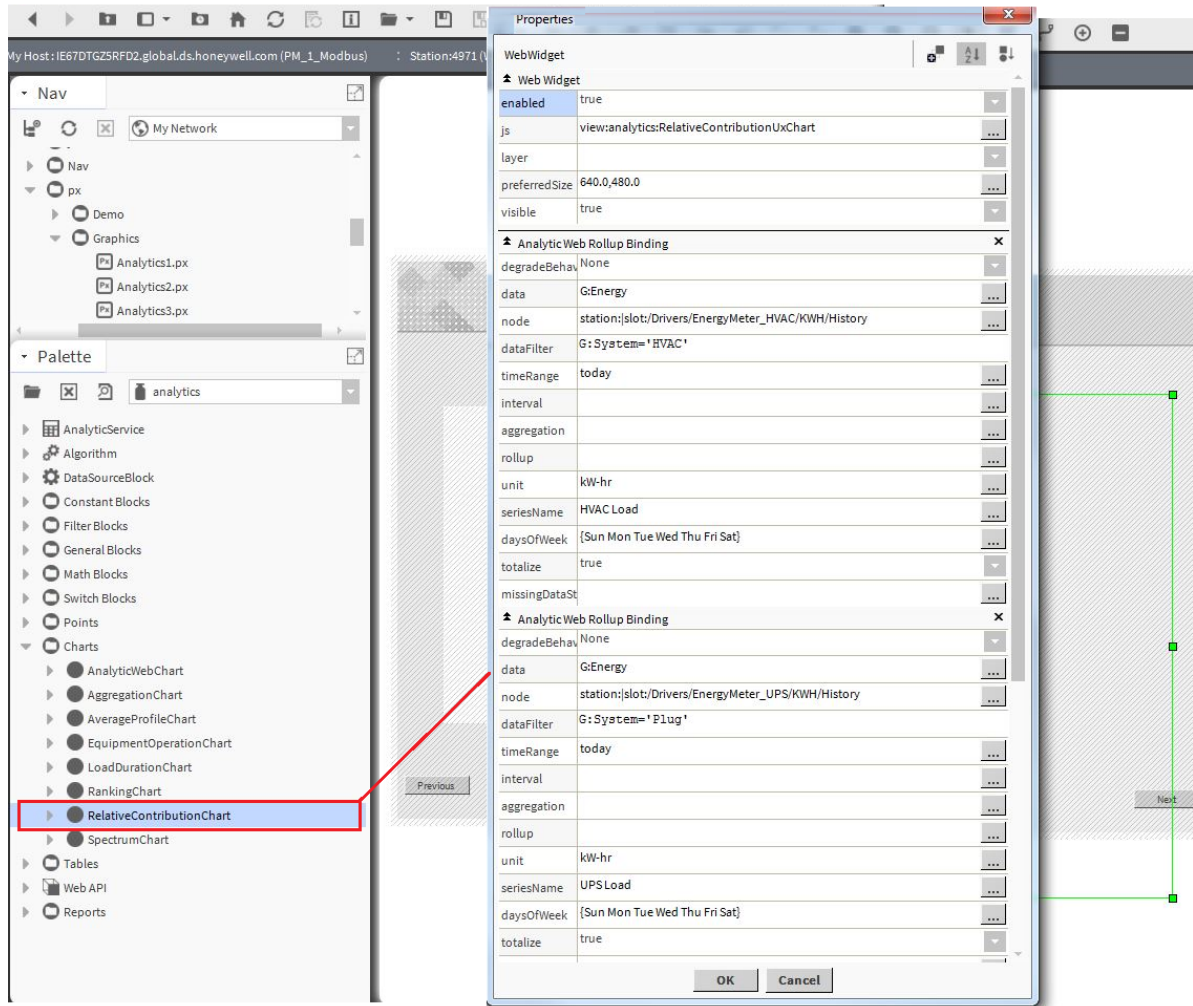
This topic documents how to create an HTML5 graphic in Workbench.

Prerequisites: A Px file is available in the **File** container.

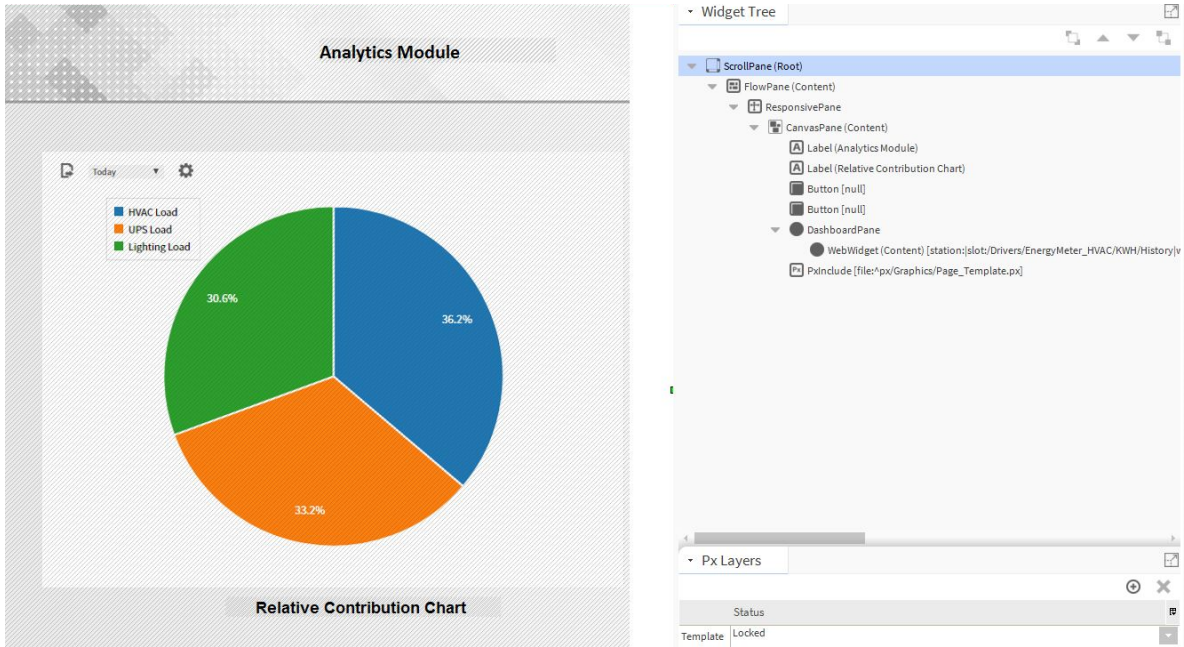
- Step 1 In the **Palette** sidebar, open **bajauri** palette.
- Step 2 Drag **Flowpane** from the **Panes** folder onto the **ScrollPane**.
- Step 3 Drag **ResponsivePane** and **CanvasPane** from the **Panes** folder onto the **FlowPane**.



- Step 4 In the **Palette** sidebar, open **analytics** palette.
- Step 5 Open **Charts** folder and drag **RelativeContributionsChart** widget onto the Px page.



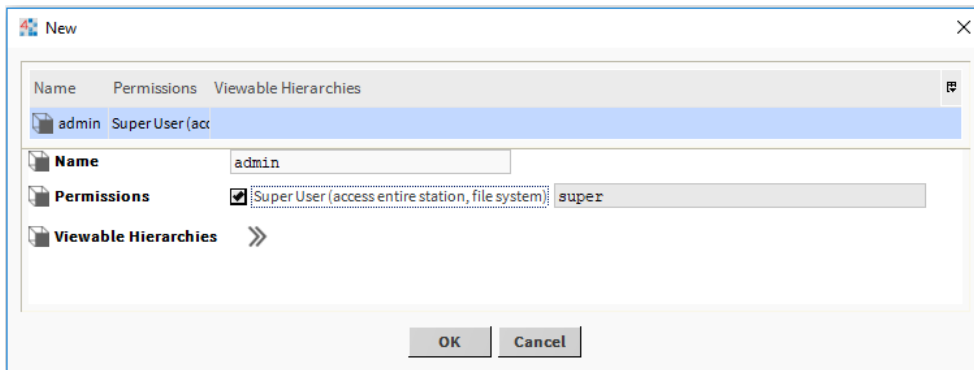
Step 6 Edit the properties of RelativeContributionsChart widget and click **OK**.
New widget displays on the Px Editor canvas.



Add an Admin Role

This procedure documents how to add admin role to assign ability to view 'admin' level views. This prevents users from accessing some views even if they have a bookmark to them.

- Step 1 In the Nav tree, expand the **Config**→**Services**→**RoleService** and double-click a **RoleService** component.
The **Role Manager** view opens.
- Step 2 Click a **New** button at the bottom of the view.
A first **New** window opens.
- Step 3 In **Type to Add** property, select the **Role**.
- Step 4 Enter the number of roles to add (default value is 1), and click **OK**.
The second **New** window opens.



- Step 5 In **Name** property, select the **admin**.
- Step 6 Set **Permission** property to **SuperUser** to add admin level role and click **OK**.
The admin role is added in the system.

Create user account

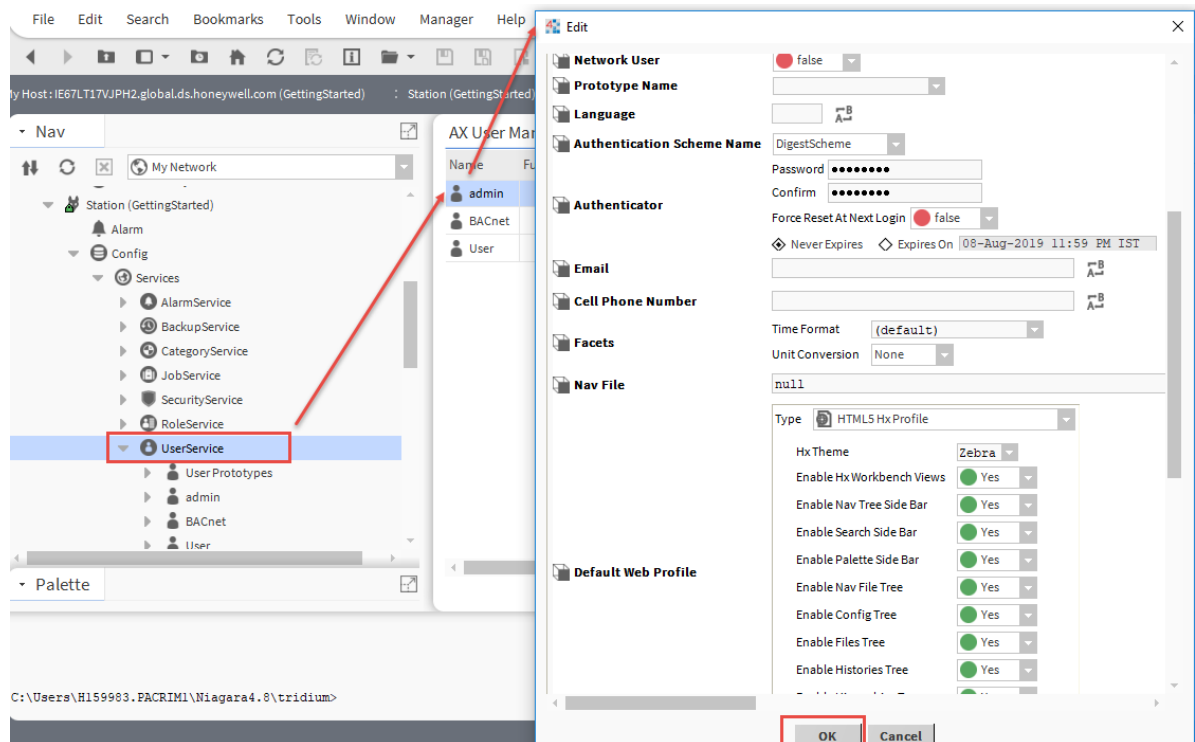
After creating admin role, create user account to provide default web profile (HTML5 Hx Profile) access.

Step 1 Go to **Services**→**UserService** and double click on the **UserService**.

The **AX User Manager** view displays.

Step 2 Select and double click on the admin role from the list.

The **Edit** window opens.



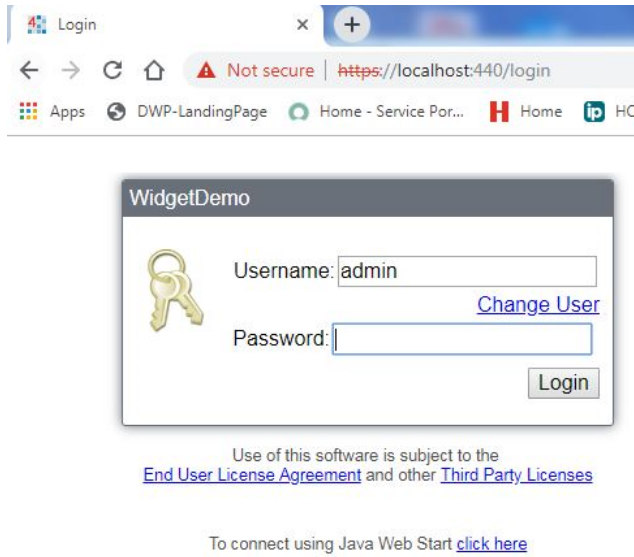
Step 3 Set and confirm the password.

Step 4 Provide the nav file path of the widget in the Nav File.

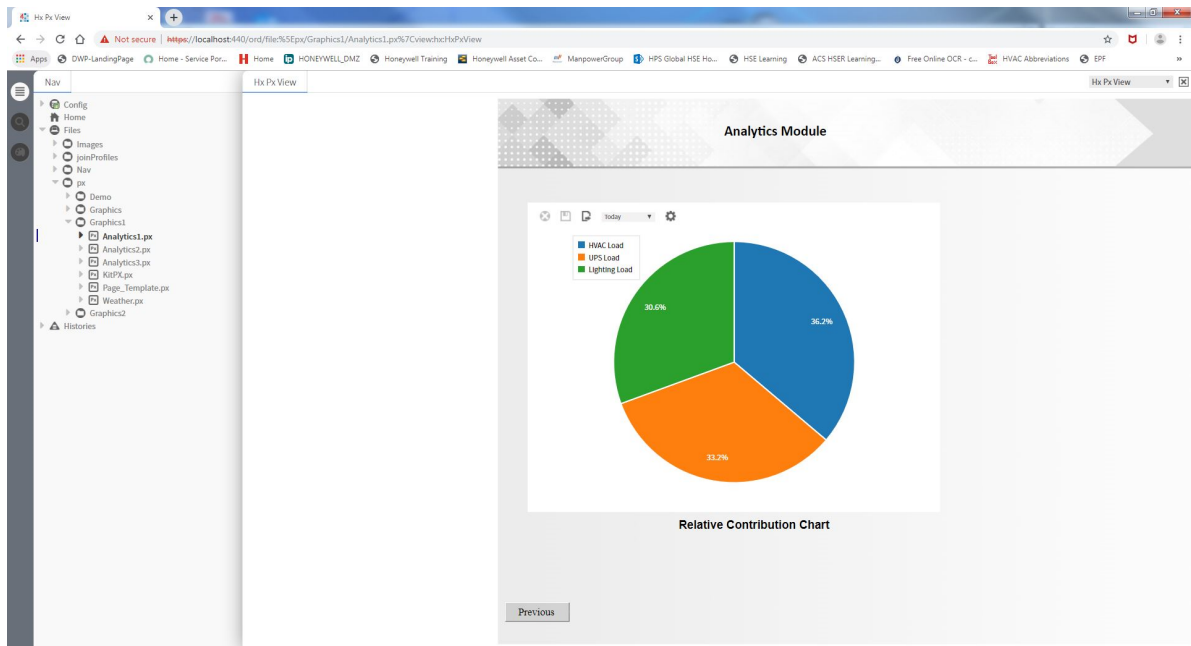
Step 5 Set the HTML5 HX profile and click **OK**.

Step 6 Open the browser and type `https://localhost` in the address bar.

This opens the login window of the station in the browser.



Step 7 Enter Username and Password.
The HTML5 graphic displays in the browser.



Default Mobile Web Profile

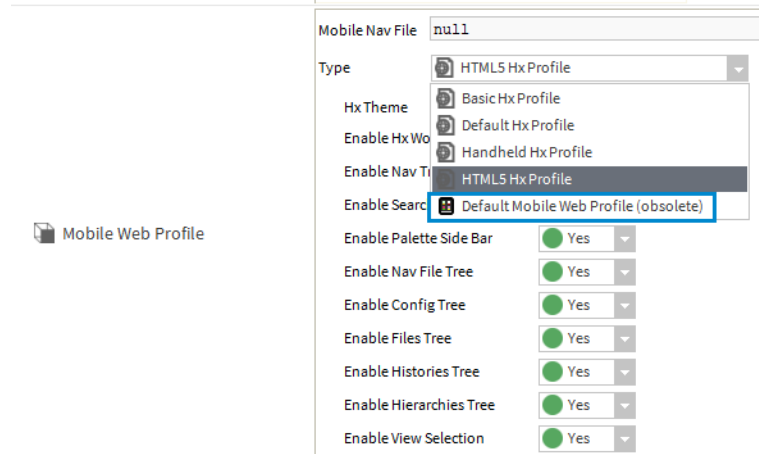
In Niagara 4.6 and later, this profile is deprecated.

IMPORTANT:

In Niagara 4.6 and later, legacy Niagara Mobile technology is deprecated. Although it continues to be supported and works as expected, you should not use it for any new projects. Instead, use the mobile-friendly features of the HTML5 Hx Profile which provides a significantly improved mobile user experience.

NOTE: Also in Niagara 4.6 and later, you can view HxPx pages from a mobile profile by setting the **Px Target Media** to **HxPx**. This feature is on by default. However, if you prefer to continue viewing only mobile pages you can turn this feature off by going to your MobilePxApp and setting the **showHxPx** property to **false**.

Figure 37 User properties shows Default Mobile Web Profile as obsolete



In NiagaraAX releases (AX-3.7–AX-3.8), Mobile apps use Bajascript technology and require no Java-plugin download. The Default Mobile Web Profile provides a reduced set of features (compared to the Java-plugin profiles) but includes most of the frequently-used controls and field editors in a way that is easy to use on a cell phone or tablet.

Figure 38 Default Mobile Profile is supported for NiagaraAX releases (AX-3.7–AX-3.8)



Major features of the Default Mobile Profile include:

- Header navigation

The **Header Bar** appears across the top of the view area and displays a **Back** button and a **Command** button for navigation. This bar can be disabled (hidden) if desired.

- Real-Time Data

Data is displayed in real time using Bajascript technology and the NiagaraAX **BOX service** instead of the Java plugin.

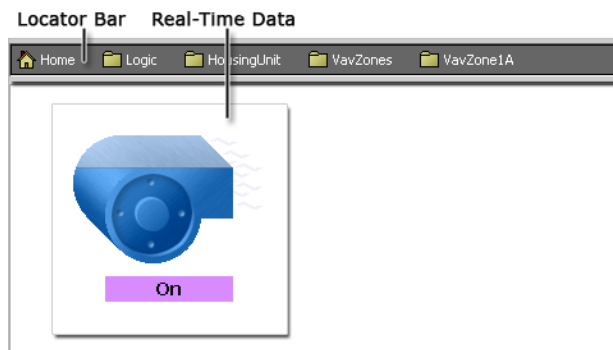
- View Selector

A limited set of view selections are available through the **Select Views** menu under the **Command** button.

Basic Wb Web Profile

This profile uses a reduced set of features but provides a rich user interface using the Java plugin download. Here is an example of the Basic Wb Web Profile:

Figure 39 Basic Wb Web Profile example



Major features of the Basic Wb Web Profile include:

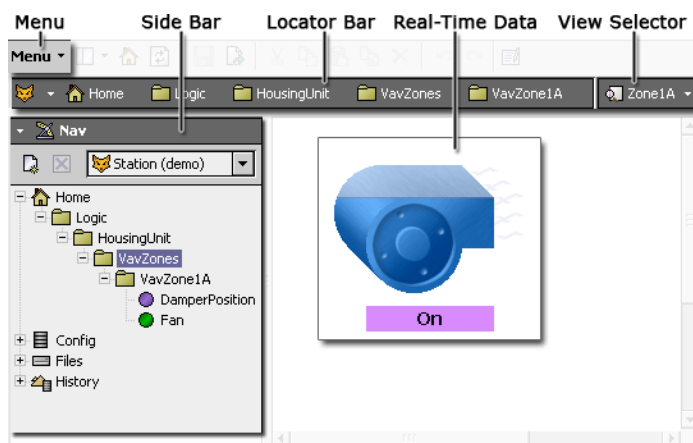
- A Locator Bar appears across the top of the view area.
- Data appear in real-time using the Java plugin.

Default Wb Web Profile

This profile provides all the features of the Web Workbench, using the full Java plugin download.

Here is an example of the Default Wb Web Profile:

Figure 40 Default Wb Web Profile example



Major features of the Default Wb Web Profile include:

- A drop-down list includes these submenus: **File**, **Edit**, **Search**, **Tools**, **SideBars**, and **PxEditor**. These menus provide commands that are explained in Types of menu bar items, in *Getting Started with Niagara*.
- A side bar, which you may show or hide. When used it may include the **Nav** side bar and the **palette** side bar. The Nav side bar displays the navigation hierarchy that is defined by the nav file.
- A path bar/locator bar just below the toolbar contains the path or ord for the current view and functions as a browser address. Users may enter an ord or a URL.
- Real-time data provided by the Java plugin.
- a view selector provides the same view selection options that are available in the desktop Workbench views.

Handheld Wb Web Profile

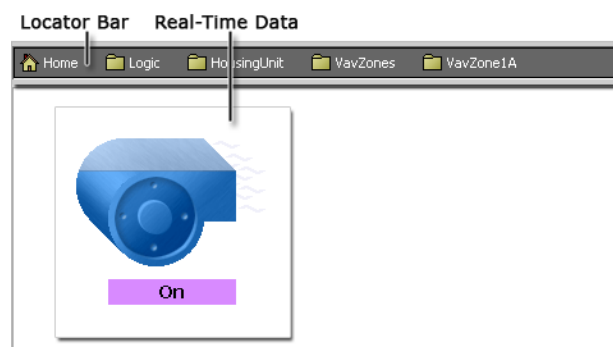
This profile removes all extra interface background information, displaying only the current view. This profile provides the most bare bones web experience possible.

Simple Admin Wb Web Profile

This profile provides all the access of the Default Wb Web Profile without using the full Java plugin download. It provides the same user interface as the Basic Wb Web profile but allows the full admin access as the Default Wb Web profile.

Here is an example of Simple Admin Wb Web Profile:

Figure 41 Simple Admin Wb Web Profile example



Major features of the Simple Wb Web Profile include:

- Access to all views (as allowed by user permissions).
- A locator bar across the top of the view area that is not restricted to the nav file.
- Real-time data provided by the Java plugin.

About Kiosk Profiles

Kiosk mode runs a station so that it fulfills many of the needs of a stand-alone operator workstation as well as a touchscreen interface.

NOTE: Currently, the KioskService and the Kiosk Profiles are not supported in Niagara 4. Although, it is possible that this will change in future releases.

To automatically start Kiosk mode, you must have a locally connected display.

NOTE:

The Kiosk profile is not invoked or displayed by remote browser clients. You cannot use a remote computer with a touchscreen and expect to get the same Kiosk interface.

While Kiosk mode is not limited to touchscreen applications, it does provide advantages that make it well suited for use in a touchscreen application.

Basic Kiosk Profile

This profile supports Workbench in Kiosk mode.

NOTE: Currently, the KioskService and the Kiosk Profiles are not supported in Niagara 4. Although, it is possible that this will change in future releases.

The Basic Kiosk Profile displays a read-only locator bar as a bread crumb trail and provides **Back**, **Forward**, and **Logoff** buttons. This profile disables:

- all side bars
- the entire menu
- the entire toolbar
- all non-admin views

Default Kiosk Profile

This is the default Workbench profile used in Kiosk mode.

NOTE: Currently, the KioskService and the Kiosk Profiles are not supported in Niagara 4. Although, it is possible that this will change in future releases.

Default Kiosk Profile supports all the features of a normal desktop Workbench profile except for:

- **File > Close command**
- **File > Exit command**
- **Tools > Credentials Manager**

Handheld Kiosk Profile

This profile provides a reduced display that is designed to help handheld users view and navigate the application.

NOTE: Currently, the KioskService and the Kiosk Profiles are not supported in Niagara 4. Although, it is possible that this will change in future releases.

Chapter 9 Px graphics reference

Topics covered in this chapter

- ◆ Px view component property sheet
- ◆ New Px View window
- ◆ About Px Editor
- ◆ About SVG support
- ◆ About the View Source XML Window
- ◆ Widget layout
- ◆ About the PxInclude Widget
- ◆ About Nav file elements
- ◆ About the ReportPxFile
- ◆ About the Grid Table view
- ◆ About the Grid Label Pane view
- ◆ About the Grid Editor view
- ◆ About Mobile Px App

This section provides additional information about Px graphics features.

The additional reference information focuses on these functions:

- Px views viewer and the Px Editor
- SVG support
- Source xml view
- Widget layout, palettes and properties
- Types of data bindings
- Types of reports
- Grid tables

Px view component property sheet

A Px view is a custom graphical view of control logic that you define in a Px file. The view provides a visualization of information in a rich, dynamic format that is easy to create and to edit.

Using the Px Editor to build Px views, you can create the desired visualization of your control logic without having software programming skills. When attached to a component, the Px view becomes the default view for the component.

Looking at an object's Px view in the property sheet (where it may be edited) helps illustrate what a Px view is. A Px view is comprised of the following parts.

- **Icon**

Assign or change the icon file that appears to the left of the Px view display name. After assigning this icon to the view, it appears in the view selector menu, the Nav tree side bar, and in the property sheet view.

Icon graphics must be linked into your view from a module (for example, use the following path to find a folder icon: `module://icons/x16/folder.png`). Icon graphics are not supported outside of modules.

- **Px File**

Assign the Px file that is associated with active Px view.

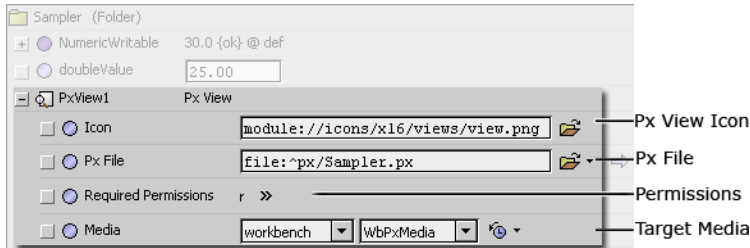
- **Required Permissions**

Assign or edit the Operator and Admin permissions that are applicable when the view is active: Read (r), Write (w), Invoke (I).

- **Media**

Edit the target media of the Px view: `workbench` or `html`.

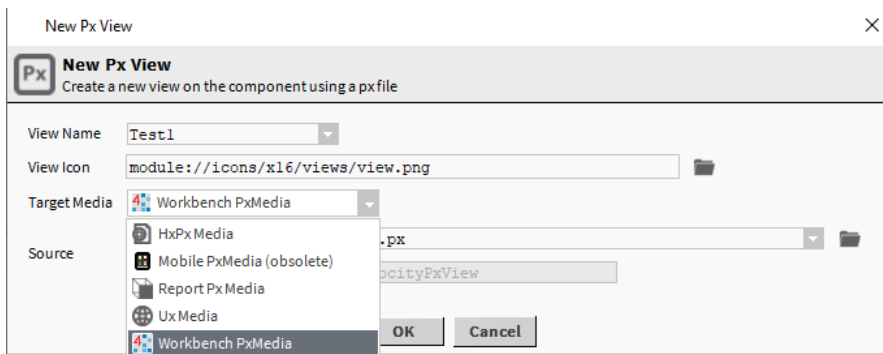
Figure 42 Px view in a component property sheet



New Px View window

The New Px View wizard configures the information needed for Workbench to create a Px view.

Figure 43 New Px View



| Property | Value | Description |
|----------------------|---|---|
| View Name | text | The name that appears at the top of the view and serves as the file name in the station. |
| View Icon | ORD | Identifies an icon graphic to link to the view. |
| Target Media | drop-down list (defaults to <code>Workbench PxMedia</code>) | Indicates the media type for the Px view. |
| Source: Px File | ORD | Identifies the location of the file for the Px view. This is an xml file with a <code>.px</code> extension. |
| Source: Dynamic View | drop-down list (defaults to <code>axvelocity: VelocityPxView</code>) | Selects a specific, pre-defined view. |

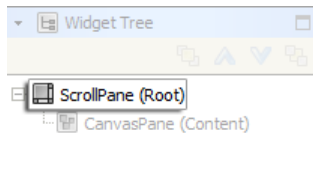
Target Media

In addition to the default, `Workbench PxMedia` type, you can select one of the following when creating a new view on a component:

- **HxPx Media**

In Niagara 4.6 and later, choosing the HxPx Media type takes full advantage of the HTML5 functionality which provides a significantly improved mobile user experience. This includes responsive Px improvements, bajaux widgets, updated Alarm Console, Scheduler view, Manager views, Property Sheet field editors, etc. Choosing this media type provides you with a Px file that has a ScrollPane with a CanvasPane. With this property value set, the Px Editor can warn you, as you design your Px file, if you choose a widget that is not supported by Hx media.

Figure 44 HxPx Media creates Px page with Scroll and Canvas Panes



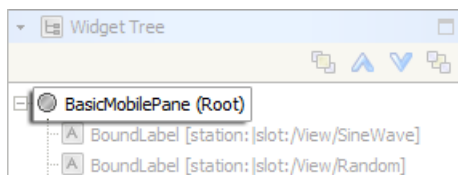
In addition to `Workbench PxMedia`, a suite of technology called Hx is available. The Hx framework is comprised of a set of server side servlets and a client side JavaScript library. Hx allows a real-time user interface to be built without use of the Java Plugin. It requires only web standards: HTML5/HTML, CSS, and JavaScript. If you are developing for a target media that is limited to browsers with HTML and CSS with no applet plugins available, be sure to test your Px views often in the appropriate browser as you proceed in your development. The goal of Hx is to satisfactorily present your Px views in non-applet browsers, however, due to the difference in display technologies, you should verify that the Px view displays in the browser as you expect it to.

- **Mobile PxMedia**

Deprecated in Niagara 4.6 and later. Although, this option is marked `Obsolete` you can still select it and use it. However, this media type provides none of the HTML5 functionality available in the `HxPxMedia` type.

If you choose the `Mobile PxMedia` you must use a `BasicMobilePane` at the root of your Px page so it will display correctly on a mobile device. Assigning a `MobilePxMedia` type when creating a new Px page, causes the new page to come with a `BasicMobilePane` at the root of the Px file, as shown in the following figure.

Figure 45 Mobile PxMedia creates Px page with BasicMobile Pane

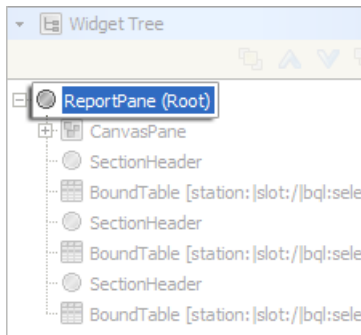


NOTE: Setting the `CanvasPane scaleMode` property to `Fit Ratio` causes your Px page to automatically scale to fit various device display sizes.

- **Report PxMedia**

Assigning this media type when creating a new Px page, creates a new Px file with a `Report Pane` containing a timestamp and page numbering at the root of the Px file.

Figure 46 Report PxMedia creates Px page with Report Pane



- Workbench PxMedia

This option allows the standard Workbench software to run inside a web browser using the Java Plugin. Workbench uses a small applet to download modules as needed to the client machine and to host the Workbench shell. These modules are cached locally in the browser.

- Ux Media

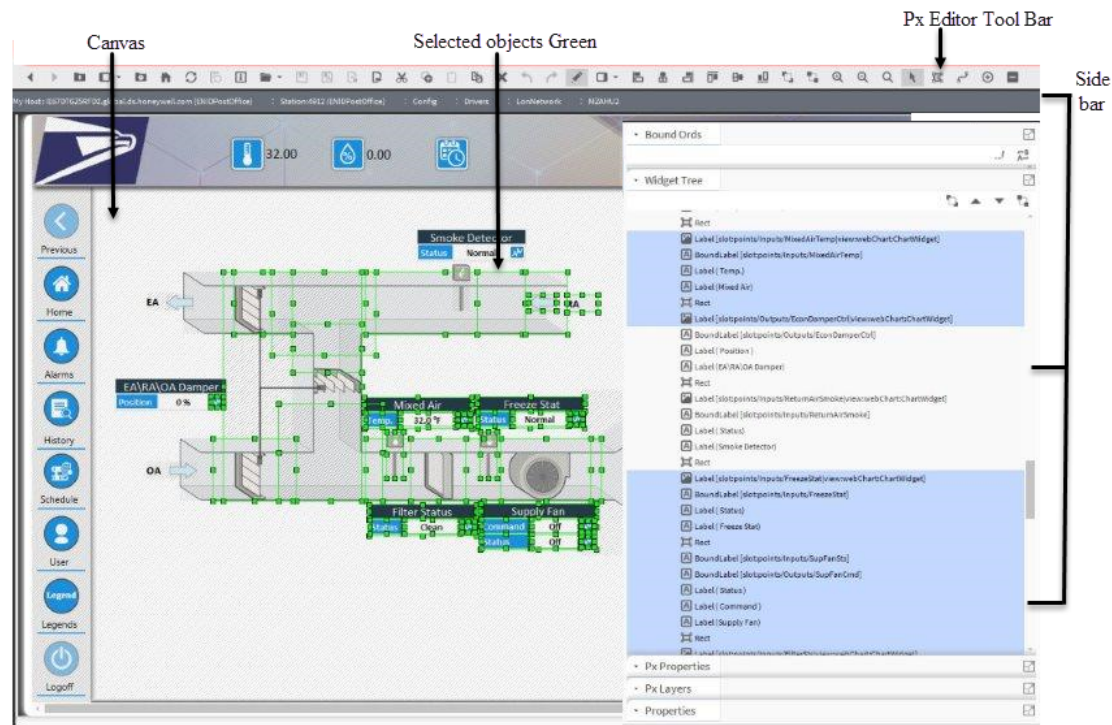
Niagara 4.10 and later provides the Ux Media type for improved graphics performance on controllers. Use the Ux Media type to achieve faster graphics load time and enable instantaneous feedback on changing values. A built-in migration tool helps you convert your existing graphics to a contemporary UI based on browser standards

About Px Editor

When you select the Px Editor view of a component, or of a Px file, the Px Editor displays the Px file in the Workbench view pane.

The figure below shows a Px View of a component with several items selected.

Figure 47 Px Editor mode



The default view of the Px Editor comprises three main areas:

- **Canvas**

The largest area of the Px Editor (by default) is the canvas pane. Typically, you place widgets on the canvas and edit them using one or more of the three side bars and additional windows. If you want to change the default size of the canvas pane that appears in new Px files, then edit the canvas pane that is in `/Files/PxFile.px`.

- **Side bar pane**

The Px Editor side bar pane appears on the far right side of the view pane only when the Px Editor is active. Use the Px Editor toolbar menu to hide or display individual side bars or to show or hide the Px Editor side bar pane.

- **Px Editor toolbar**

The Px Editor toolbar displays above the view pane when the Px Editor is active.

About Px Editor canvas

The Canvas defines the visual boundaries of the graphic page that you produce in the **Px Editor**. The Canvas is also your work area for previewing the Px file as you develop it using the tools in the **Px Editor**.

The Canvas, shown below, provides a view of the widgets as you add them and bind data to them. Most of the time, the Canvas provides a live view of any widgets that are added—without having to return to the **Px Viewer**. However, some graphic features may only appear in the **Px Viewer**.

The Canvas has the following optional work aids:

You can change the default settings of many of the **Px Editor** display features in the **Px Editor** view of the **Options** dialog (select **Tools**→**Options** from the Workbench menu bar).

- **Grid**

This is a visual aid for graphical alignment. The grid lines display vertical and horizontal lines as well as define the visible area of the page. Toggle the **Grid** on or off using the **Px Editor** menu.

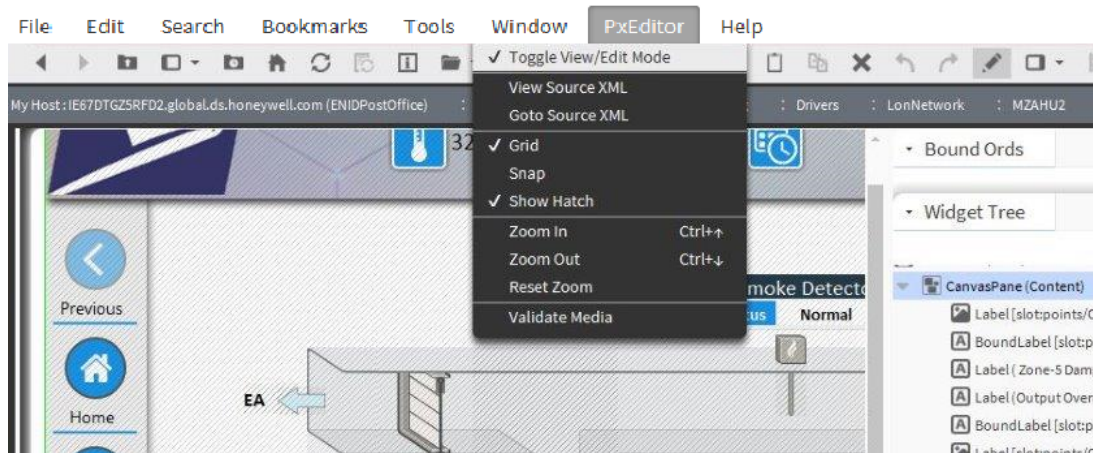
- **Show Hatch**

Hatching is an area of light-gray diagonal lines that define the boundaries of items that are placed on the Px Editor Canvas. Toggle **Show Hatching** on or off using the **Px Editor** menu.

- **Snap**

This feature is an aid to precise placement when you are using the mouse to drag an object to a particular location. Snap pulls the item to the nearest grid line or nearest object for a matching vertical or horizontal alignment. Toggle **Snap** on or off using the **Px Editor** menu.

Figure 48 Px Editor Canvas



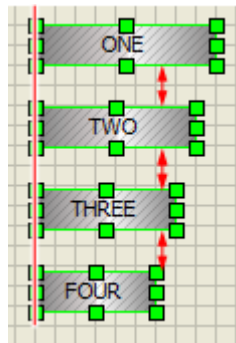
About alignment and distribution tools

You can use Px Editor tools to help you align and distribute objects in the Px Editor. The align actions are available from the Px Editor toolbar and both align and distributed features are available directly from the right-click popup menu.

- **Distribute**

If you select three or more widgets in the Px Editor, you can use the **Distribute** feature to move the widgets so that the empty space between the objects is distributed evenly. You can choose to distribute the widgets either horizontally or vertically. To use this feature, select the desired widgets and right click on one of them. Select **Distribute**→**Vertical** or **Distribute**→**Horizontal** to invoke the distribution action.

Figure 49 Four labels left-aligned, vertically-distributed



- **Align**

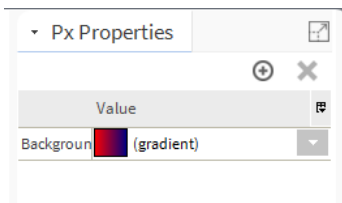
If you select two or more widgets in the Px editor, you can use the **Align** feature to move the widgets so that the left edge, right edge, or center line of the objects align. To use this feature, select the desired objects and right-click over one of them. Select **Align** from the popup menu and choose the desired align option.

About Px Properties

You can use Px Properties to facilitate graphic view design in the Px Editor view. Px Properties are entities based on a Px file markup convention that provides stylesheet-like control for properties in a Px file.

You specify Px properties using a Px Properties palette and a widget’s context-sensitive popup menu to link properties to Px objects in the appropriate object properties field. The following figure shows an example of two Px properties that are defined in the Px Properties palette.

Figure 50 Px Properties palette with two defined properties

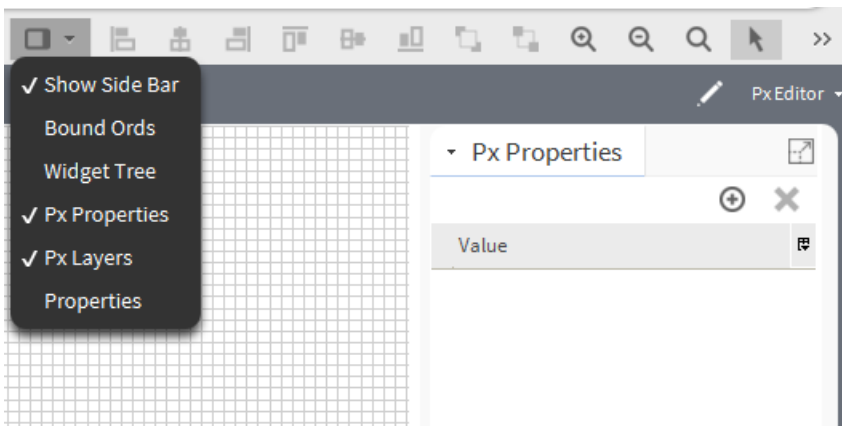


Px Properties can set values for widget properties such as font family, font size, color, background image, and others. After you create a property, you link the property to one or more specific widget properties. This gives you a single source for setting all linked property values on a page. You can use these to easily theme and update your graphics (colors, font types, font sizes, etc.) on a single Px view, as needed.

Use the PxEditor Sidebar icon (located on the main menu bar) to display or hide the Px Properties palette in the Px Editor side bar pane.

NOTE: Px properties are not globally applicable; they work only on the local Px page where they are defined. However, you can copy and paste the content of the `<properties>` xml elements from one Px file to another using the **Text File Editor** view.

Figure 51 Select the Px Properties palette from the Px Editor Side Bars option menu



Px Property Concepts

Use Px Properties to easily apply property values to Px file properties. Once defined, property parameters are assigned to or removed from individual Px object properties by **Linking** or **Unlinking**.

When you create a Px property, a `<properties>` tag pair is entered into the Px file code to hold all property definitions. Each property that is created is defined by a set of parameters (such as name, type, and value) and enclosed in a pair of `<property>` tags.

Figure 52 Px Properties assigned in the Px file code

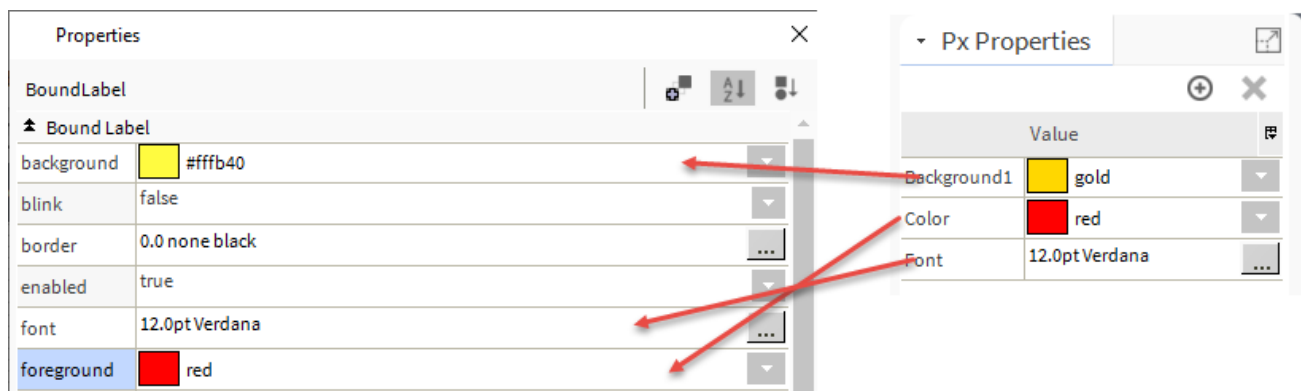
```

<properties>
  <property name="Background" type="gx:Brush" value="gold">
  </property>
  <property name="Font" type="gx:Font" value="bold 18pt Verdana">
  </property>
  <property name="Color" type="gx:Brush" value="red">
  </property>
</properties>
<content>
<Sc Px Properties
  Applied to Label Object <ent viewSize="500.0,400.0">
  <BorderPane layout="340,340,150,50" padding="0" border="1.0 inset black">
  <Label name="content" text="Floor 1 V&V 1" font="bold 18pt Verdana"
    foreground="red" background="gold"/>
  </BorderPane>
  </CanvasPane>
</ScrollPane>
</content>

```

These property parameters are assigned to or removed from individual Px object properties through a process of Linking or Unlinking. When an object property is assigned a Px Property (using the **Link** popup menu item), the property sheet displays the Px Property name in the property value field with a light green background shading.

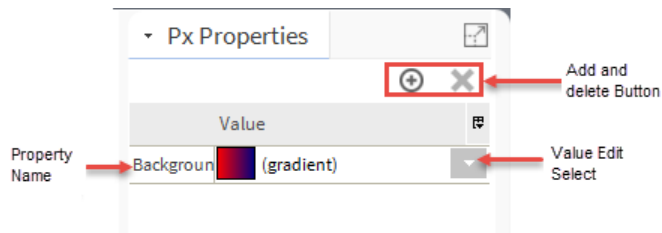
Figure 53 Linked Px Property Names Display in Property Value Field with Shading



About the Px Property palette

The **Px Property** palette works just like other Px palettes and has similar controls and displays. The following illustration shows an example Px Properties palette.

Figure 54 Px Properties Palette



Note the following about the Px Properties palette:

- Palette controls are located across the title bar and work like other Px palette controls.
- The **Add** button \oplus opens the **Add** window for adding a new Px property.
- The **Delete** button \otimes deletes any currently selected property.
- Property names are displayed in the left column of the palette table.
- Property values are displayed in the right Value column of the table.
- The value select ∇ and edit buttons are located to the right side of each property value, as required.

More About Px Layers

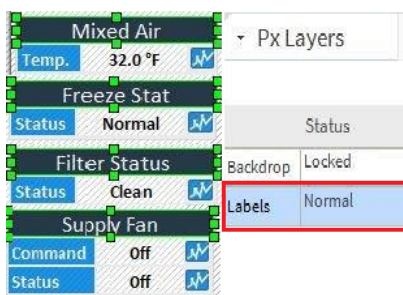
You can use Px Layers to help with graphic design work in the **Px Editor** view. Px Layers allow you to group objects in the Px Editor view by assigning them to a common layer.

The following list describes some of the things you can do with layers.

- **Select all objects assigned to a layer**

You can select all objects in a layer by selecting the desired layer in the Px Layers palette. This saves you from having to individually select each item in the layer. When objects are selected using the Px Layers palette, most Px Editor edit actions work the same way as if you had selected the objects individually. For example, when you right-click on a layer in the Px Layers palette, choosing the **Rename** or **Remove** pop-up menu items renames or deletes the selected layer (not the objects). However, the **Cut**, **Copy**, **Paste**, **Duplicate**, and **Delete** (and other) menu items perform actions on the selected objects, not the layer.

Figure 55 Selecting a layer



- **Lock objects assigned to a layer**

You can lock all layer objects by selecting the `Locked` option in the Px Layers palette **Status** column. Locked objects display normally but you cannot select them on the Px Editor canvas (you can still select them in the **Widget Tree**). This is convenient for working with stacked or overlapping objects.

NOTE: Locked object property values cannot be edited and display with a gray background in the Properties palette.

- **Hide objects assigned to a layer**

You can hide all layer objects by selecting the `Invisible` value in the Px Layers palette **Status** column. When objects are invisible you cannot select them on the Px Editor canvas but you can still see and select them in the **Widget Tree**.

NOTE: Hidden object property values cannot be edited and display with a gray background in the Properties palette.

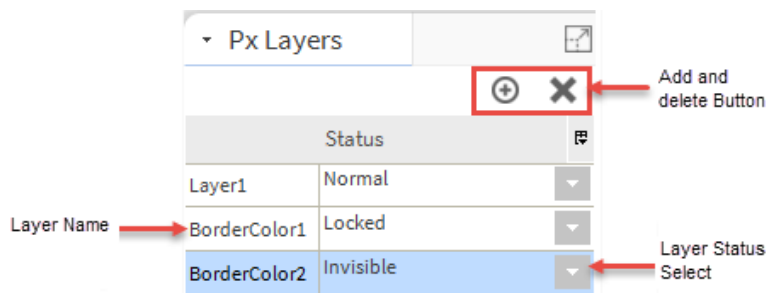
Px Layers work using a Px file markup convention that adds a `<layer>` element to the Px file to define each new layer and a `<LayerTag>` element to each object that is assigned to the layer. Normally, you work with Px layers directly in the Px Editor canvas as described in Using Px layers.

About the Px Layers palette

The **Px Layers** palette works just like other Px palettes and has similar controls and displays.

The following illustration shows an example Px Layers palette.

Figure 56 Px Layers palette



Note the following about the Px Layers palette:

- Palette controls are located across the title bar and work like other Px palette controls.
- The **Add** \oplus button opens the **Add** window for adding a new Px layer.
- The **Delete** \otimes button deletes any currently selected layer.
- Layer names are displayed in the left column of the palette table.
- Layer status values (`Normal`, `Locked`, `Invisible`) are displayed in the right Status column of the table.
- The value select \blacktriangledown button is located to the right side of each layer value.

About SVG support

The Workbench supports SVG (Scalable Vector Graphics) rendering. With the `svgBatik` module installed, you can use SVG images in your Px pages just as you would use a GIF, PNG, or JPG image.

SVG images are supported as:

- the image and background properties of a Label or Button
- the image property of a Picture
- the background property of a CanvasPane

The benefit of using an SVG image is that you can scale it up to a larger size without the image quality degrading, becoming pixelated, as happens with other image formats. Also, large SVG images scale down in size, to fit mobile device displays, retaining image quality and legibility. In order to provide this scaling functionality, a new widget called a Picture has been added to the `bajau` palette. You can create a Picture by a number of methods:

- by dragging it from the `bajau` palette
- by dragging an image file from the Nav tree

- or through the right-click menu

A Picture has a `scaleMode` property that you can use to stretch and skew an image to fit within the Picture widget's borders. A Picture can be backed with any image, including JPGs and PNGs, but to get the most benefit out of Picture scaling, SVGs are recommended.

NOTE: Labels, Buttons, and CanvasPanels support SVG display, but not scaling. These widgets will only display the SVG at its original size.

About SVG rendering

In Workbench, SVG images are rendered using the Apache Batik library, packaged in the `svgBatik` module. To keep module size small, only the display features of SVG images are supported, not the interactive features.

How an SVG image is rendered is determined by the program rendering it, in this case, Workbench. The capabilities have been reduced in order to keep the module to a reasonable size for installation on a controller. For this reason, certain features such as embedded Javascript and mouse events are not supported. In other words, the display features of SVG images are supported, but not the interactive features. CSS- and XML-based animations are supported, but Javascript-based animations are not.

NOTE: Interactive features of SVG images, such as embedded Javascript-based animations, are not supported.

Browser compatible SVG images

Scalable Vector Graphics (SVG) is an XML-based vector image format for two-dimensional graphics that has support for interactivity and animation. All major HTML-5 web browsers support SVG images.

SVG images and behaviors are defined in XML text files. As XML files, SVG images can be created and edited with any text editor, but it is more convenient to create and edit them using a drawing program.

Niagara Workbench supports rendering of SVG 1.1 images.

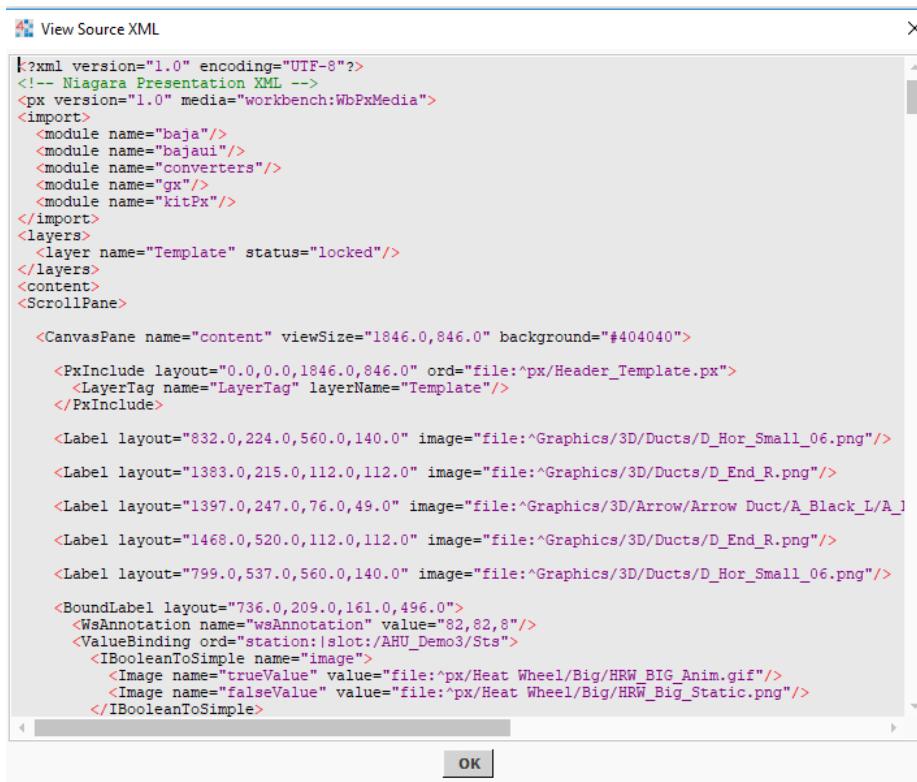
NOTE: Interactive features of SVG images, such as embedded Javascript-based animations, are not supported.

About the View Source XML Window

This separate read-only window displays the source XML of a Px file.

Although you cannot edit the Px file in this window, you can copy and paste text from the window into the **Text File Editor** or any other text editor, such as Notepad++.

Figure 57 View Source XML Window



The View Source XML window is a simple window with an **OK** button that closes the window when you click it.

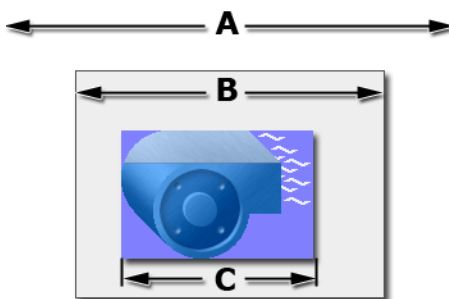
Widget layout

A tree-type hierarchy of Px elements defines widget relationships and physical layout, in a structure where parent-child relationships exist between components.

All widgets occupy a rectangular area that is defined by a position and a size. The position of a widget is defined by its x, y coordinates relative to its parent’s coordinate system and each widget may have a default (or preferred) size. The size of the widget is defined by the width and height properties. The figure below shows a simple layout of a widget (C), held by a parent widget (B), that is the child object of a widget (A).

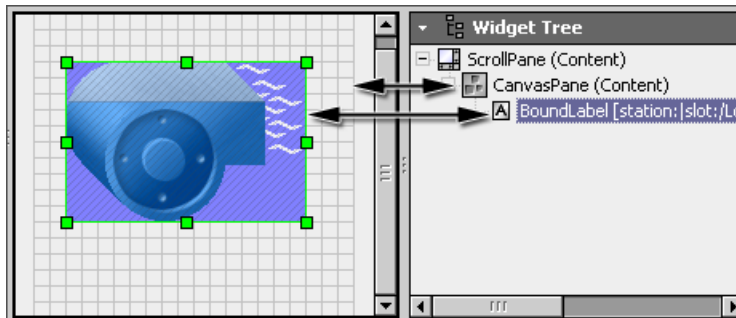
NOTE: If either of the child objects are positioned outside of the view area of the parent, they will be clipped and not be totally visible.

Figure 58 Scroll pane (A) holds canvas pane (B) with widget (C)



Some widgets are designed specifically to be container widgets that hold other widgets. These widgets are called panes. The figure below shows the pane hierarchy in the widget tree and on the Px Editor canvas.

Figure 59 Pane hierarchy in the widget tree



Types of panes

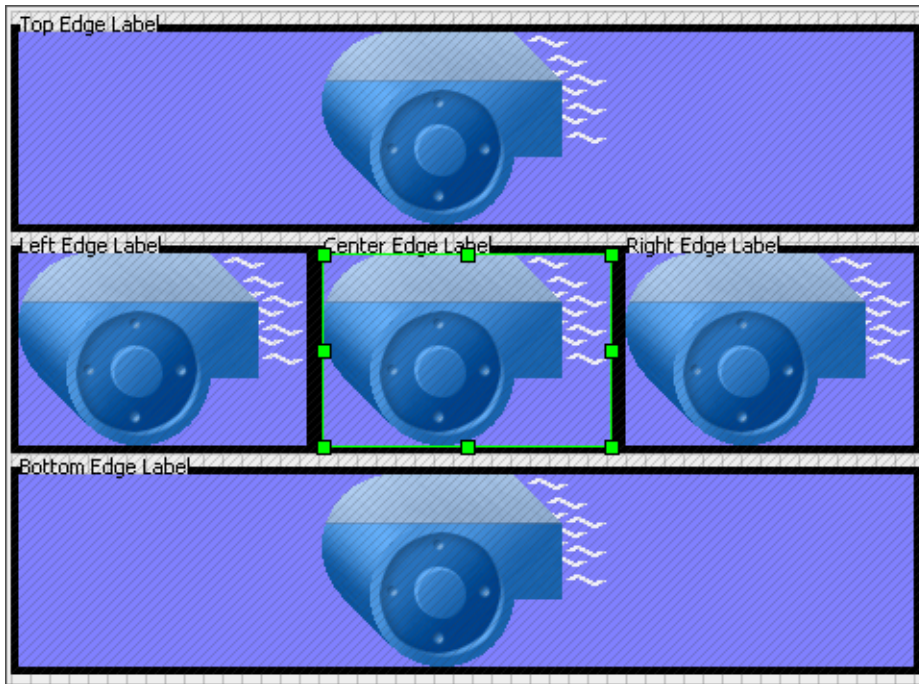
Summary of commonly used panes (widgets designed to be containers for child widgets).

Widgets that are designed to be containers for child widgets are called panes. Different types of panes provide different functions. The following list is a summary of some commonly used panes:

| Name | Description |
|-----------------|--|
| CanvasPane | Used for absolute positioning. |
| BorderPane | Used to wrap one widget and provide margin, border, and padding similar to the CSS box model. |
| ConstrainedPane | Used to constrained the graphic. |
| EdgePane | Supports five potential children: top, bottom, left, right, and center. The top and bottom widgets fill the pane horizontally and use their preferred height. The left and right widgets use their preferred width, and occupy the vertical space between the top and bottom. The center widget gets all the remaining space. See the figure below for an example. |
| ExpandablePane | Used to expand and collapse the pane. |
| GridPane | Lays out its children as a series of columns and rows. You can configure extra space in the rows and columns a number of ways using the pane's properties. |
| SplitPane | Supports two children with a movable divider between them. |
| TabbedPane | Supports multiple children - only one is currently selected using a set of tabs. NOTE: Even though only one tab is visible at a time, all tabs must be loaded on the page, even if they are not visible. If many tabs, with a lot of information, are on a single page, the page may load very slowly. |
| ScrollPane | Supports a single child that may have a preferred size larger than the available bounds. The scroll pane provides a set of scroll bars for viewing areas of the child widget that go outside of its bounds. |
| FlowPane | Lays-out its children from left to right and top to bottom, fitting as many children as possible in each row. |
| ResponsivePane | Makes Px pages render well on a variety of devices and window or screen sizes. |

The figure below illustrates the edge pane layout using borders and labels.

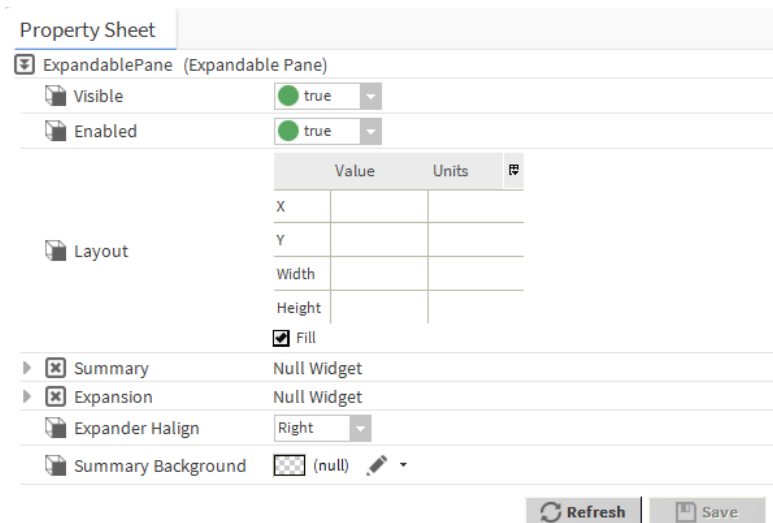
Figure 60 Edge panes, borders, and labels on a canvas pane



About ExpandablePane

ExpandablePane is available in the `bajauri` palette.

ExpandablePane contains two widgets. A summary widget is displayed all the time, with a button to the right used to expand and collapse the pane. When expanded, the expansion widget is displayed under the summary.



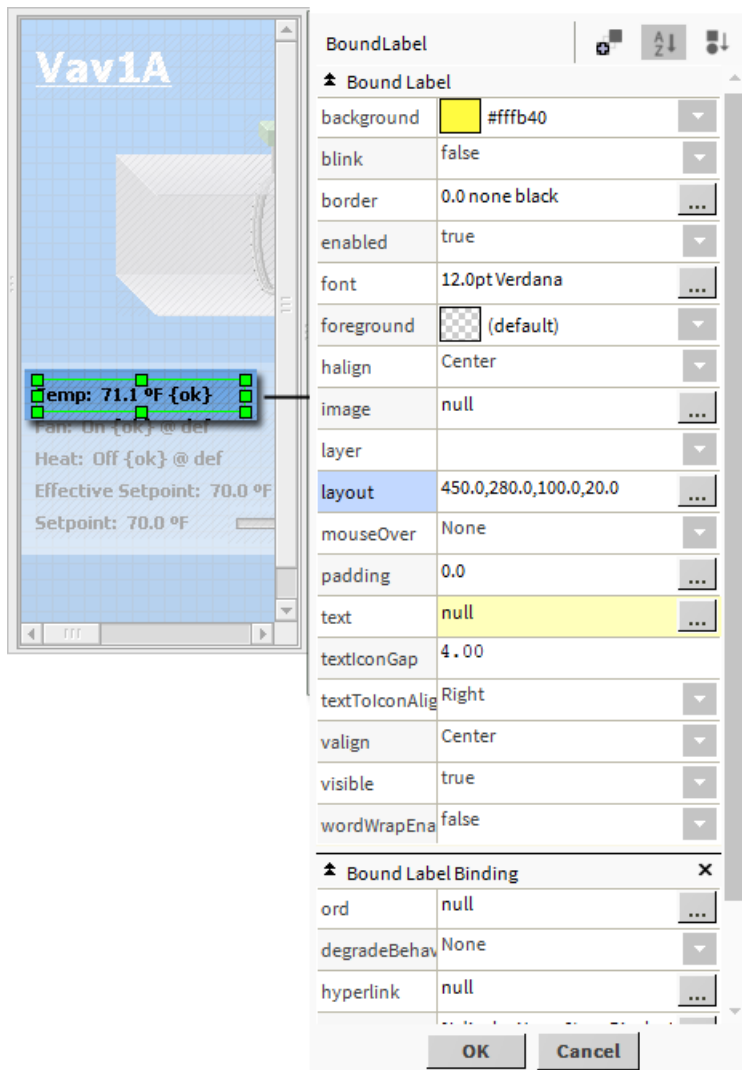
Widget properties

Widget properties define many of the features, behaviors and appearance characteristics of widgets. By editing widget properties, you set or change the widget attributes.

In the **Px Editor**, you can edit widget properties using the properties side bar or properties window and their secondary windows. Many of the properties (those that appear with the ellipsis icon) have these secondary dialog boxes that display when you click a property field. The secondary window provides more detailed property fields and options.

The figure below shows the property side bar fields for a bound label. In this example, the text property of the bound label is bound to the `slot:tempvalue`, as shown in the text property field. The binding of this value allows the dynamic presentation of the temperature information in the Px view.

Figure 61 Widget properties



About the kitPx palette

This palette provides basic Px images.

This palette contains buttons, a checkbox graphic, , meter, bar graph, set point slider and other graphics.

About the kitPxHvac palette

This palette serves the need for graphics to represent HVAC systems.

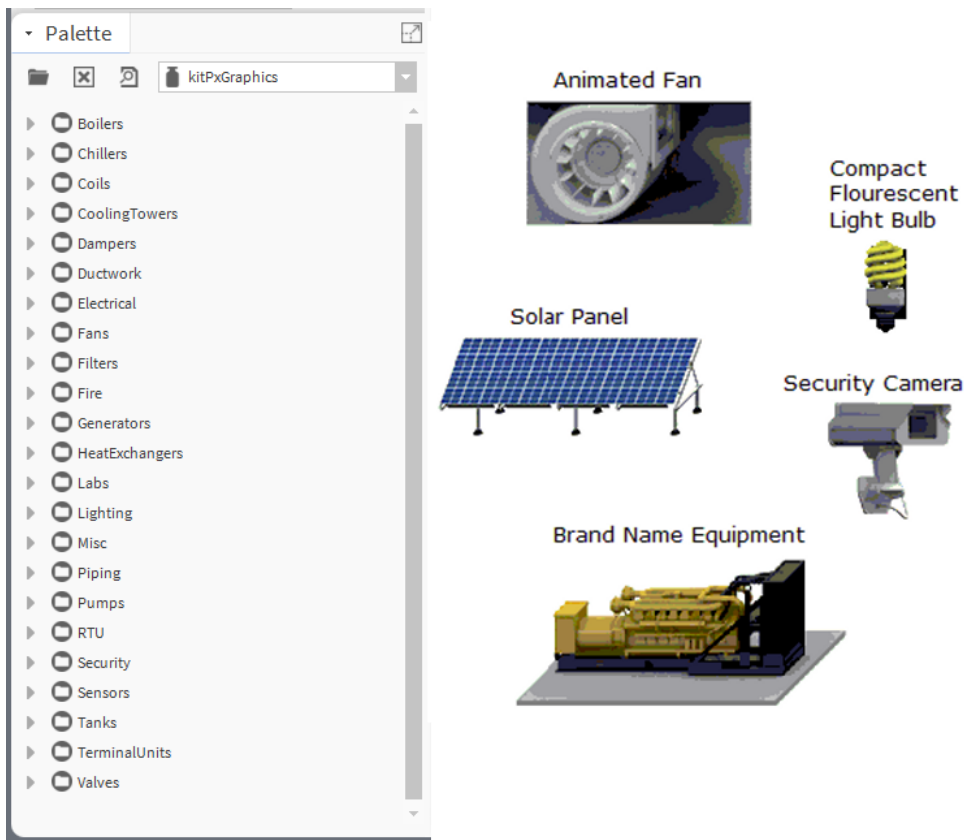
This palette contains graphics for coils, dampers, ducts, equipment, piping, valves and other HVAC components.

About the kitPxGraphics palette

This palette contains images and updated versions of many HVAC images found in the `kitPxHvac` palette, as well as many additional images.

The images in the `kitPxGraphics` palette are professionally designed, higher quality images than those in `kitPxHvac` palette. Consequently, the files take up more storage space (for example, the file size for the animated fan image increased by 40%). The figure below shows the `kitPxGraphics` palette along with sample images.

Figure 62 kitPxGraphics palette and sample images



NOTE: The `kitPxGraphics` palette is NOT a replacement for `kitPxHvac`. The `kitPxHvac` palette continues to be available in later versions.

Due to the larger file sizes and a greater number of images the `kitPxGraphics` module is quite large (19MB) compared to other graphic library modules. At that size, most controllers in the field are not able to store the full `kitPxGraphics` module. That, coupled with the fact that most job sites require only a fraction of the images in `kitPxGraphics`, there is a change in how the PxEditor handles image copying.

The PxEditor conserves storage space by copying only individual images to a remote station rather than copying the entire graphics module. This applies only to images contained in modules that are larger than 2MB, such as `kitPxGraphics`.

When using the PxEditor to copy either an image from a module or to copy a label from a module's palette that contains images, the framework automatically copies only the referenced image to the station and changes the Ord for the label to reflect the new location of the image: `^px/moduleName/pathToImage`.

These rules apply when the PxEditor copies widgets from the `kitPxGraphics` palette:

- The station must be running.
- If the module already exists on the remote station, the PxEitor does not copy the image.
- If the module does not exist on the remote station, the PxEitor copies only the referenced individual image to: `^px/moduleName/pathToImage`.
- If the station is running locally, the PxEitor copies only the referenced individual image if the module size is greater than 2MB.

NOTE: This size is set via `niagara.ui.px.maxImageModuleFileSize` in `!defaults\system.properties`.

- If the station is off line (not a running station) no image copying occurs since the PxEitor is not able to copy files to an offline station.

About the `kitPxN4svg` palette

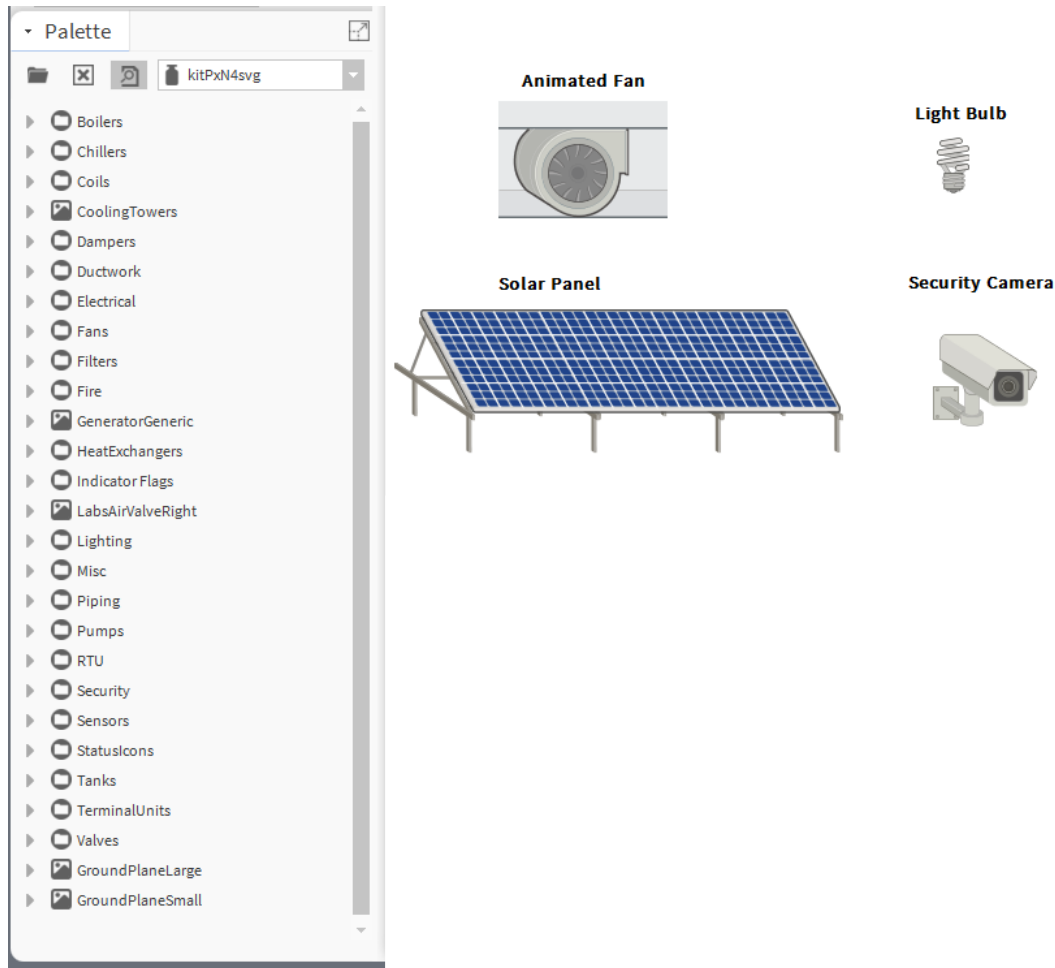
The Niagara 4.1 and later installation includes a new graphics library module, `kitPxN4svg`, that contains Scalable Vector Graphics (SVG) images and updated versions of many HVAC images found in the `kitPxGraphics` palette, as well as many new SVG images.

The images in the `kitPxN4svg` palette are professionally designed and higher quality SVG images. SVG images can be resize smaller and/or larger as per the requirement.

The files take up less storage space than the `kitPxGraphics` palette files. The figure below shows the `kitPxN4svg` palette along with sample images.

NOTE: The new `kitPxN4svg` palette is not a replacement for `kitPxGraphics`. The `kitPxGraphics` palette will continue to be available in later versions.

Figure 63 kitPxN4svg palette and sample images



In the `kitPxN4svg` palette extra properties are removed from the images which are not required.

When using the PxEditor to copy either an image from a module or to copy a label from a module's palette that contains images, only the referenced image is automatically copied to the station and the Ord for the label is changed to reflect the new location of the image: `^px/moduleName/pathToImage`.

NOTE: For the PxEditor to copy an image file to a remote station, the station must be running.

About third-party graphics collections

You can use images from a third-party graphics collection in Workbench. You must manually copy the needed images to your remote controller, add the bound labels to a Px page and then create the graphics bindings.

An alternative to buying a 3rd-party graphics collection is to create your own graphics library module and `module.palette` file of bound labels for the graphics. The `module.palette` file makes the graphics and label bindings visible in PxEditor's **Make Widget** window. Also, putting the graphics files in a module allows you to take advantage of the new individual image copying functionality in PxEditor.

About developing for portability

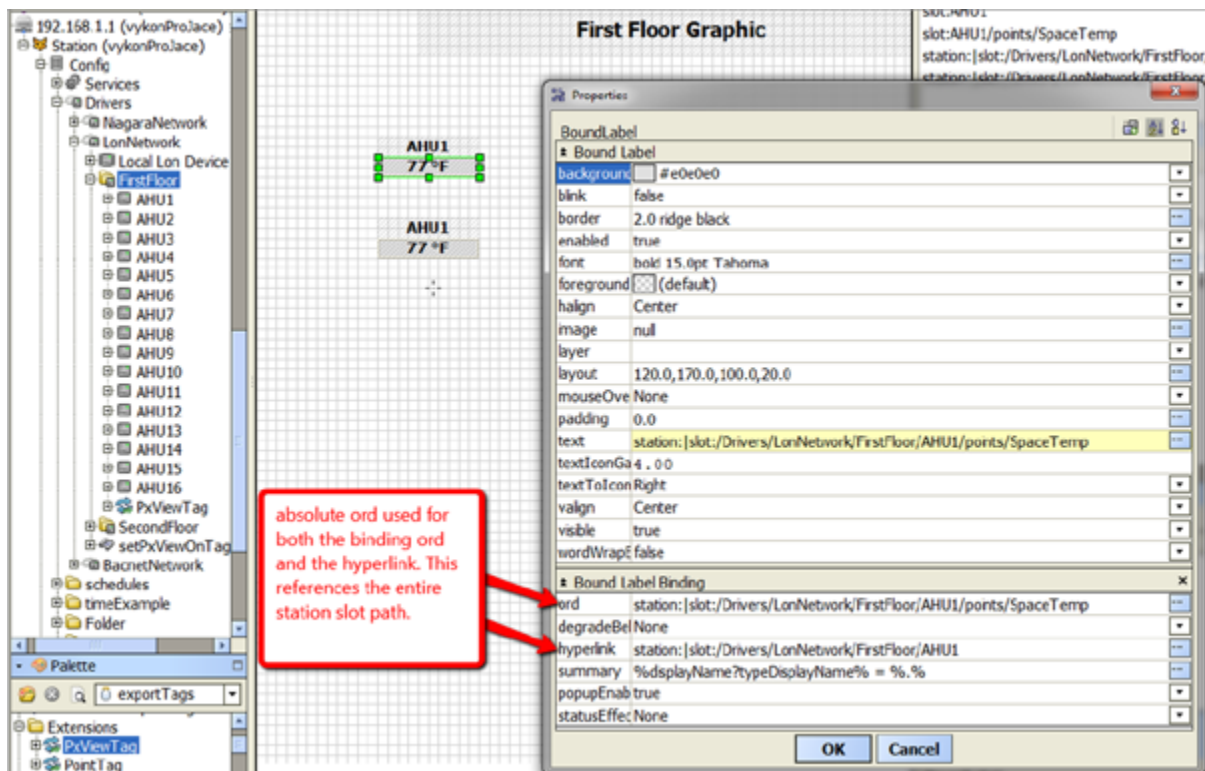
Develop Px Views that you can easily use in multiple scenarios by specifying relative ORDs in bindings and hyperlinks.

For purposes of this discussion, the term portability means reusability. In other words, being able to use the same Px View in different stations without having to edit the ORD properties for bindings and hyperlinks. Another example of portability, is being able to use the same Px view in a controller station and a Supervisor station without having to edit ORD properties. You can develop portable Px views by avoiding the use of absolute ORDs in bindings and hyperlinks. The reason to avoid using absolute ORD properties in Px views is that they are not portable between stations.

Bound label using absolute ORD

The floor plan graphic shown below has a bound label configured with a hyperlink to a detailed graphic for a specific device. You can see that both the ord binding and hyperlink properties use an absolute ord, meaning the value specifies the entire station slot path.

Figure 64 Floor plan graphic using components configured with absolute ORDs



If you wanted to use the same floor plan graphic in the Supervisor station it you would have to create NiagaraNetwork proxy points and assign the same Px view (or you could use Px view export tags). In either case, the absolute slot path used in the Supervisor must be different than that used in the JACE.

- **Supervisor absolute ORD**

```
station:|slot:/Drivers/NiagaraNetwork/VA/Richmond/myController/points/First-Floor/AHU1
```

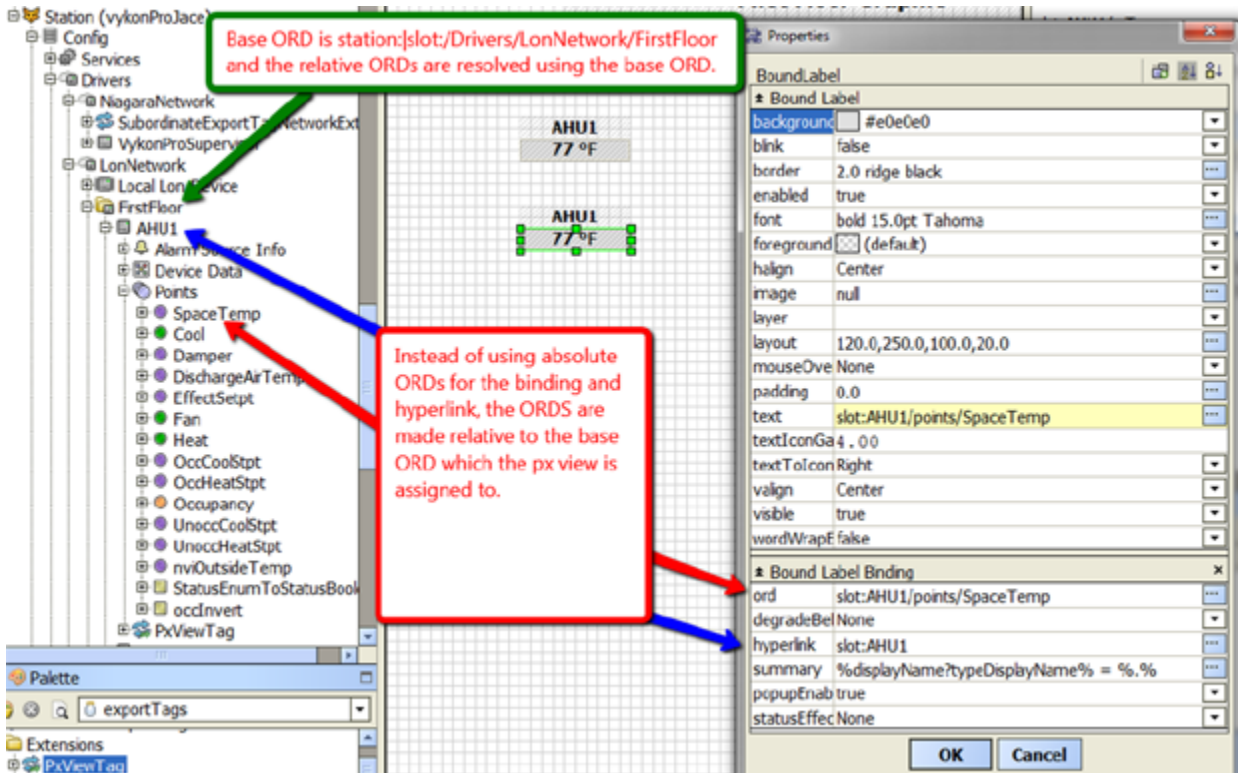
- **JACE absolute ORD**

```
station:|slot:/Drivers/LonNetwork/FirstFloor/points/FirstFloor/SpaceTemp
```

Bound label using relative ORD

Here, you have the same scenario but in this case using relative ORDs instead of absolute ORDs.

Figure 65 Floor plan graphic using components configured with relative ORDs

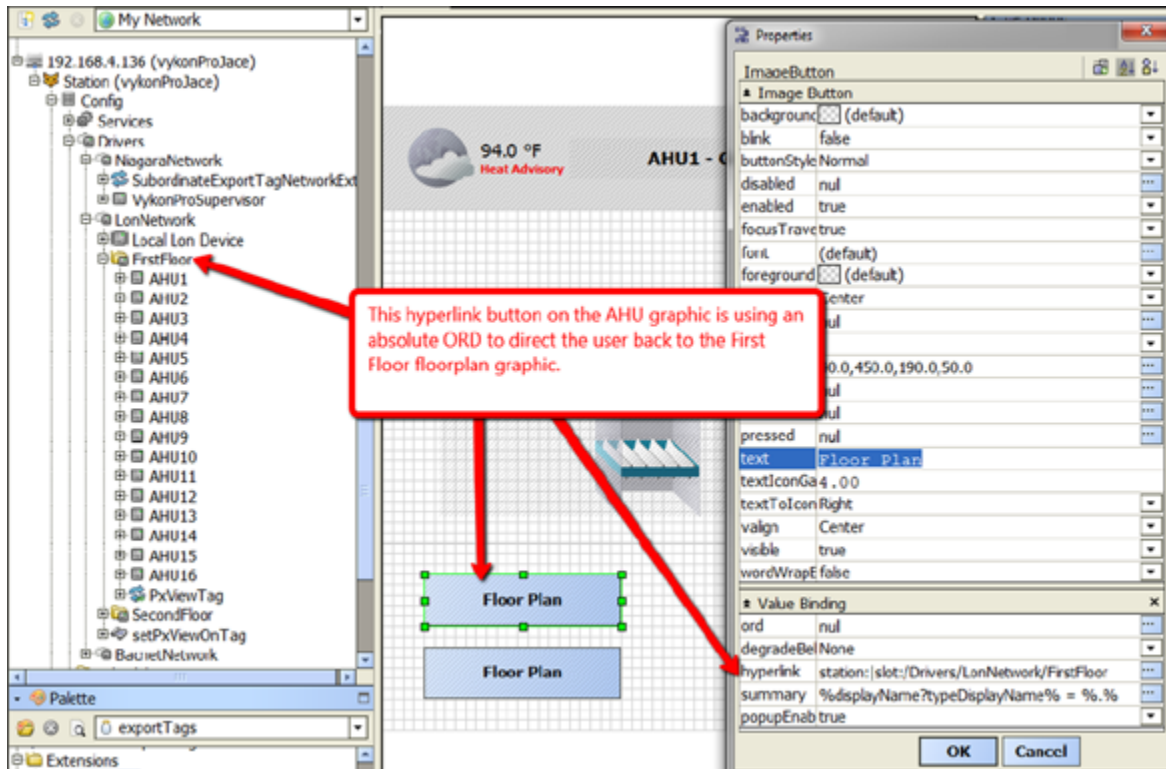


The same relative ORDs can be used in both the JACE and the Supervisor station since it is being applied against different base ORDs

Hyperlink using absolute ORD

The following example, the user has hyperlinked from the floor plan graphic to a more detailed graphic of the air handling unit (AHU). This detailed AHU graphic includes a button that is a hyperlink back to the floor plan graphic (FirstFloor). The button hyperlink is configured with an absolute ORD.

Figure 66 Graphic that includes a button with a hyperlink configured with an absolute ORD



In order to use the above graphic in the Supervisor station, you must use NiagaraNetwork proxy points and assign a Px view or by using a Px view export tag.

The absolute ORD to the First Floor graphic in the JACE is different from that in the Supervisor.

- **Absolute ORD in JACE**

```
station:|slot:/Drivers/LonNetwork/FirstFloor
```

- **Absolute ORD in Supervisor**

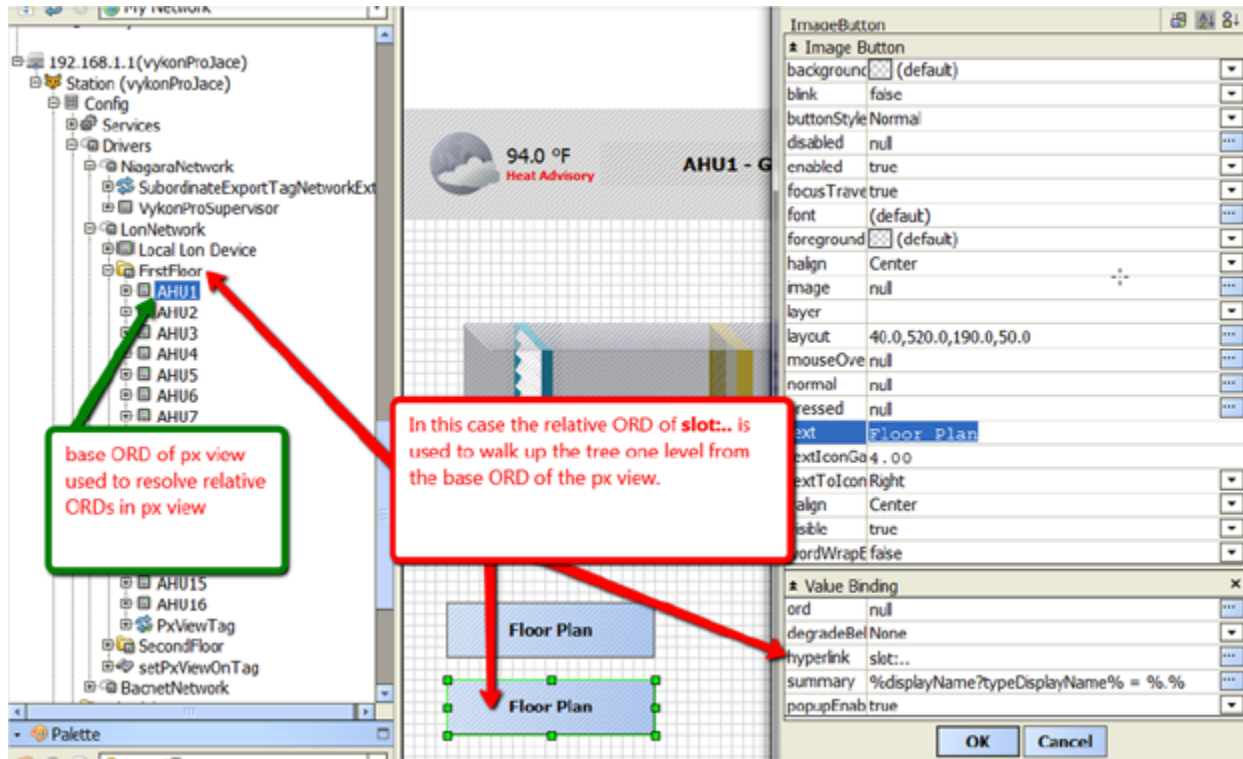
```
station:|slot:/Drivers/NiagaraNetwork/VA/Richmond/myController/points/FirstFloor
```

Hyperlink using relative ORD

Using relative ORD properties, it is possible to navigate back up the tree by entering two periods in the path, similar to how directories can be navigated in a DOS command window.

NOTE: You can back up multiple levels using this syntax: ". . .".

Figure 67 Graphic includes a button with a hyperlink configured with a relative ORD



- **Relative ORD in Px page**

slot:..

- **Base ORD in JACE**

station:|slot:/Drivers/LonNetwork/FirstFloor/AHU1

- **Base ORD in Supervisor**

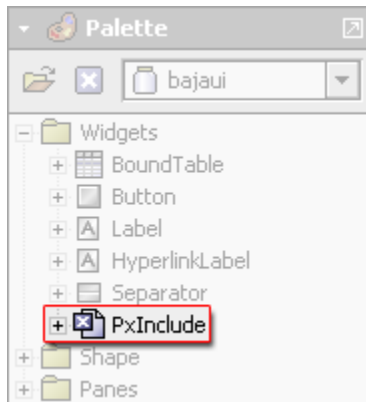
station:|slot:/Drivers/NiagaraNetwork/VA/Richmond/myController/points/First-Floor/AHU1

About the PxInclude Widget

This widget lets you embed a single Px file in another Px file.

The `PxInclude` is a widget located in the `bajau` palette, as shown in the following illustration.

Figure 68 The PxInclude widget



This widget provides a way for users to embed a single Px file in another Px file. The key benefit of this feature is the ability to create reusable Px views and embed them into one or many other Px views. The following sections describe the widget.

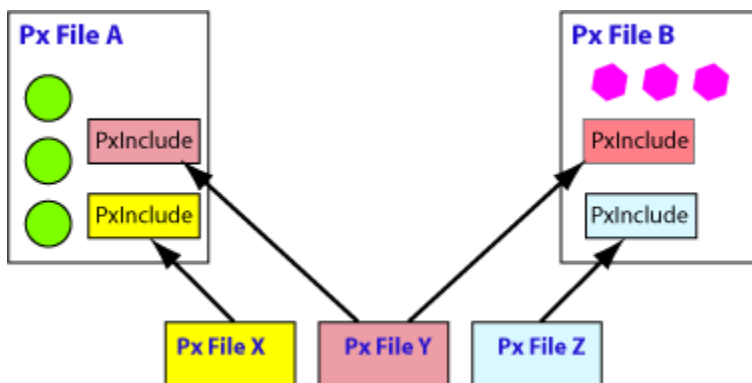
PxInclude Widget concepts

The PxInclude widget may have a single Px file associated with it, providing one include per widget, however, you can put more than one PxInclude widget in a Px file.

Depending on the design of a Px file, a single parent Px file may contain many child Px files by using many PxInclude widgets.

The following diagram illustrates the basic concept of embedding one Px file in one or more other Px files.

Figure 69 PxInclude Concept Diagram

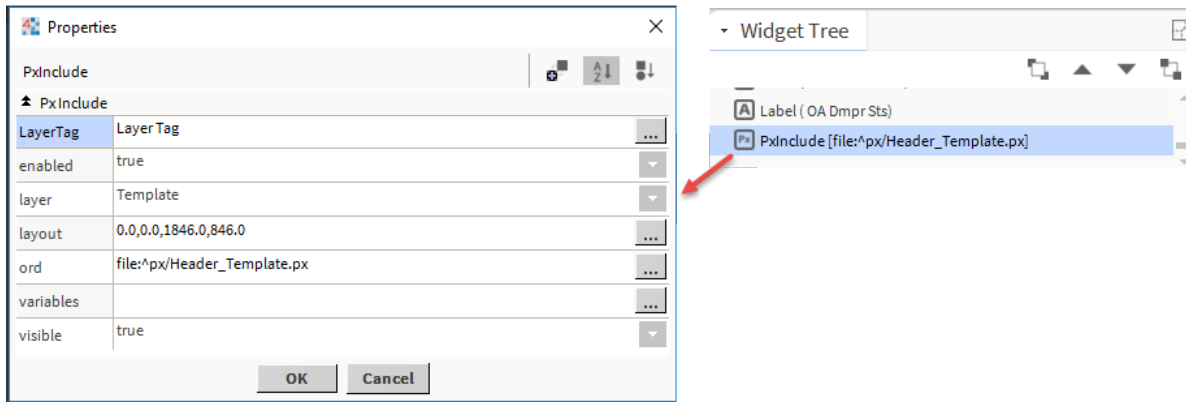


Note the following about the Px files represented in this diagram:

- Px files X, Y, and Z are included in other Px files
- Px files X, Y, and Z may display as independent views, as well as be embedded using the PxInclude widget.
- Px file Y is included in both Px file A and Px file B.
- Changes to files X, Y, or Z are displayed in their parent Px files (A or B), via the PxInclude, when the parent view is refreshed.

PxIncludes work by defining an Ord property value that points to the included file, as shown in the following illustration.

Figure 70 PxInclude points to the file to be embedded



This example shows the `incl.px` file embedded in another Px file. The embedded filename is visible in both the **Properties** window and in the **Widget Tree**.

When PxIncludes are added to a Px file, the include markup appears as follows in the Px code (visible in the **Text File Editor** view).

Figure 71 Simple PxInclude markup in a parent Px file

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Niagara Presentation XML -->
<px version="1.0" media="workbench:WbPxMedia">
<import>
  <module name="baja"/>
  <module name="bajaiui"/>
  <module name="gx"/>
  <module name="history"/>
  <module name="workbench"/>
</import>
<content>
<ScrollPane>

  <CanvasPane name="content" viewSize="500.0,400.0">
    <PxInclude layout="100,80,350,250" ord="file:^px/incl.px"/>
  </CanvasPane>
</ScrollPane>
</content>
</px>
```

Included File

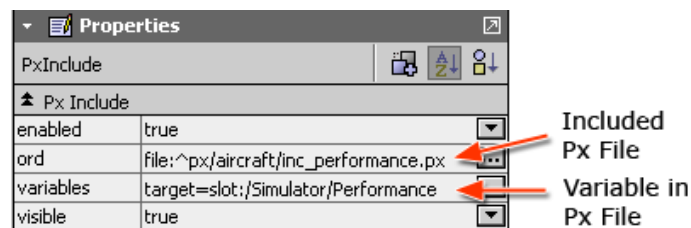
Types of PxInclude Widget properties

The PxInclude Widget has the following properties.

| Name | Description |
|------------------|---|
| enabled | This property has two options: true and false. Setting the option to false causes the widget to stop working. Values may display but are not reliable and do not update. |
| ord | The value of this property (an ORD) identifies the Px file that is linked using the PxInclude widget. See an example in the figure below. |
| variables | This optional property identifies the value of any variables in the Px file specified by the ord property. Clicking on the variable field displays the Variables window, where you can define the variable value. See an example in the figure below. |

| Name | Description |
|----------------|--|
| layout | his property contains parameter fields for values that prescribe PxInclude widget size and location. |
| visible | This property has two options: true and false. Setting the option to false causes the widget to not display. |

Figure 72 Example PxInclude Properties Palette in the Px Editor



Note the following about this illustration:

- The **ord** property `file:^px/aircraft/inc_performance.px` specifies to include the Px file named `inc_performance.px`. This file is located in the station's (^) `px/aircraft` folder.
- The **variables** property value indicates that the included file has a single variable named `target` and that this variable is defined as `slot:/Simulator/Performance`.

The **visible** property value is set to the default value `true`.

Use PxInclude Widgets

Use the **Make Widget Wizard** to select the desired Px file and choose which variables to bind.

Like other widgets, you can add a PxInclude to the Px Editor canvas in several ways, including the following:

- **Drag (or cut and paste) the widget from a palette**
This method simply drops a widget onto the Px Editor canvas, adding the widget to the current Px file. After adding the widget to the Px canvas, you must edit the `ord` property to add the desired Px file.
- **Drag (or cut and paste) a Px file from the Nav tree**
This method adds a PxInclude widget to the current Px file with a Px file already connected to the PxInclude `ord` property.
- **Drag (or cut and paste) a component from the Nav tree**
This initiates the **Make Widget Wizard**. This method allows you to use the **Make Widget Wizard** to select the desired Px file and choose which variables to bind.

NOTE: Only the `ord` variables that are selected in the bottom left window of the **Make Widget Wizard** are available for configuring after the PxInclude file is imported.

NOTE: For the Px file that you are including, you probably want to remove the root scroll pane that comes with the default Px file. Just use a canvas pane as the root. The scroll pane has a property called `border policy`, which is set to 'always' by default. This border is visible when the include is viewed in its parent Px file.

Figure 73 Remove the scroll pane from a Px Include to avoid displaying a border

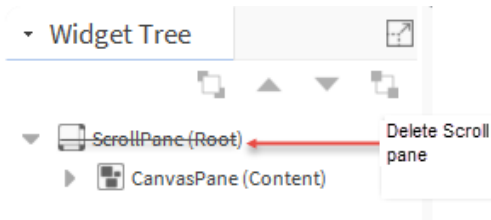
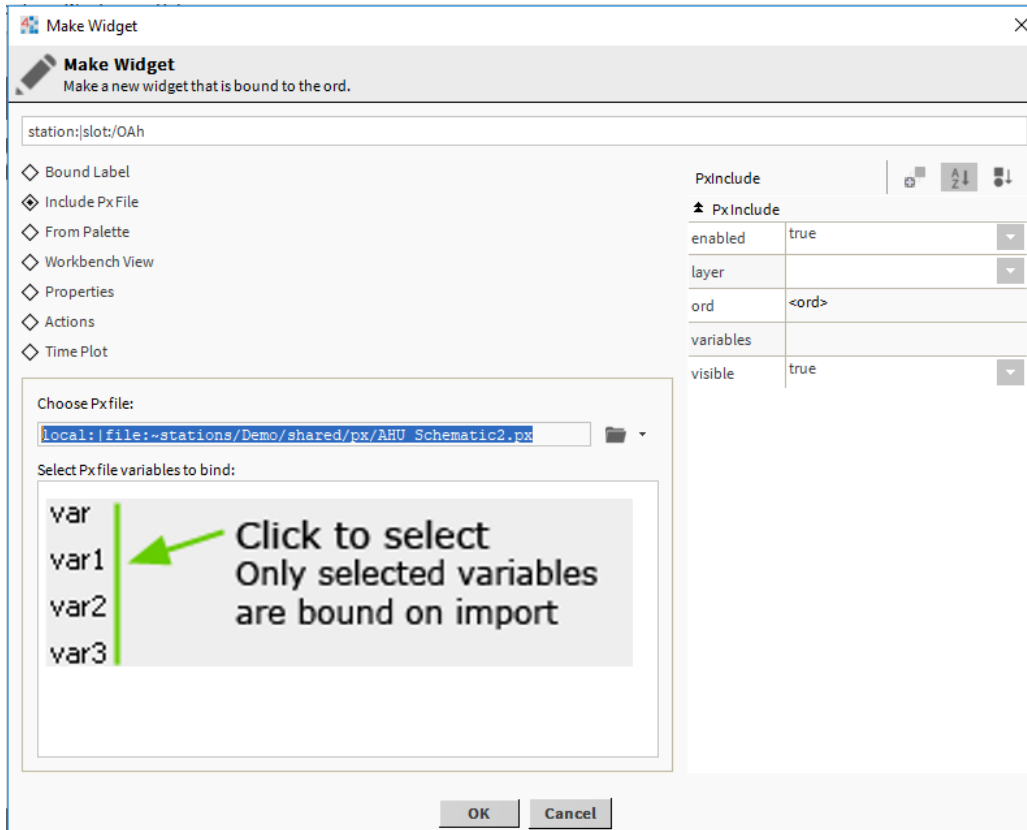


Figure 74 Adding a PxInclude Using the Make Widget Wizard



When you complete the **Make Widget Wizard**, you may have the PxInclude widget configured fully, or edit it further, if needed.

About ORD Variables

ORD variables are optional portions of an ORD binding that you can create in order to provide more flexibility or reusability in your Px files.

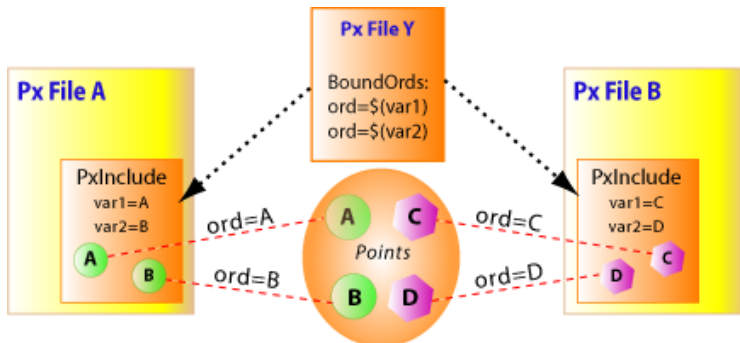
You can use a variable for the complete ORD or identify a variable part of an ORD by using the following special syntax in a bound component ord property field. $\$(var)$

Where:

- $\$$ is a special character used to identify an ORD variable
- (var) is a named variable (may be any text string) inside a set of parentheses. The name in parentheses is the name that is substituted by a literal text string in the ord Variables window after the child Px file is included in the parent Px file.

The following illustration shows the general relationships between Px files that use ORD variables in PxInclude widgets.

Figure 75 PxIncludes Allow Px File Reuse With Variable ORDs



Note the following about this example:

- Px File Y uses two ORD variables: `$(var1)` and `$(var2)`.
- Px File Y is embedded in two different Px files using PxInclude widgets.
- In Px File A, `var1` is defined as A, resulting in a binding `ord=A`. The `ord=A` value binds the widget to a Point A, so a value from that Point A is displayed in the parent view (Px File A).
- In Px File B, `var1` is defined as C, resulting in a binding `ord=C`. The `ord=C` value binds the widget to a Point C, so a value from that Point C is displayed in the parent view (Px File B).
- Similar to the descriptions above, in Px files A and B, `var2` is defined as `ord=B` and `ord=D`, respectively, resulting in different point values displayed in Px Files A and B.

About Nav file elements

You can use nav file elements to create custom nav files.

When you create a new nav file, in addition to the xml declaration, the file includes two default elements, as listed and shown below.

- **<nav> element**
a single opening and closing set of `<nav>` elements hold all the nodes in the tree
- **<node> element**
the default `<node>` element includes an ORD path that points to the station root directory:
`ord='station:|slot:/'`

You can edit these nodes directly in the **Text Editor** or you can use the tools provided in the **Nav File Editor**.

Figure 76 Default nav file with home node only

```
<?xml version='1.0' encoding='UTF-8'?>
<nav version='1.0'>
  <node name='Home' ord='station:|slot:/' icon='module://icons/x16/home.png'>
  </node>
</nav>
```

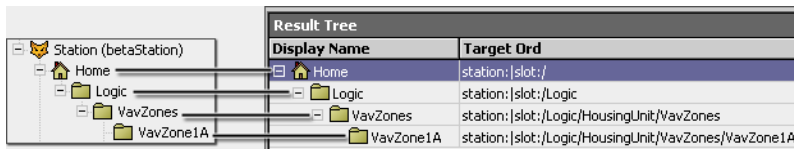
The following graphic shows what the elements in your nav file look like (when viewed in a text editor) once you add a few nodes to your tree.

Figure 77 Simple nav file with home node only

```
<?xml version="1.0" encoding="UTF-8"?>
<nav version="1.0">
<node name='Home' ord='station:|slot:/' icon='module://icons/x16/home.png'>
  <node name='Logic' ord='station:|slot:/Logic' icon='module://icons/x16/folder.png'>
    <node name='VavZones' ord='station:|slot:/Logic/HousingUnit/VavZones'
      icon='module://icons/x16/folder.png'>
      <node name='VavZone1A' ord='station:|slot:/Logic/HousingUnit/VavZones/VavZone1A'
        icon='module://icons/x16/folder.png' />
      </node>
    </node>
  </node>
</nav>
```

The following graphic shows what the elements in a nav file look like in the Nav tree and in the **Nav File Editor** once you add a few nodes to your tree.

Figure 78 Nav file viewed in Nav tree and in Nav File Editor (Result Tree)



About Nav File Editor source objects

Nav File source objects are those objects that may be used for creating your nav file.

In order to appear in the source object pane, the source object must be a Px file or a component with a Px view assigned to it. The source objects populate the pane view based on the position of the **Show Components** and **Show Files** buttons, as described in [About Nav File Editor buttons, page 127](#).

Figure 79 Source objects available in the source pane

| Source Objects | | | 28 objects |
|----------------|-------------|---|------------|
| Name | Type | Path | |
| VavZone1A | baja:Folder | station: slot:/Logic/HousingUnit/VavZones/VavZone1A | |
| VavZones | baja:Folder | station: slot:/Logic/HousingUnit/VavZones | |
| Logic | baja:Folder | station: slot:/Logic | |
| AirHandler.px | file:PxFile | file:^px/building/AirHandler.px | |
| Building.px | file:PxFile | file:^px/building/Building.px | |
| FloorPlan.px | file:PxFile | file:^px/building/FloorPlan.px | |

To use the source objects pane for creating the nav file structure, you drag components or files from the source objects pane down to desired tree location in the result tree pane.

About Nav File Editor result tree

The result tree pane is where you build the nav file structure.

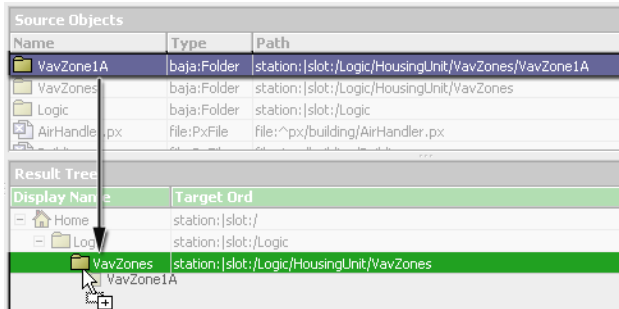
You can drag and drop components and files from the source objects pane to create new nodes in your tree. You can also create nodes directly, using the **New** button or by using the popup menu directly in the result tree pane.

Figure 80 Result tree pane

| Result Tree | | 4 objects |
|--------------|---|-----------|
| Display Name | Target Ord | |
| Home | station: slot:/ | |
| Logic | station: slot:/Logic | |
| VavZones | station: slot:/Logic/HousingUnit/VavZones | |
| VavZone1A | station: slot:/Logic/HousingUnit/VavZones/VavZone1A | |

The figure below shows an example of a source object being dragged to a specific location in the Nav tree. The result tree shows the exact hierarchy of your Nav tree and allows you to set the indent levels by where you drop the component.

Figure 81 Dragging a source object to the result tree pane



About Nav File Editor buttons

The Nav File Editor buttons are aligned horizontally across the bottom of the Nav File Editor. They include controls that you use to create, edit, delete, and rearrange nodes in the nav file.

The following list describes each of the of the Nav File editor buttons.

- **New**

This button displays the **New Node** window that allows you to create a new node in the nav file tree by specifying a name, target ORD and icon for the node. The new node is created as a child node to the active point in the tree (the one you have selected).

- **Edit**

This button is available when you select a node in the Result Tree pane. It displays the **Edit Node** window that allows you to edit the node name, target ORD, and icon.

- **Delete**

This button deletes a selected node in the Result Tree pane.

- **Move Up**

This button allows you to move a selected node (or nodes) up in the nav file tree hierarchy.

- **Move Down**

This button allows you to move a selected node (or nodes) down in the nav file tree hierarchy.

- **Show Components**

This button toggles on and off. Toggle this button **ON** to display all eligible components in the Source Objects pane.

- **Show Files**

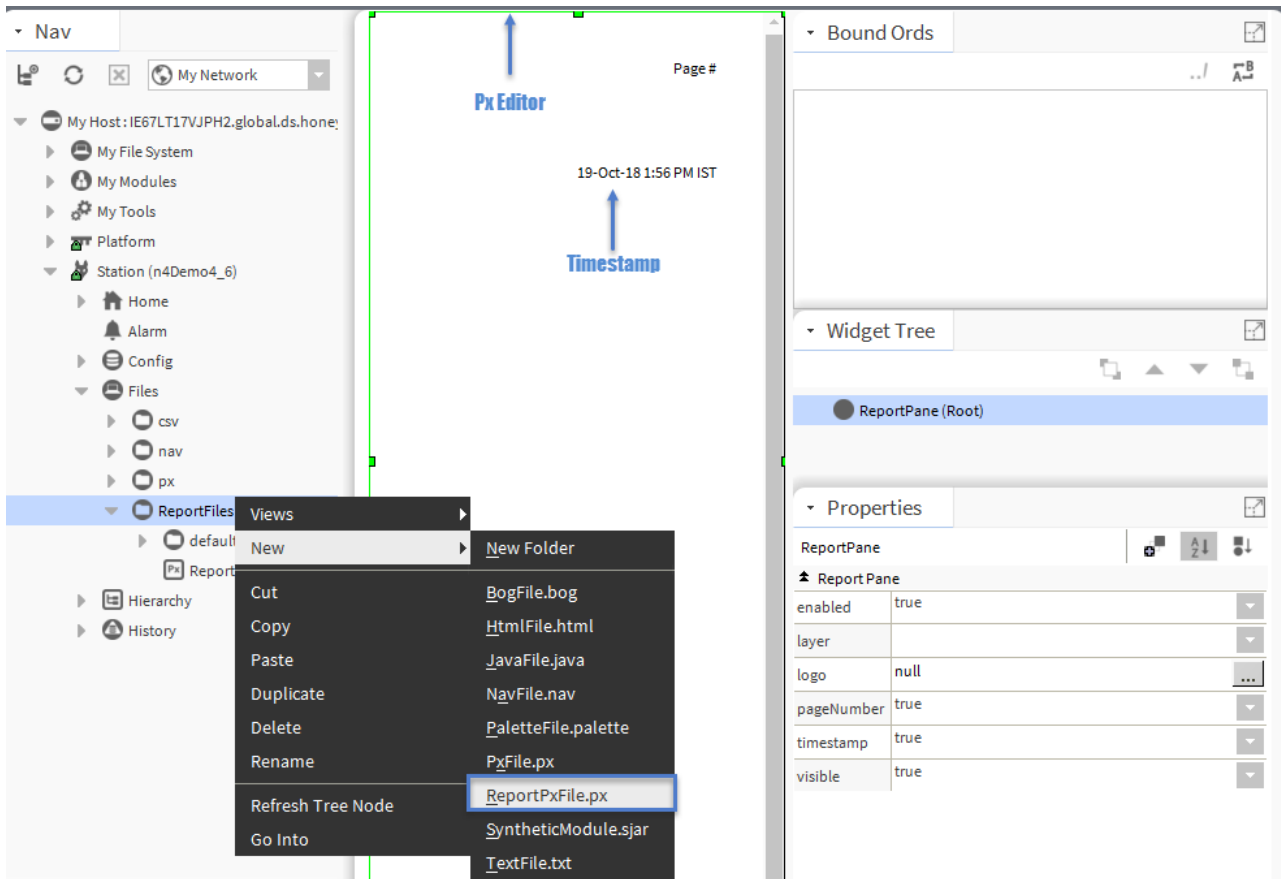
This button toggles on and off. Toggle this button **ON** to display all eligible Px files in the Source Objects pane.

About the ReportPxFile

ReportPx files are available from the **New** menu item on a popup menu when you right-click on an appropriate container for a Px file.

The ReportPxFile is a regular Px file that has been pre-loaded with a ReportPane widget, as shown here.

Figure 82 Creating a new ReportPxFile



About the Grid Table view

The Grid Table view is a view of the ComponentGrid component. This view provides a tabular display of all the points that are assigned to the ComponentGrid.

In the Grid Table view, you can write to points (issue commands) using the popup menu, if you have write permissions. Points in the table also display color-coded status, as shown.

Figure 83 Grid Table view

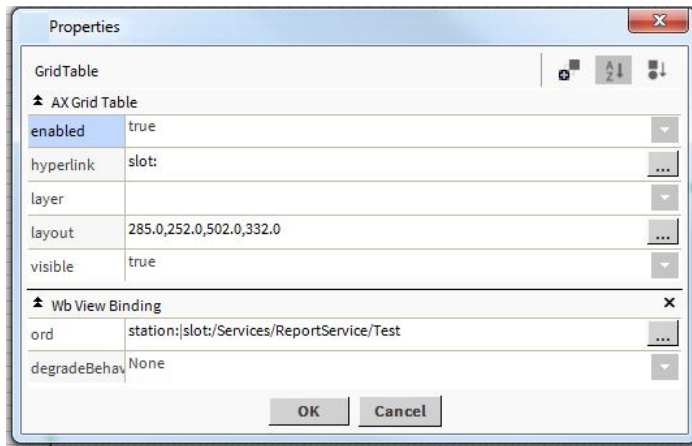
| Name | Point | Value | Date | Time | |
|----------|----------------|----------|-----------|---------|--|
| VavZoneA | DamperPosition | 70.79 % | 23-Apr-07 | 8:04 AM | |
| VavZoneA | Fan | On | 23-Apr-07 | 8:04 AM | |
| VavZoneA | Occupied | Occupied | 23-Apr-07 | 8:04 AM | |
| VavZoneA | AirFlow | 603.9 | 23-Apr-07 | 8:04 AM | |
| VavZoneA | HeatingMode | Off | 23-Apr-07 | 8:04 AM | |
| VavZoneA | HeatingCoil | 70.79 % | 23-Apr-07 | 8:04 AM | |
| VavZoneA | SetpointTemp | 72.00 °F | 23-Apr-07 | 8:04 AM | |

Emergency Override
 Emergency Auto
Override
 Auto
 Set

The table controls and options are similar those that are used in most Workbench table controls.

When you use a GridTable view in a Px page, the editable table properties are accessed in the Properties window, as shown below:

Figure 84 Grid Table properties in Px Editor



- **Enabled**
When set to `true`, the table in the Px page interface is commandable using the popup menu. When this property is set to `false`, the display is still visible but not commandable.
- **Visible**
When set to `true`, the table in the Px page interface is visible. When this property is set to `false`, the display is not visible.
- **Ord**
This Ord is the link binding the display to the GridComponent that defines the table.
- **Degrade Behavior**
These options specify how the table behaves when the binding communications are not available.

About the Grid Label Pane view

The Grid Label Pane view is a view of the ComponentGrid component. This view provides a tabular display of all the points that are assigned to the ComponentGrid.

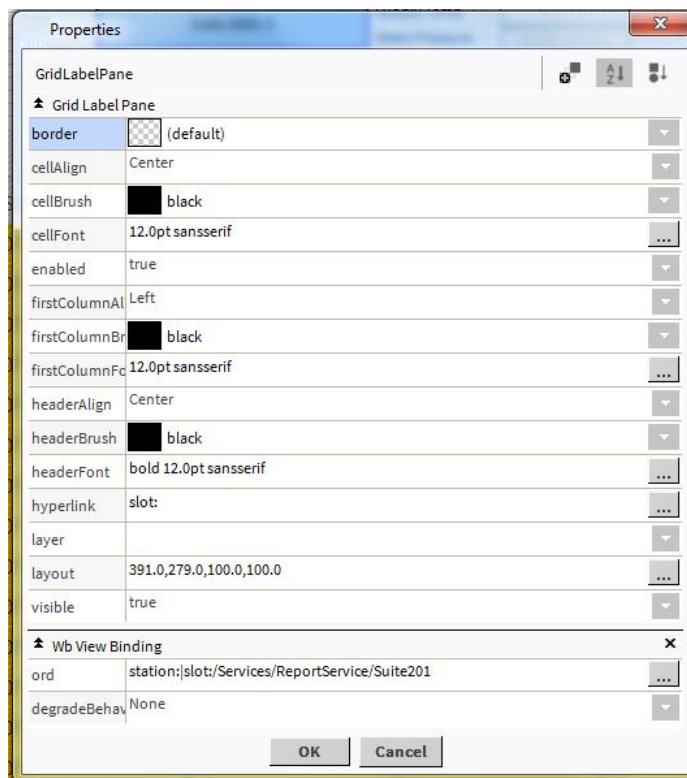
In the Grid Table Pane view, you can write to points (issue commands) using the popup menu, if you have write permissions. Points in the table also display color-coded status, as shown.

Figure 85 Grid table view of the GridComponent

| Zone ID | Occupancy | Space Temp | Setpoint | Occupied Cool | Occupied Heat | Air Flow | SAT |
|----------|-----------|------------|----------|---------------|---------------|----------|-------|
| VAV_M_82 | Occupied | 0.0°F | 0.0°F | 0.0°F | 0.0°F | 0 dm | 0.0°F |
| VAV_M_89 | Occupied | 0.0°F | 0.0°F | 0.0°F | 0.0°F | 0 dm | 0.0°F |
| VAV_M_91 | Occupied | 0.0°F | 0.0°F | 0.0°F | 0.0°F | 0 dm | 0.0°F |
| VAV_M_93 | Occupied | 0.0°F | 0.0°F | 0.0°F | 0.0°F | 0 dm | 0.0°F |
| VAV_M_94 | Occupied | 0.0°F | 0.0°F | 0.0°F | 0.0°F | 0 dm | 0.0°F |
| VAV_M_95 | Occupied | 0.0°F | 0.0°F | 0.0°F | 0.0°F | 0 dm | 0.0°F |
| VAV_M_96 | Occupied | 0.0°F | 0.0°F | 0.0°F | 0.0°F | 0 dm | 0.0°F |
| VAV_M_97 | Occupied | 0.0°F | 0.0°F | 0.0°F | 0.0°F | 0 dm | 0.0°F |
| VAV_M_98 | Occupied | 0.0°F | 0.0°F | 0.0°F | 0.0°F | 0 dm | 0.0°F |
| VAV_M_95 | Occupied | 0.0°F | 0.0°F | 0.0°F | 0.0°F | 0 dm | 0.0°F |

In the Px Editor view, you can use the GridLabelPane widget to layout the table on a Px page and edit the properties of the display properties, as shown in the figure below. These properties allow you to change the font-type and color, as well as set the text alignment of the table cells, table heading, and first column properties.

Figure 86 Grid Label properties in Px editor



GridLabelPane properties include the following:

- **Cell Align**

This property provides options for setting the alignment of cell content (left, right, center, fill).
- **Cell Brush**

This property specifies the color of the content (for example, font color) within the table cells.
- **Enabled**

When set to `true`, the table in the Px page interface is commandable using the popup menu. When this property is set to `false`, the display is still visible but not commandable.
- **First Column Align**

This property provides options for setting the alignment of cell content (left, right, center, fill) for the first column in the table.
- **First Column Brush**

This property specifies the color of the content within the first column table cells.
- **Header Align**

This property provides options for setting the alignment of cell content (left, right, center, fill) for the top row of the table (the table heading).
- **Header Brush**

This property specifies the color of the content within the table heading cells
- **Header Font**

This property allows you to set the font-type and font properties for the table heading.
- **Layout**

This property allows you to specify the size and location of the table in relation to its parent (X, Y, coordinates, height and width). Alternatively, you can select a fill option to have the table expand to the limits of its parent container.
- **Visible**

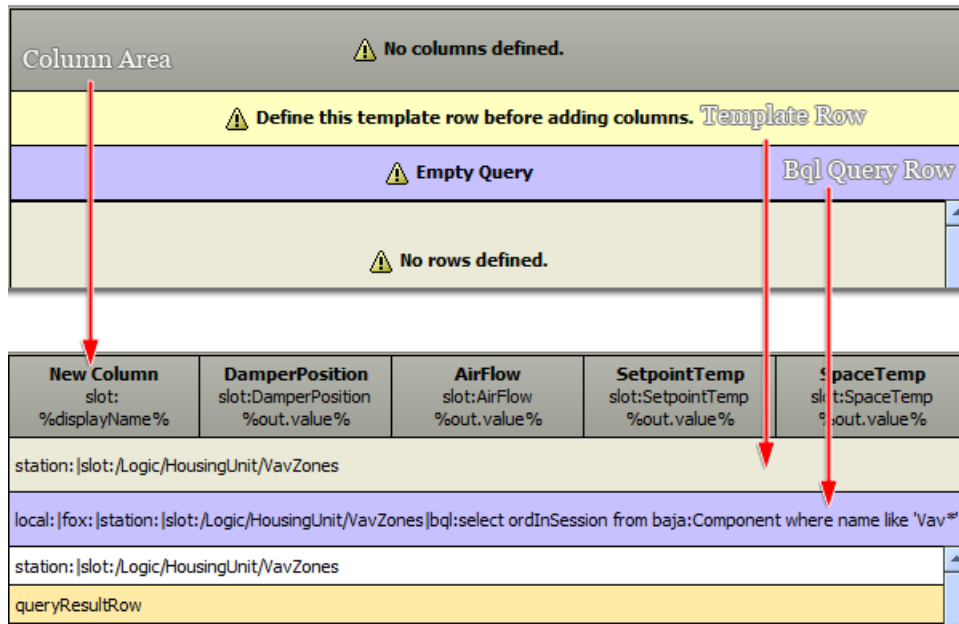
When set to `true`, the table in the Px page interface is visible. When this property is set to `false`, the display is not visible.

About the Grid Editor view

The Grid Editor view is one view of the ComponentGrid component. This view provides a layout display that you use to assign points that you are going to use in a report. The figure below shows the Grid Editor view with and without points assigned.

In the Grid Editor view, you can assign components to areas using the **Add Column** and **Add Row** commands from the popup menu or from the Workbench main menu. You can edit existing rows or columns by double-clicking on them to display the **Edit Row** or **Edit Column** window, as shown here:

Figure 87 Adding rows and columns in the Grid Editor view



- Columns area**

This area extends across the top of the Grid Editor view. Columns specify the slot path that identifies the point data that goes in the table cell. Add columns *after* you have defined a template row in the template area.
- Template area**

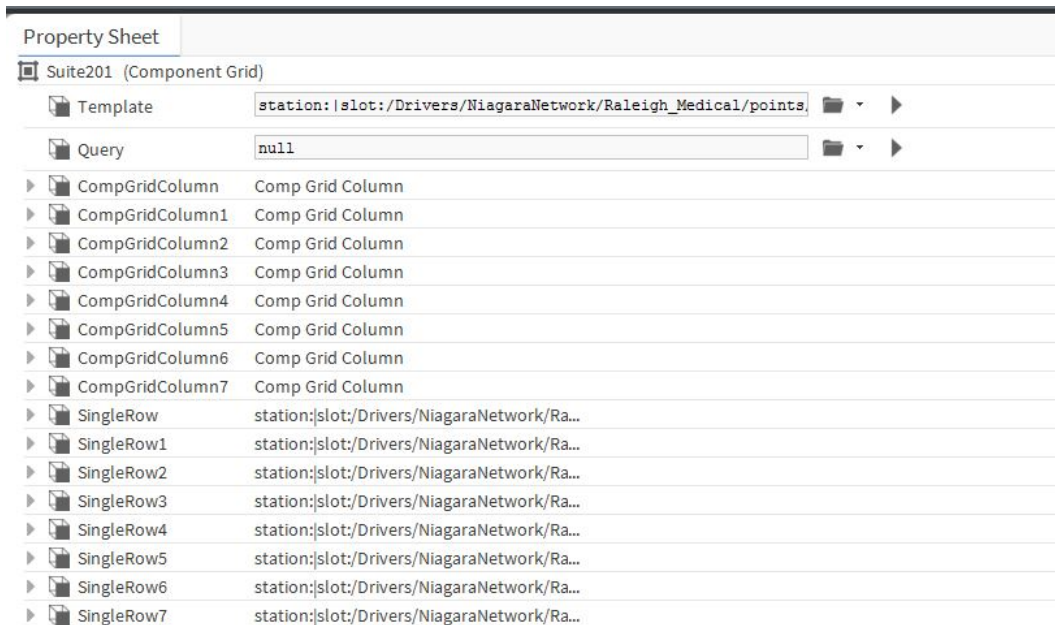
This area is just below the column area and displays as yellow, with a caution message when it is empty. The template row is used as a reference for picking columns, and therefore, there must be a template row defined before you can add and save a column in the Grid Editor view. Double-click on the template area to add a row using the **Edit Template** window.

NOTE: If you add a row without first adding a Template Row, then the first row you add becomes the Template Row by default and automatically populates the template area.
- Row area**

The row area extends across the lower part of the Grid Editor view. Add rows to define any point data that goes in the table row. Rows work in conjunction with the columns to present data in a table format.
- Bql Query**

This field allows you to enter a Bql query to resolve rows in the grid. This feature improves the performance of the BqlGrid component for resolving rows. It ensures reliable point subscription for those points defined by the BqlQuery.

Figure 88 Edit Query



Using the Grid Editor view

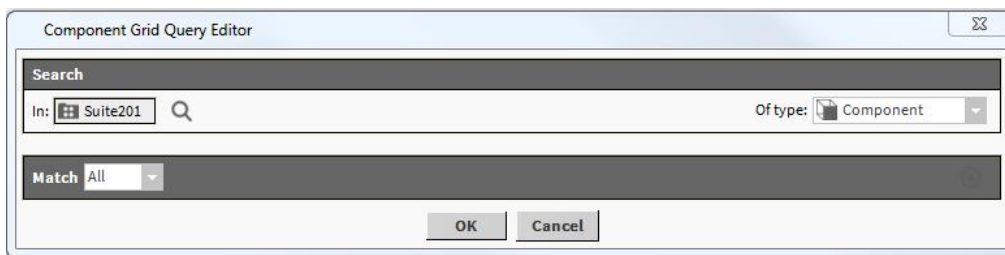
The typical workflow for using the Grid Editor view is:

- Add a template.
Drag a component from the Nav tree area onto the yellow Template Row.
- Add a row
You can double-click directly on the Empty Query row to use the **Component Grid Query Editor**.

NOTE:

Using a query (as opposed to dragging and dropping a component on the row) provides better point subscription service to points.

Figure 89 Component Grid Query Editor



- Add columns
Use the **Add Column** button on the main toolbar to add desired columns in the Grid Editor.

About editing rows/columns in a Component Grid

In the **Grid Editor** view, you can assign components to areas using the **Add Column** and **Add Row** commands from the popup menu or from the Workbench main menu. You can edit existing rows or columns by double-clicking on them to display the **Edit Row** or **Edit Column** window, as shown.

NOTE: Add columns after you have defined a template row in the template area.

Step 1 In the Grid Editor view, assign a component to a column or row:

- right-click in the Column Area and select **Add Column** from the popup menu.
- right-click in the Row Area and select **Add Row** from the popup menu.

Depending on your selection in step 1, either the **Edit Column** or **Edit Row** window displays.

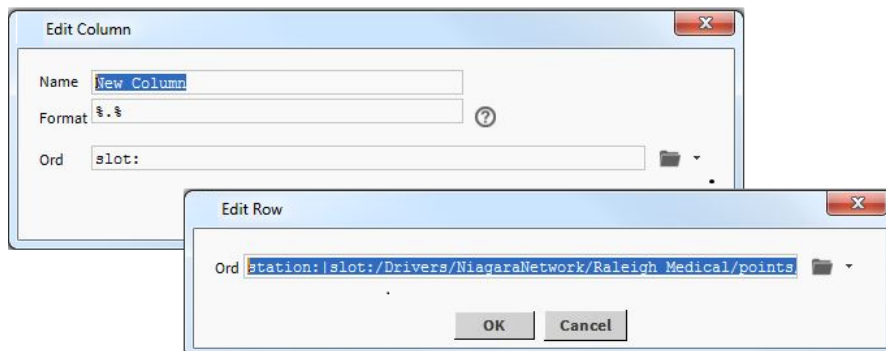
Step 2 If adding a column you can enter the following details in the **Edit Column** window:

- **Name**
Enter a name to assign and display name/title to the column
- **Format**
Enter literal text or BFormat scripting.
- **Ord**
Displays in **Edit Column** and **Edit Row** windows. Designate a point by browsing or entering the ord of the component that holds the value that you want to display.
NOTE: If assigning rows, you can drag and drop a point onto the Grid Editor view and the Ord automatically appears in the Ord field.

About the Report Edit Column/Edit Row dialog boxes

The **Edit Row** or **Edit Column** windows display when you add a new row or column—or when you choose to edit an existing row or column in the Grid Editor view.

Figure 90 Edit Row and Edit Column windows



The following fields are associated with these edit windows:

- **Name**
This field is associated with the **Edit Column** window. Type in a literal name to assign a display name or title to the column.
- **Format**
This field is associated with the **Edit Column** window. You can use literal text and BFormat scripting to assign a value to the column.
- **Ord**
This field displays in both the **Edit Column** window and the **Edit Row**.
Designate a point by browsing to or typing the Ord of the component that holds the value that you want to display.

NOTE: For assigning rows, you can drag and drop a point onto the Grid Editor view and the Ord automatically appears in the Ord field.

About Mobile Px App

You can use the Mobile Px app to create your own app views in the Px Editor. Use the standard Px Editor to design views with navigation and custom controls that display and work well in a handheld device.

IMPORTANT:

In Niagara 4.6 and later, legacy Niagara Mobile technology is deprecated. Although it continues to be supported and works as expected, you should not use it for any new projects. Instead, use the mobile-friendly features of the HTML5 Hx Profile which provides a significantly improved mobile user experience.

The Mobile Px app allows you to create your own app views in the Px Editor, without having to use any software programming language. This includes the ability to use the standard Px Editor to design views with navigation and custom controls that display and work well in a hand held device. Essentially, you can think of this app as being an app builder itself.

Figure 91 Mobile Px app in Workbench and Mobile Desktop view



Mobile Px provides a way for you to present live data and field editors in expandable lists that are designed to work well with mobile devices. It is not for absolutely positioning graphics or other media on the page, such as you would for designing a visual layout of a controls system. So, when you create views with this Mobile Px, the display is not an exact representation of what you have in the Px Editor or what you would expect to see when you are displaying views created in Hx. Instead, for Mobile Px views, the Px Editor works as a tool to create views that the Mobile Px app interprets and presents using such mobile-friendly features as: rounded buttons, liquid layout, dynamic views, and others.

However, many of the standard Px graphic widget and property options are still available for configuring your Px page, including:

- **Buttons**

Add standard buttons (Action Button, Hyperlink Button, Save Button) from the kitPx palette. These buttons work the same but appear rounded in mobile views.

- **Labels and bound labels**

Add standard labels from the kitPx palette.

- **Field editors**
- **Graphic alignment**
- **Font family**
- **Font size and style**

Chapter 10 Components, views and windows

Topics covered in this chapter

- ◆ Components in bajai module
- ◆ Components in chart module
- ◆ Components in gx module
- ◆ Components in kitPx module
- ◆ Components in pxeditor module
- ◆ Components in report module
- ◆ Plugins in pxEditor module
- ◆ Plugins in report module

The user interface includes components, views and windows, which provide the means for communicating with the system.

The Help topics include context sensitive information about each component and view, as well as information about individual windows.

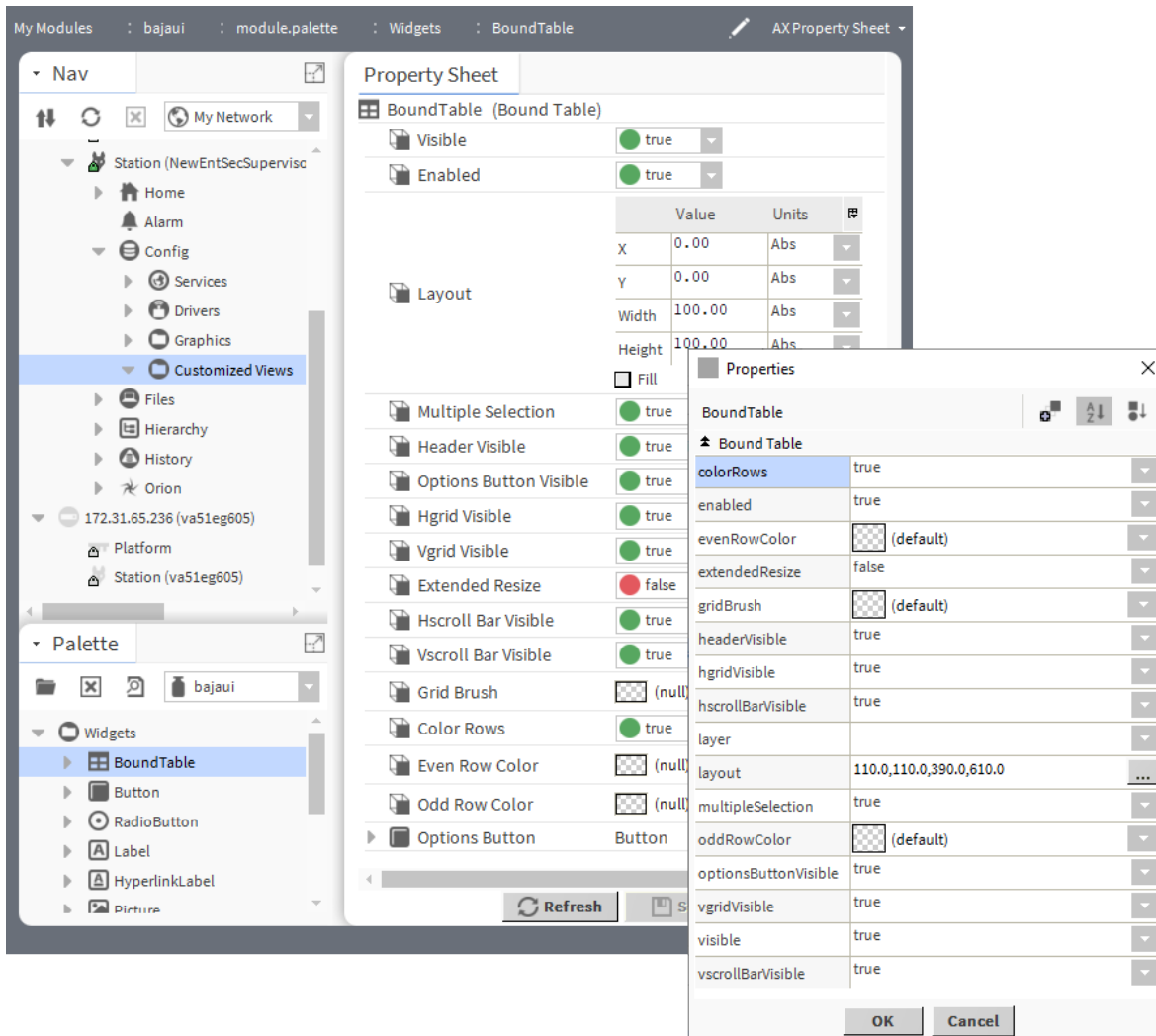
Components in bajai module

The following are components in the **bajai** module.

bajai-BoundTable




This component in the `bajai` palette, is a table implementation that is ready to use with `TableBindings` to populate its model.

Figure 92 BoundTable properties



You access these properties by expanding `bajau` palette and double-click **Bound Table**

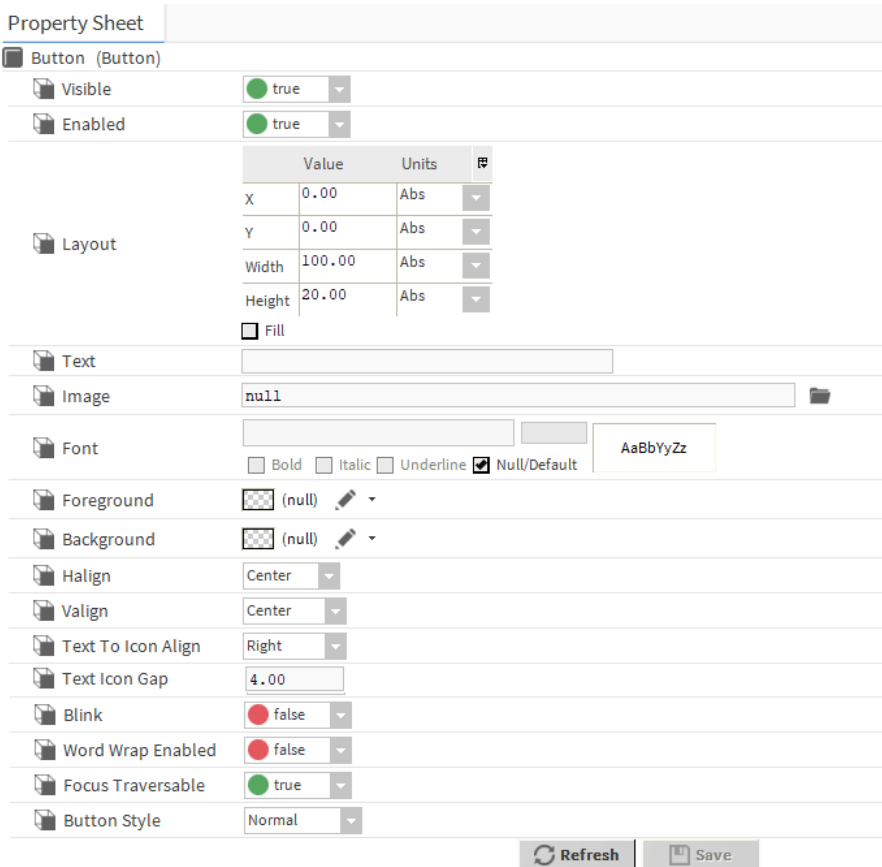
| Property | Value | Description |
|---------------------------------------|-------------------------|--|
| Visible, visible | true (default) false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled, enabled | true (default) false | Activates (true) and deactivates (false) the object (network, device, point, component, table, schedule, descriptor, etc.). When set to true, the table you access table properties in the Px page interface using the popup menu. When set to false, the table is visible but its properties cannot be configured. |
| Layout, layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Multiple Selection, multipleSelection | true (default) false | Permits (true) and restricts (false) the selection of multiple objects. |

| Property | Value | Description |
|--|-----------------------------------|--|
| Header Visible, headerVisible | true (default) false | Controls if the header is visible (true) or not (false). |
| Options Button Visible, optionsButtonVisible | true (default) false | Controls if the options button is visible (true) or not (false). |
| Hgrid Visible, hgridVisible | true (default) false | Controls if the horizontal grid is visible (true) or not (false). |
| Vgrid Visible, vgridVisible | true (default) false | Controls if the vertical grid is visible (true) or not (false). |
| Extended Resize, extendedResize | true false (default) | Permits (true) extending object size or not false. |
| HScroll Bar Visible, hscrollBarVisible | true (default) false | Makes the horizontal scroll bar visible (true and invisible (false). |
| VScroll Bar Visible, vscrollBarVisible | true (default) false | Makes the vertical scroll bar visible (true) and invisible (false). |
| Grid Brush, gridBrush | drop-down list (defaults to Null) | Selects the layout for the brush. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no grid brush. |
| Color Rows, colorRows | true (default) false | Adds color to the rows (true) or selects white for the rows (false, that is no color). |
| Even Row Color, evenRowColor | drop-down list (defaults to Null) | Selects the color for even rows. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no even row color. |
| Odd Row Color, oddRowColor | drop-down list (defaults to Null) | Selects the color for odd rows. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no row color. |
| Options Button | additional properties | "bajoui-Button" documents these properties. |

Button


Button, in the `bajoui` palette, is a control button which fires an action when clicked.

Figure 93 Button Property Sheet



You access these properties by expanding the `Bajoui` palette and double-clicking `Button`

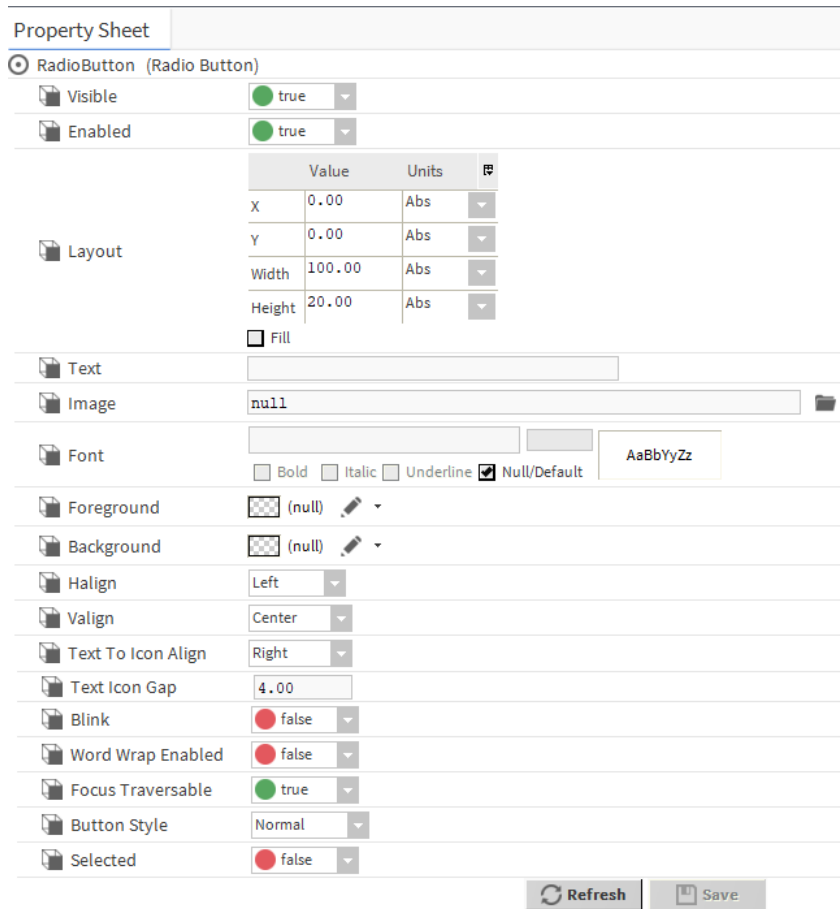
| Property | Value | Description |
|----------|--|---|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<code>true</code>) or not (<code>false</code>). |
| Enabled | true (default) false | Activates (<code>true</code>) and deactivates (<code>false</code>) the object (network, device, point, component, table, schedule, descriptor, etc.). When set to <code>true</code> , the table you access button properties in the Px page interface using the popup menu. When set to <code>false</code> , the table is visible but its properties cannot be configured. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Text | text | Specifies the text for the button. |
| Image | File Chooser (defaults to null—no image) | Selects the image file with the help of file chooser. |

| Property | Value | Description |
|--------------------|--|---|
| Font | drop-down lists for font and point size with check boxes for attributes (defaults to <code>Null/Default</code>) | Selects the font, point size and attributes for the button text. To configure, click to remove the <code>Null/Default</code> check box. |
| Foreground | drop-down list | Specifies foreground fill. <code>Solid</code> opens the color chooser window. <code>Gradient</code> opens the gradient editor window. <code>Image</code> opens the texture window, click the browser icon to open File chooser, <code>ord</code> chooser to select the image file. <code>Null</code> indicates no foreground fill. |
| Background | drop-down list | Specifies background fill color. <code>Solid</code> opens the Color Chooser window. <code>Gradient</code> opens the Gradient Editor window. <code>Image</code> opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. <code>Null</code> indicates no color (white). |
| Halign | drop-down list (defaults to <code>Center</code>) | Selects from among horizontal alignment options. |
| Valign | drop-down list (defaults to <code>Center</code>) | Selects from among vertical alignment options: <code>Top</code> , <code>Center</code> (default), <code>Right</code> , <code>Fill</code> |
| Text to Icon align | drop-down list (defaults to <code>Right</code>) | Defines the alignment of the text in relationship to the icon: <code>Right</code> , <code>Top</code> , <code>Bottom</code> , <code>Left</code> , <code>Center</code> |
| Text Icon Gap | number | Defines the gap between the text and icon. |
| Blink | <code>False</code> (default) <code>True</code> | Configures the background color to blink. |
| Word Wrap Enabled | <code>False</code> (default) <code>True</code> | Selects how to handle long words on buttons. <code>true</code> wraps the word on the button. <code>false</code> does not wrap. |
| Focus Traversable | <code>False True</code> (default) | Configures how this button responds when a user presses the Tab key in a Px view that contains the button. <code>true</code> includes this button in the set of Mouse Down Buttons that can sequentially receive focus when the user presses Tab . <code>false</code> disables focus for the button when the user presses Tab . |
| Button Style | drop-down list (defaults to <code>Normal</code>) | Selects the button style: <code>Normal</code> , <code>Tool Bar</code> , <code>Hyperlink</code> , <code>None</code> |

RadioButton


`RadioButton`, in the `bajauri` palette, is a specialized `ToggleButton` which displays its label next to a circle which can be checked and unchecked. It is used with groups of other `RadioButtons` to provide a choice that is exclusive of other options.

Figure 94 RadioButton Property Sheet



You access these properties by expanding **Bajoui** palette and double-click **RadioButton**

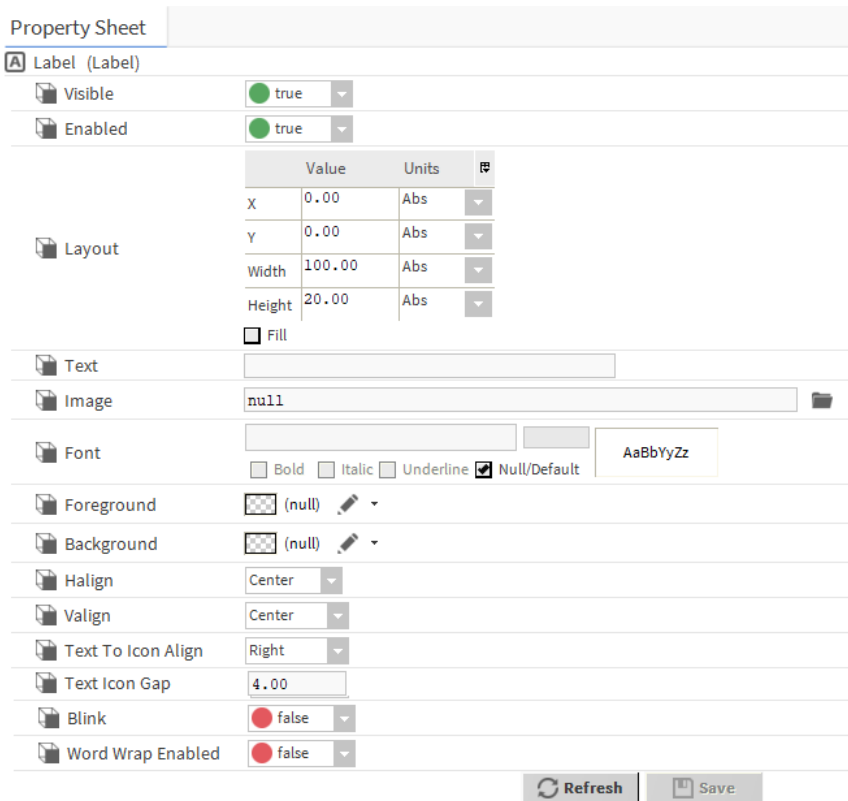
| Property | Value | Description |
|------------|-------------------------|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Text | text | Specify the text which should be there on the button. |
| Image | File chooser | Select the image file with the help of file chooser. |
| Font | null (default) | Default font is selected as Null. uncheck the Null checkbox and customise the font needed on the button. |
| Foreground | drop-down list | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, or d chooser to select the image file. Null indicates no foreground fill. |

| Property | Value | Description |
|--------------------|--|---|
| Background | drop-down list | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |
| Halign | drop-down list | Selects from among horizontal alignment options. |
| Valign | Top, Center (default), Right, Fill | Specifies to vertically align the pane. |
| Text to Icon align | Right (default), Top, Bottom, Left, Center | Defines the alignment of the text in relationship to the icon: Right, Top, Bottom, Left, Center |
| Text Icon Gap | Number | Defines the gap between the text and icon. |
| Blink | False (default) True | Configures the background color to blink. |
| Word Wrap Enabled | False (default) True | When set to <code>true</code> allows the word on the button to be wrapped. When set to <code>false</code> does not allows the word on the button to be wrapped. |
| Focus Traversable | False True (default) | |
| Button Style | Normal(default), Tool Bar, Hyperlink, None | Selects the button style: Normal, Tool Bar, Hyperlink, None |
| Selected | False (default) True | |

Label


Label, in the `bajoui` palette, is used to display text and/or an icon. Labels are always transparent, so their background is defined by their parent. Their preferred size is always an exact fit to contain the text and/or icon. Labels may be used by themselves to display information which cannot respond to user input, or they may be embedded in widgets such as Buttons.

Figure 95 Label Property Sheet



You access these properties by expanding **Bajauri** palette and double-click **Label**

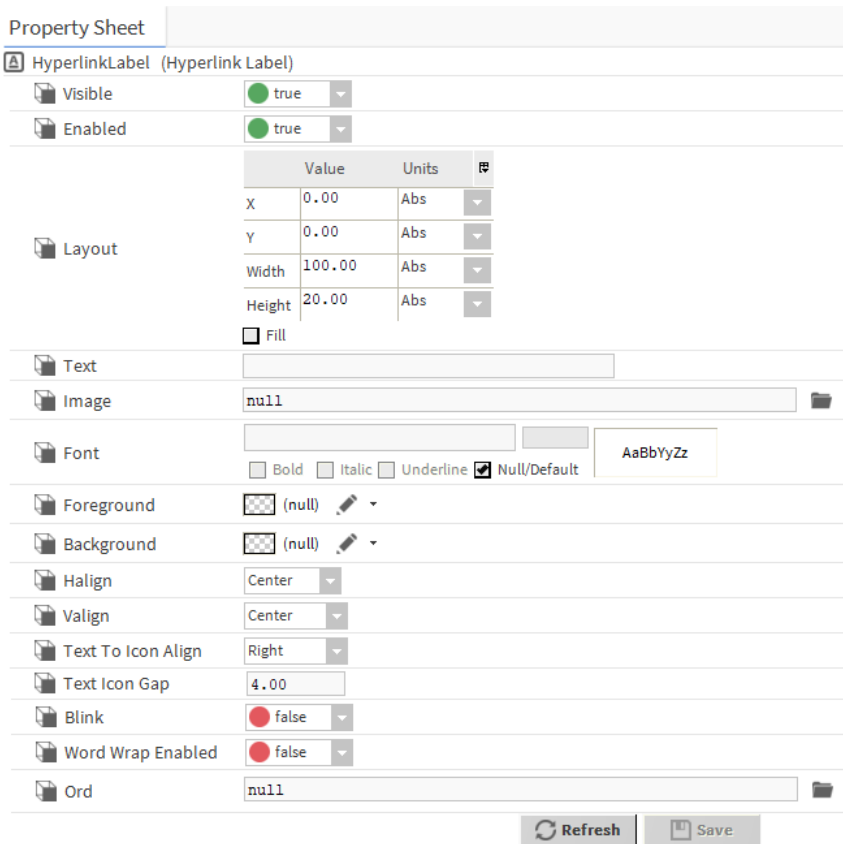
| Property | Value | Description |
|------------|-------------------------|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Text | text | Specify the text which should be there on the button. |
| Image | File chooser | Select the image file with the help of file chooser. |
| Font | null (default) | Default font is selected as Null. uncheck the Null checkbox and customise the font needed on the button. |
| Foreground | drop-down list | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, or d chooser to select the image file. Null indicates no foreground fill. |
| Background | drop-down list | Specifies background fill color. Solid opens the Color Chooser window. |

| Property | Value | Description |
|--------------------|--|---|
| | | <p>Gradient opens the Gradient Editor window.</p> <p>Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file.</p> <p>Null indicates no color (white).</p> |
| Halign | drop-down list | Selects from among horizontal alignment options. |
| Valign | Top, Center (default), Right, Fill | Specifies to vertically align the pane. |
| Text to Icon align | Right (default), Top, Bottom, Left, Center | Defines the alignment of the text in relationship to the icon: Right, Top, Bottom, Left, Center |
| Text Icon Gap | Number | Defines the gap between the text and icon. |
| Blink | False (default) True | Configures the background color to blink. |
| Word Wrap Enabled | False (default) True | When set to <code>true</code> allows the word on the button to be wrapped. When set to <code>false</code> does not allows the word on the button to be wrapped. |

HyperlinkLabel


The `HyperlinkLabel`, in the `bajoui` palette, has the same properties as the `Label` (see `bajoui-Label`), plus an `ORD` property that allows you to assign an `ORD` to the label. When an `ORD` path is supplied for this property, the `HyperlinkLabel` causes a mouse cursor to change to a standard link cursor and the component performs a hyperlink when clicked.

Figure 96 HyperLinkLabel Property Sheet



You access these properties by expanding **Bajoui** palette and double-click **HyperLinkLabel**

| Property | Value | Description |
|------------|-------------------------|---|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<code>true</code>) or not (<code>false</code>). |
| Enabled | true (default) false | When set to <code>true</code> , the table in Px page interface is commandable using the popup menu. When set to <code>false</code> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Text | text | Specify the text which should be there on the button. |
| Image | File chooser | Select the image file with the help of file chooser. |
| Font | null (default) | Default font is selected as Null. uncheck the Null checkbox and customise the font needed on the button. |
| Foreground | drop-down list | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no foreground fill. |
| Background | drop-down list | Specifies background fill color. |

| Property | Value | Description |
|--------------------|--|--|
| | | <p>Solid opens the Color Chooser window.</p> <p>Gradient opens the Gradient Editor window.</p> <p>Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file.</p> <p>Null indicates no color (white).</p> |
| Halign | drop-down list | Selects from among horizontal alignment options. |
| Valign | Top, Center (default), Right, Fill | Specifies to vertically align the pane. |
| Text to Icon align | Right (default), Top, Bottom, Left, Center | Defines the alignment of the text in relationship to the icon: Right, Top, Bottom, Left, Center |
| Text Icon Gap | Number | Defines the gap between the text and icon. |
| Blink | False (default) True | Configures the background color to blink. |
| Word Wrap Enabled | False (default) True | When set to <code>true</code> allows the word on the button to be wrapped. When set to <code>false</code> does not allows the word on the button to be wrapped. |
| Focus Traversable | False True (default) | |
| Button Style | Normal(default), Tool Bar, Hyperlink, None | Selects the button style: Normal, Tool Bar, Hyperlink, None |

Picture

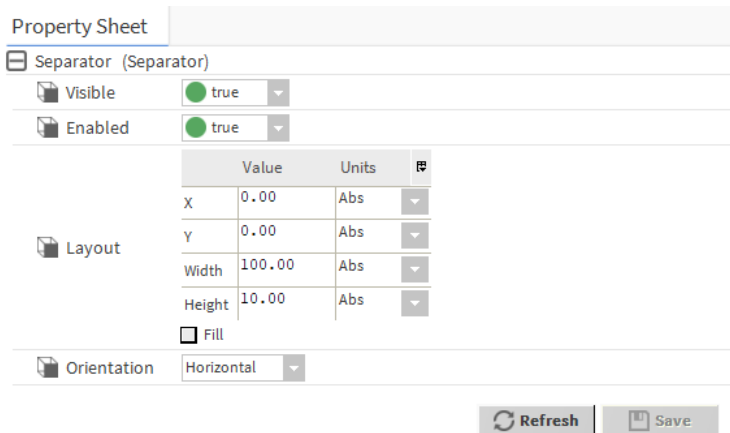
bajoui-Picture

The Picture widget, in the `bajoui` palette provides scaling functionality. A `scaleMode` property allows you to stretch and skew an image to fit within the Picture widget's borders. A Picture can be backed with any image, including JPGs and PNGs, but you will get the most benefit out of Picture scaling when you use SVGs images.

Separator

Separator, in the `bajoui` palette, is used in ToolBars and Menus to provide a visible separator between groups of buttons.

Figure 97 Seperator Property Sheet



You access these properties by expanding **Bajai** palette and double-click **Seperator**

| Property | Value | Description |
|-------------|----------------------------------|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<i>true</i>) or not (<i>false</i>). |
| Enabled | true (default) false | When set to <i>true</i> , the table in Px page interface is commandable using the popup menu. When set to <i>false</i> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Orientation | Horizontal (default) Vertical | Specifies the orientation of the pane. |

PxInclude

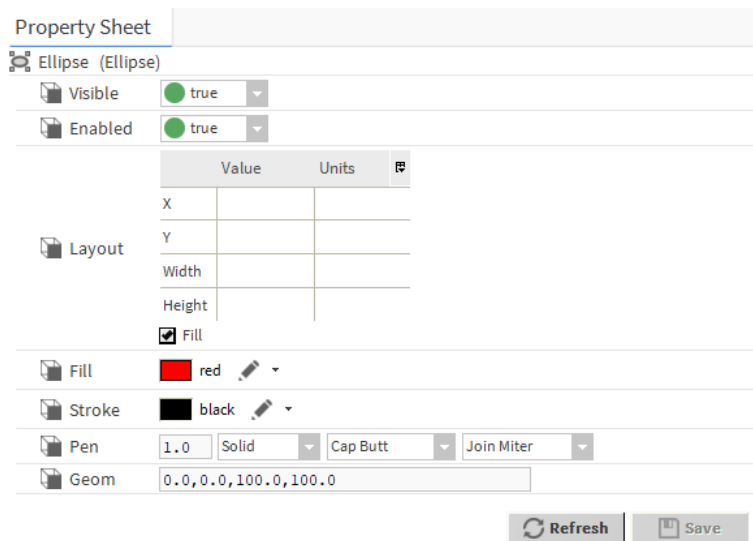
bajai-PxInclude

This widget provides a way for users to embed a single Px file in another Px file. The PxInclude widget is located in the **bajai** palette.

Ellipse

Ellipse, in the **bajai** palette, renders a ellipse geometry.

Figure 98 Ellipse Property Sheet



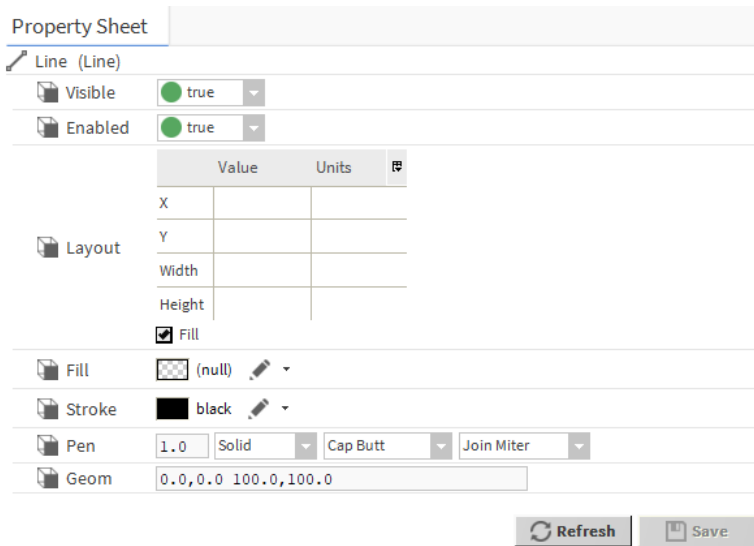
You access these properties by expanding **Bajauri** palette and navigate to **Shapes**, double-click **Ellipse**

| Property | Value | Description |
|----------|--|---|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<code>true</code>) or not (<code>false</code>). |
| Enabled | true (default) false | When set to <code>true</code> , the table in Px page interface is commandable using the popup menu. When set to <code>false</code> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Fill | Solid, Gradient, Image, Null (default) | Specifies color fill of the pane . Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no background fill. |
| Stroke | Solid, Gradient, Image, Null (default) | Specifies color fill of the pane . Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no background fill. |
| Pen | Additional Properties | Allows to configure additional properties. |
| Goem | number | |

Line

Line, in the `bajauri` palette, renders a line between two points.

Figure 99 Line Property Sheet



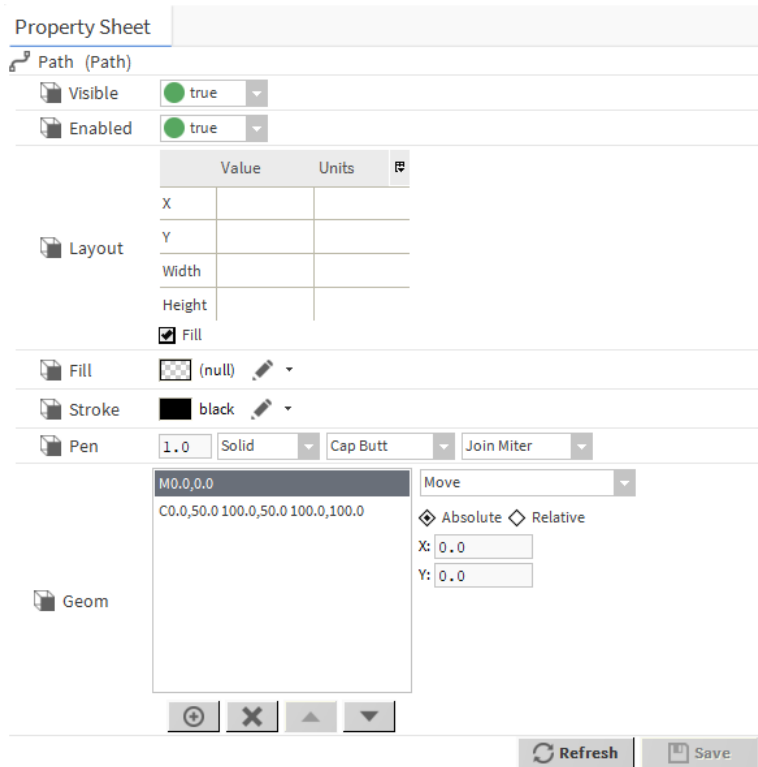
You access these properties by expanding **Bajauri** palette and navigate to **Shapes**, double-click **Line**

| Property | Value | Description |
|----------|--|---|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Fill | Solid, Gradient, Image, Null (default) | Specifies color fill of the pane . Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no background fill. |
| Stroke | Solid, Gradient, Image, Null (default) | Specifies color fill of the pane . Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no background fill. |
| Pen | Additional Properties | Allows to configure additional properties. |
| Goem | number | |

Path

Defines a general path to draw or fill, using a model and string format based on the SVG path element. This component is in the `bajauri` palette.

Figure 100 Path Property Sheet



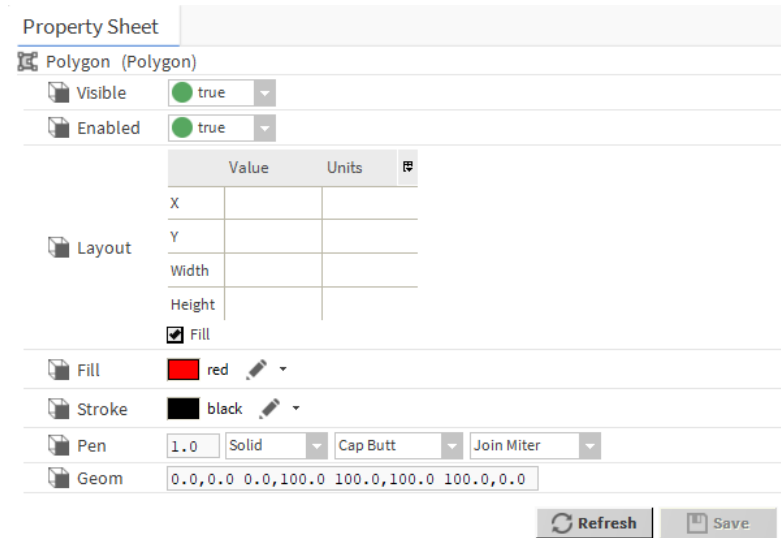
You access these properties by expanding **Bajoui** palette and navigate to **Shapes**, double-click **Path**

| Property | Value | Description |
|----------|--|---|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<code>true</code>) or not (<code>false</code>). |
| Enabled | true (default) false | When set to <code>true</code> , the table in Px page interface is commandable using the popup menu. When set to <code>false</code> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Fill | Solid, Gradient, Image, Null (default) | Specifies color fill of the pane . Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no background fill. |
| Stroke | Solid, Gradient, Image, Null (default) | Specifies color fill of the pane . Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no background fill. |
| Pen | Additional Properties | Allows to configure additional properties. |
| Goem | number | |

Polygon

Models a closed area defined by a series of line segments, using string format x,y,width,height. This component is in the **bajai** palette.

Figure 101 Polygon Property Sheet



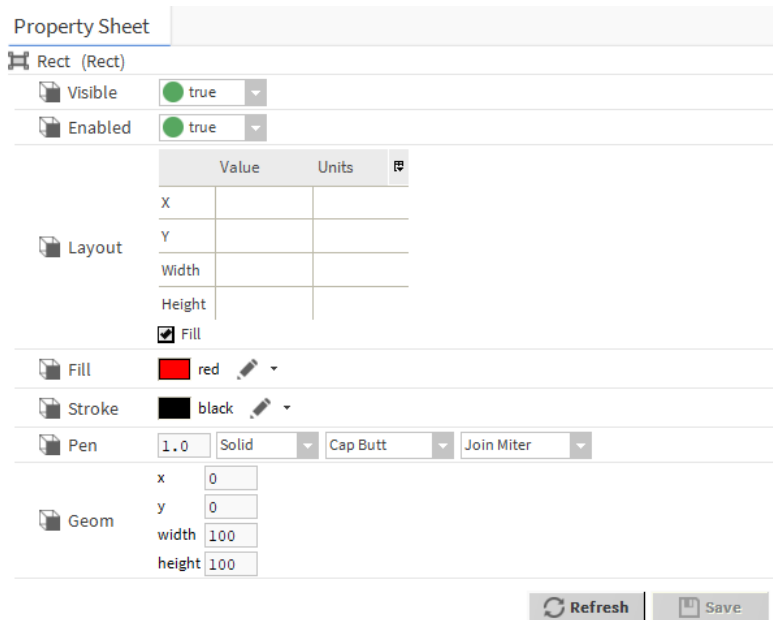
You access these properties by expanding **Bajai** palette and navigate to **Shapes**, double-click **Polygon**

| Property | Value | Description |
|----------|--|---|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<i>true</i>) or not (<i>false</i>). |
| Enabled | true (default) false | When set to <i>true</i> , the table in Px page interface is commandable using the popup menu. When set to <i>false</i> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Fill | Solid, Gradient, Image, Null (default) | Specifies color fill of the pane . Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no background fill. |
| Stroke | Solid, Gradient, Image, Null (default) | Specifies color fill of the pane . Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no background fill. |
| Pen | Additional Properties | Allows to configure additional properties. |
| Goem | number | |

Rect

Rect, in the `bajau` palette, renders a rectangle shape.

Figure 102 Rect Property Sheet



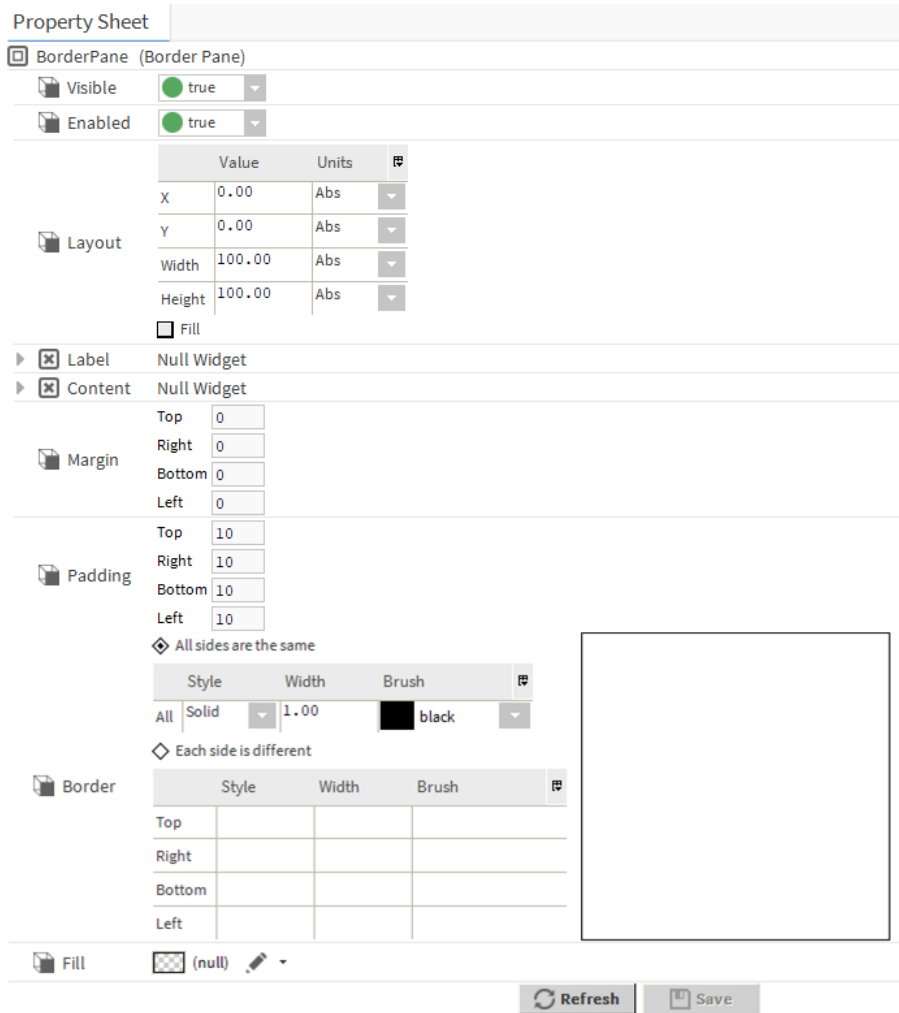
You access these properties by expanding `Bajau` palette and navigate to **Shapes**, double-click **Rect**

| Property | Value | Description |
|----------|--|---|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<code>true</code>) or not (<code>false</code>). |
| Enabled | true (default) false | When set to <code>true</code> , the table in Px page interface is commandable using the popup menu. When set to <code>false</code> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Fill | Solid, Gradient, Image, Null (default) | Specifies color fill of the pane . Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no background fill. |
| Stroke | Solid, Gradient, Image, Null (default) | Specifies color fill of the pane . Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no background fill. |
| Pen | Additional Properties | Allows to configure additional properties. |
| Goem | number | |

BorderPane

BorderPane, in the `bajoui` palette, provides border content for a widget similar to the CSS Box Model. Margin defines the space between the bounds and the border. Padding defines the space between the border and the child bounds. Border defines the look and width of the border around each of the sides.

Figure 103 BorderPane Property Sheet



You access these properties by expanding `Bajoui` palette and navigate to `Panes`, double-click `BorderPane`

| Property | Value | Description |
|----------|-------------------------|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<code>true</code>) or not (<code>false</code>). |
| Enabled | true (default) false | When set to <code>true</code> , the table in Px page interface is commandable using the popup menu. When set to <code>false</code> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Label | additional properties | Allows to configure additional properties for the label of the border pane. |

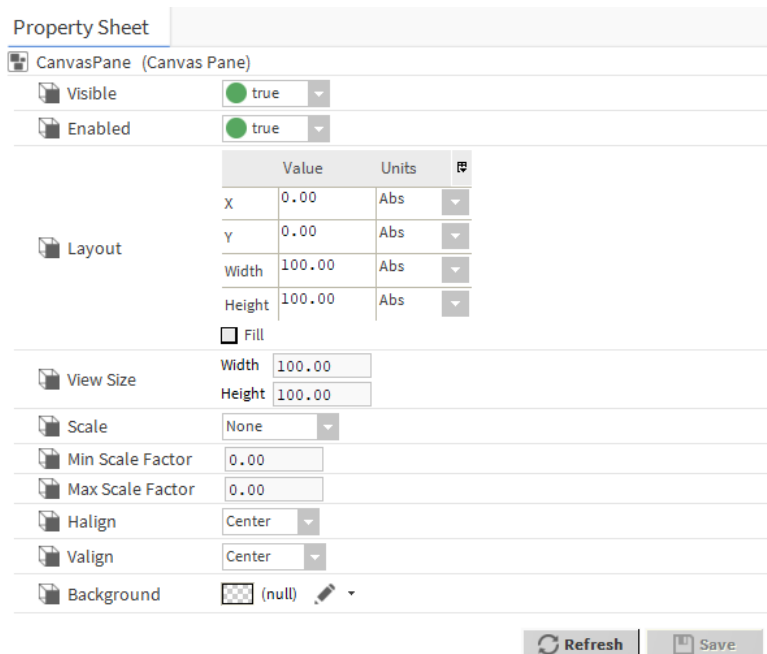
| Property | Value | Description |
|----------|--|---|
| Content | additional properties | Allows to configure additional properties for the label of the border pane. |
| Margin | top Right Bot- tom Left | Allows to set the borders of the pane. |
| Padding | top Right Bot- tom Left | Allows to set the padding of the pane. |
| Border | text | Displays the border design. |
| Fill | Solid, Gradient, Image ,Null (default) | Specifies color fill of the pane . Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no background fill. |

CanvasPane


Found, in the `ba_jaui` palette, this component is commonly used to specify positioning for most Widgets and Geom for Shapes in a Px graphic. In addition to providing support for absolute positioning, it also provides support for scaling and/or aligning its contents.

The added Min/Max Scale Factor properties in Niagara 4.6 and later, provide support for responsive positioning, giving you more control over how graphics respond when displayed on mobile devices. The responsive CanvasPane scales contents to the screen size of the device in use, which could be anything from a PC with a large screen monitor to a cellphone. Additionally, the FlowPane component (with child Responsive-Pane) adjusts the display of contents in a CanvasPane to make best use of the available space. For example when using a cellphone, data that initially displays in a single column adjusts to flow into two columns when the device is rotated 90–degrees.

Figure 104 CanvasPane properties



NOTE: The CanvasPane component also is available in the `mobile` palette (`mobile` palette is deprecated in Niagara 4.6 and later). It provides support for absolute positioning for Widgets in a Mobile Px page. When used in a Mobile Px file, the CanvasPane `scale` options are limited to `None` or `Fit Ratio`.

| Type | Value | Description |
|------------------|---|---|
| Visible | <code>true</code> (default), <code>false</code> | When true the object is visible. When false the object is hidden. |
| Enabled | <code>true</code> or <code>false</code> | Activates (<code>true</code>) and deactivates (<code>false</code>) the object (network, device, point, component, table, schedule, descriptor, etc.). |
| Layout | additional properties | See following section, Layout properties |
| View Size | Width 100.00, Height 100.00 | Specifies the logical size of the CanvasPane's coordinate system, measured as a percentage. |
| Scale | <code>None</code> (default), <code>Fit</code> , <code>Fit Ratio</code> , <code>Fit Width</code> , <code>Fit Height</code> | Specifies how to scale the ViewSize to fit the CanvasPane's bounds: <code>Fit Ratio</code> preserves the aspect ratio. <code>Fit</code> causes the graphics to fill the entire available screen space, possibly causing some distortion. |
| Min Scale Factor | 0.05 (default) | Defines the minimum scale for the graphic. the range is 0.0–1.0. Setting a value smaller than 0.5 could allow a graphic to become unusable and setting a value to 1.0 does not allow it to shrink at all. |
| Max Scale Factor | 1.00 (default) | Defines the maximum scale for a graphic. The default means the graphic will not get any bigger than it currently is. In most cases, you want to shrink images for mobile device display and you do not want to enlarge them when the page gets very big. |
| Halign | <code>Left</code> , <code>Center</code> (default), <code>Right</code> , <code>Fill</code> | Defines how to horizontally align the CanvasPane. |
| Valign | <code>Top</code> , <code>Center</code> (default), <code>Bottom</code> , <code>Fill</code> | Defines how to vertically align the CanvasPane. |
| Background | <code>Solid</code> , <code>Gradient</code> , <code>Image</code> , <code>Null</code> (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |

Layout properties

The Layout property, visible in a **Property Sheet** view of the component, allows you to set positioning coordinates and dimensions for the pane.

| Name | Value | Description |
|---------------|----------|---|
| X | 0.00 | Provides the horizontal X-coordinate for absolute positioning of the pane. Units are either Abs (default) or Percent. |
| Y | 0.00 | Provides the vertical Y-coordinate for absolute positioning of the pane. Units are either Abs (default) or Percent. |
| Width | 100.00 | Provides the width dimension of the pane in pixels. Units specify either an absolute value, a percentage, or a preferred value: Abs (default) or Percent, or Pref. |
| Height | 100.00 | Provides the height dimension of the pane in pixels. Units specify either an absolute value, a percentage, or a preferred value: Abs (default) or Percent, or Pref. |
| Fill | checkbox | When Fill is selected all widgets in a given row are sized to fill the row height. When deselected the widgets in a given row use their preferred height and alignment. |

ConstrainedPane

ConstrainedPane, in the `bajoui` palette, is a pane with a constrained size. It allows the pane to be restricted by minimum width, minimum height, and maximum height.

Figure 105 ConstrainedPane Property Sheet

The screenshot shows the 'Property Sheet' for a 'ConstrainedPane (Constrained Pane)'. The properties are organized into sections:

- Visible:** true (dropdown)
- Enabled:** true (dropdown)
- Layout:** A table with columns 'Value' and 'Units'.

| | Value | Units |
|--------|-------|-------|
| X | | |
| Y | | |
| Width | | |
| Height | | |
- Fill:**
- Content:** Null Widget
- Min Size:** Width: 0.00, Height: 0.00
- Max Size:** Width: 2147483647, Height: 2147483647

At the bottom, there are 'Refresh' and 'Save' buttons.

You access these properties by expanding `Bajoui` palette and navigate to **Panes**, double-click **ConstrainedPane**

| Type | Value | Description |
|---------|-----------------------|---|
| Visible | true (default), false | When true the object is visible. When false the object is hidden. |
| Enabled | true or false | Activates (<code>true</code>) and deactivates (<code>false</code>) the object (network, device, point, component, table, schedule, descriptor, etc.). |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |

| Type | Value | Description |
|------------------|-----------------------|--|
| Min Scale Factor | 0.05 (default) | Range is 0.0–1.0. For a new graphic, the default value is 0.5. Setting a value smaller than 0.5 could allow a graphic to become unusable and setting a value to 1.0 does not allow it to shrink at all. |
| Max Scale Factor | 1.00 (default) | The default value for this property is 1.00. This means the graphic won't get any bigger than it currently is. In most cases, you want to shrink images for mobile device display and you do not want to enlarge them when the page gets very big. |
| Content | additional properties | Allows to configure additional parameters for the content of the pane. |

EdgePane

bajoui-EdgePane

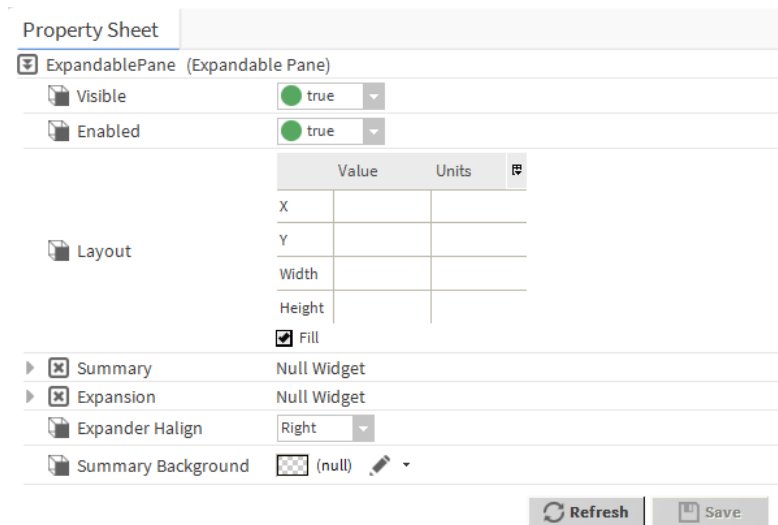
EdgePane, in the `bajoui` palette, is a container with a layout like the `java.awt.BorderLayout`. It only supports five potential children in the frozen slots top, bottom, left, right, and center. The top and bottom widgets fill the pane horizontally use their preferred height. The left and right widgets use their preferred width and occupy the vertical space between the top and bottom. The center widget gets all the remaining space. Any of the widgets may be a `NullWidget`.

ExpandablePane


ExpandablePane, in the `bajoui` palette, contains two widgets. A summary widget is displayed all the time, with a button to the right used to expand and collapse the pane. When expanded, the expansion widget is displayed under the summary.

Niagara 4.10 and later supports applying style to the widget and nesting of the panes.

Figure 106 ExpandablePane Property Sheet



You access these properties by expanding `Bajoui` palette and navigate to `Panes` and double-click `ExpandablePane`

| Property | Value | Description |
|--------------------|---------------------------------|---|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<code>true</code>) or not (<code>false</code>). |
| Enabled | true (default) false | When set to <code>true</code> , the table in Px page interface is commandable using the popup menu. When set to <code>false</code> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Summary | additional properties | Configures additional parameters (Visible, Enable, Layout) for expandable pane summary. |
| Expansion | additional properties | Configures additional parameters (Visible, Enable, Layout) for expandable pane expansion. |
| Expander Halign | drop-down list | Selects from among horizontal alignment options. |
| Summary background | Solid, Gradient, Image and Null | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |

bajoui-FlowPane

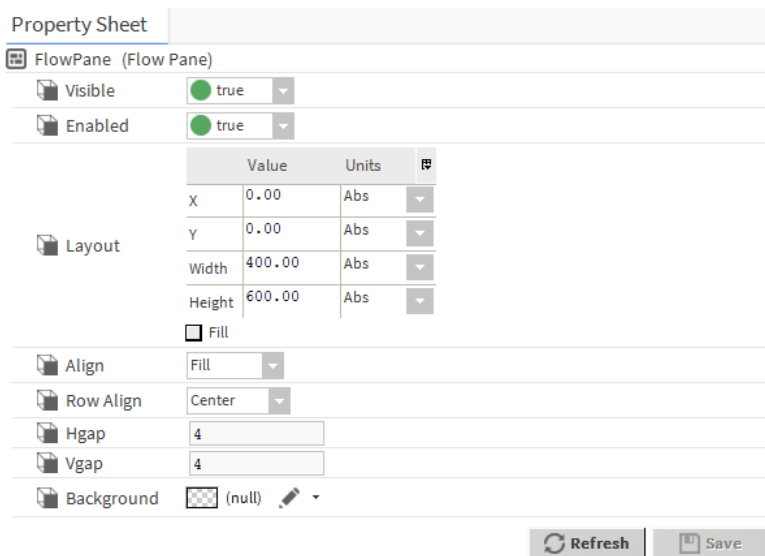
One of the mobile-friendly components available in the `bajoui` palette Niagara 4.10 (and later) that adapts the display of Px graphics for various devices (PC, cellphone, tablet, etc.). FlowPane will layout its children from left to right and top to bottom, fitting as many children as possible in each row.


For best results, make a ScrollPane the root of your Px page and then add a FlowPane to the ScrollPane's Content property.

NOTE: The `bajoui` palette provides an `Examples` folder that contains pre-configured FlowPanes and CanvasPanes to help you start designing mobile-friendly px pages.

NOTE: A FlowPane's preferred size is not responsive so it may not layout correctly if its parent is an EdgePane.

Figure 107 Example FlowPane Properties Pane



| Name | Value | Description |
|------------|---|---|
| Visible | true (default) or false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) or false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |
| Align | Left, Center, Right, or Fill | Provides options for setting the alignment of FlowPane child widgets. The Fill option expands any child ResponsivePane's set to a percentage to fill the row. |
| Row Align | Top, Center, Bottom, or Fill | Provides options for setting the alignment of row content (top, bottom, center, and fill). |
| Hgap | 4 pixels (default) | Specifies the horizontal gap between child widgets in a row. |
| Vgap | 4 pixels (default) | Specifies the vertical gap between rows. |
| Background | Solid, Gradient, Image, or Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |

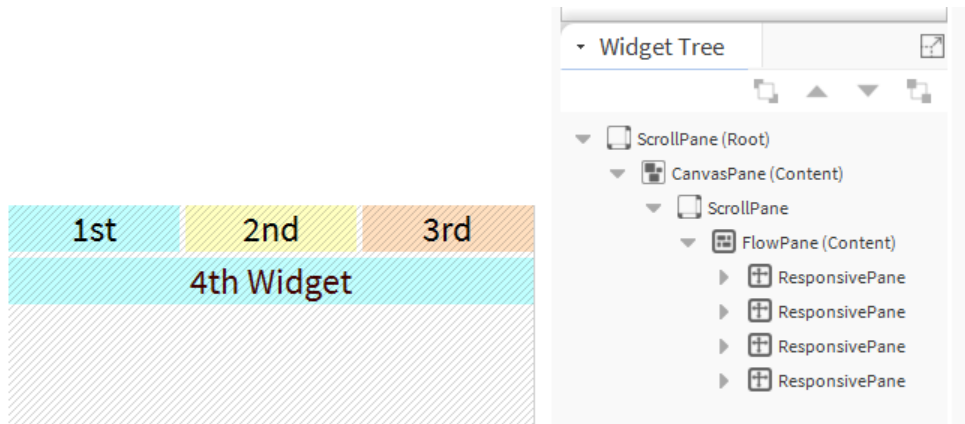
ResponsivePane

One of the mobile-friendly components available in the **ba_jau_i** palette (Niagara 4.6 and later) that adapts the display of Px graphics for various devices (PC, cellphone, tablet, etc.). ResponsivePane can be used to customize the layout of widgets within a FlowPane.

For example, if you add 4 ResponsivePanes to a FlowPane and set the Max Width property for each ResponsivePane to 25%. The ResponsivePane will adjust the width of each to try to fit the 4 children on the same

row. The last child ResponsivePane will adjust in size from its Max Width to Min Width value to try to fit on the row, but if there is not enough space for the specified Min Width, the FlowPane will let that child wrap to the next row.

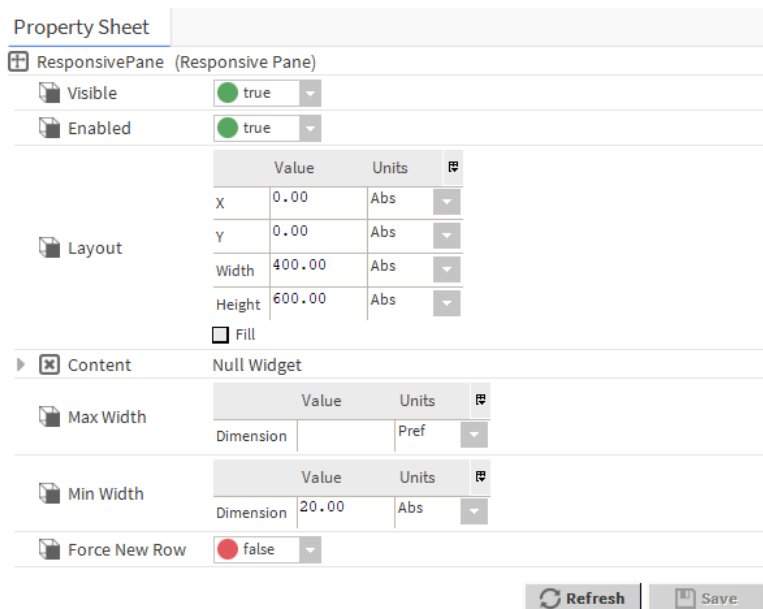
Figure 108 Responsive Panes in PxEditor and Widget Tree Pane



NOTE: The `bajoui` palette provides an `Examples` folder that contains pre-configured FlowPanes and CanvasPanes to help you start designing mobile-friendly px pages.

ResponsivePane is available in the `bajoui` palette.

Figure 109 Example ResponsivePane Properties Pane



| Property | Value | Description |
|----------|-------------------------|--|
| Visible | true (default) or false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) or false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |
| Layout | additional properties | See following section, Layout properties. |

| Property | Value | Description |
|---------------|-------------------------|---|
| Content | additional properties | The Content property is a container for an added widget. It reflects properties of the widget that is added to the pane. |
| Max Width | value, and units | Specifies the maximum width of the ResponsivePane. Max Width dimension defaults to the preferred size of the widget inserted in Content, but can be changed for better flow layout. Unit type can be Absolute, Percent, or Preferred. |
| Min Width | value, and units | Specifies the minimum width of the ResponsivePane. Min Width dimension defaults to 20.0 (pixels) Abs, but should be customized to the smallest desired width for the widget inserted in Content. Unit type can be Absolute (pixels), Percent, or Preferred. |
| Force New Row | true or false (default) | Forces the ResponsivePane to wrap to the next row in the parent container. This ensures a widget starts on a new row, which allows setting up different sections like header, footer, etc., in a FlowPane |

Layout properties

The Layout property, visible in a **Property Sheet** view of the component, allows you to set positioning coordinates and dimensions for the pane.

| Name | Value | Description |
|--------|----------|---|
| X | 0.00 | Provides the horizontal X-coordinate for absolute positioning of the pane. Units are either Abs (default) or Percent. |
| Y | 0.00 | Provides the vertical Y-coordinate for absolute positioning of the pane. Units are either Abs (default) or Percent. |
| Width | 100.00 | Provides the width dimension of the pane in pixels. Units specify either an absolute value, a percentage, or a preferred value: Abs (default) or Percent, or Pref. |
| Height | 100.00 | Provides the height dimension of the pane in pixels. Units specify either an absolute value, a percentage, or a preferred value: Abs (default) or Percent, or Pref. |
| Fill | checkbox | When Fill is selected all widgets in a given row are sized to fill the row height. When deselected the widgets in a given row use their preferred height and alignment. |

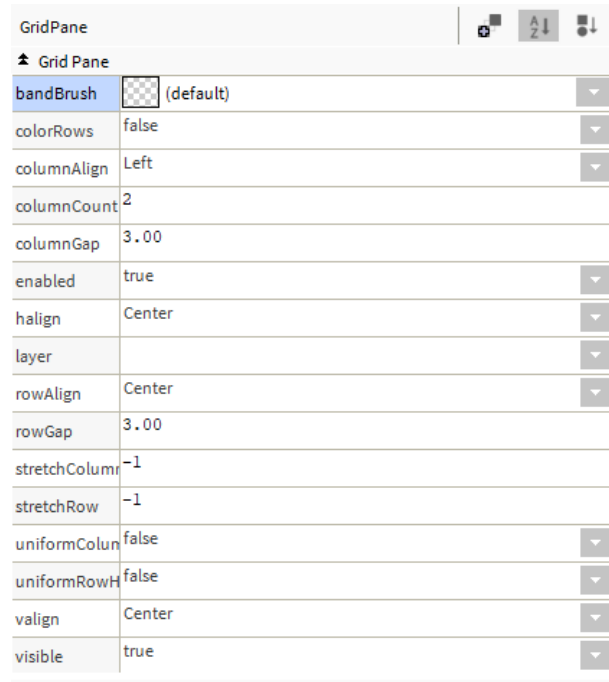
GridPane

GridPane, in the `bajau` palette, provides a flexible layout based on a grid with a predefined number of columns. GridPane will layout its children as a series of columns and rows. You can configure extra space in the rows and columns a number of ways using the pane's many properties.

Cells are laid out left to right, flowing to the next row based on the `columnCount` property. Row height is determined by the max preferred height of the widgets in the row. Column width is determined by the max preferred width of the widgets in the column. If Layout (visible via Property Sheet view) is set to Fill then all widgets in a given row are sized to fill the row height. Otherwise the widgets use their preferred height and are aligned accordingly

By default if the actual space of the pane is bigger than the preferred size, then the `hAlign` and `vAlign` properties determine where to put the extra space. Or the `stretchColumn` and `stretchRow` properties may be

used to indicate that a specific column or row should be stretched to fill the additional space. Enabling the stretch feature trumps pane alignment.



| Name | Value | Description |
|---------------|---|--|
| bandBrush | Solid, Gradient, Image, or Null (default) | Sets row color. Selecting either solid, gradient, or image opens a Chooser window. |
| colorRows | true or false (default) | When set to true, alternating rows are shaded to produced a striped effect. |
| columnCount | 2 (default) | Sets the number of columns (i.e. cells per row). |
| columnGap | 3.00 pixels | Sets the gap between column (i.e. cells in a row). |
| enabled | true (default) or false | Activates and deactivates use of the component. |
| halign | Left, Center (default), Right, or Fill | Sets horizontal alignment for the pane. |
| layer | | Allows you to add/change/remove a named layer assignment for the pane. Assigning a layer requires one or more previously configured named layers. |
| rowAlign | Top, Center (default), Bottom, or Fill | Sets vertical alignment for cells (i.e. cell contents) in a row. |
| rowGap | 3.00 pixels | Sets space between rows. |
| stretchColumn | -1 | The stretchColumn property may be used to indicate that a specific column should be stretched to fill the additional space. Enabling the stretch feature overrides |

| Name | Value | Description |
|----------------------|--|--|
| stretchRow | -1 | The stretchRow property may be used to indicate that a specific row should be stretched to fill the additional space. |
| uniformColumn-Width | true or false (default) | Column width is determined by the max preferred width of the widgets in the column. If uniformColumnWidth is true then all column widths are sized using the max column. |
| uniformColumn-Height | true or false (default) | Row height is determined by the max preferred height of the widgets in the row. If uniformRowHeight is true then all row heights are sized using the max row. |
| valign | Top, Center (default), Bottom, or Fill | Sets vertical alignment for the pane. |
| visible | true (default) or false | When true the object is visible. When false the object is hidden. |

Layout properties

The Layout property, visible in a **Property Sheet** view of the component, allows you to set positioning coordinates and dimensions for the pane.

| Name | Value | Description |
|--------|----------|---|
| X | 0.00 | Provides the horizontal X-coordinate for absolute positioning of the pane. Units are either Abs (default) or Percent. |
| Y | 0.00 | Provides the vertical Y-coordinate for absolute positioning of the pane. Units are either Abs (default) or Percent. |
| Width | 100.00 | Provides the width dimension of the pane in pixels. Units specify either an absolute value, a percentage, or a preferred value: Abs (default) or Percent, or Pref. |
| Height | 100.00 | Provides the height dimension of the pane in pixels. Units specify either an absolute value, a percentage, or a preferred value: Abs (default) or Percent, or Pref. |
| Fill | checkbox | When Fill is selected all widgets in a given row are sized to fill the row height. When deselected the widgets in a given row use their preferred height and alignment. |

ScrollPane

ScrollPane, in the `ba_jaui` palette, layouts one child using a viewport which may be scrolled to view the entire child's actual size.

Figure 110 ScrollPane Property Sheet

Property Sheet

ScrollPane (ScrollPane)

Visible true

Enabled true

Layout

| | Value | Units | |
|--------|--------|-------|--|
| X | 0.00 | Abs | |
| Y | 0.00 | Abs | |
| Width | 100.00 | Abs | |
| Height | 100.00 | Abs | |

Fill

Content Null Widget

Hpolicy As Needed

Vpolicy As Needed

Viewport Background (null)

Border Policy Always

Refresh Save

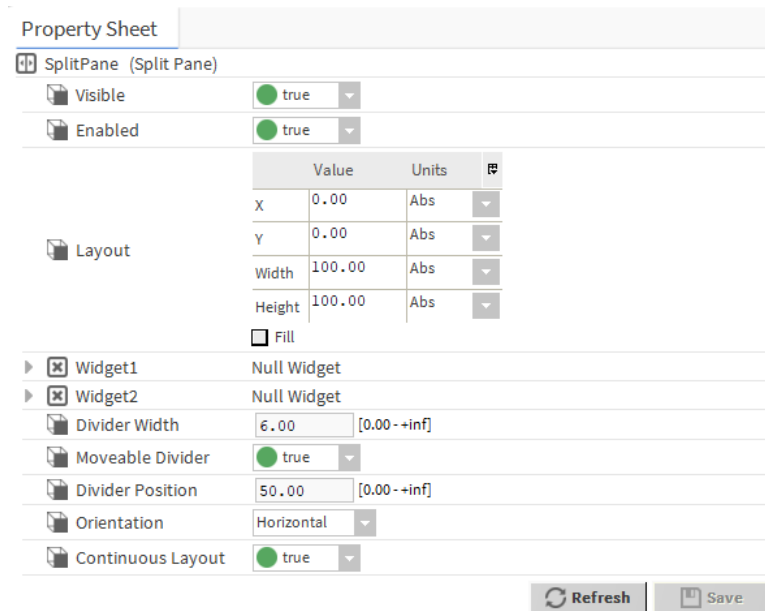
You access these properties by expanding **Bajauri** palette and navigate to **Panes**, double-click **ScrollPane**

| Property | Value | Description |
|----------|-------------------------|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<code>true</code>) or not (<code>false</code>). |
| Enabled | true (default) false | When set to <code>true</code> , the table in Px page interface is commandable using the popup menu. When set to <code>false</code> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Top | additional properties | Configures additional parameters for the scroll bar for the pane. |
| Left | additional properties | Configures additional parameters for the scroll bar for the pane. |
| Center | additional properties | Configures additional parameters for the scroll bar for the pane. |
| Right | additional properties | Configures additional parameters for the scroll bar for the pane. |
| Bottom | additional properties | Configures additional parameters for the scroll bar for the pane. |

SplitPane

SplitPane, in the `bajauri` palette, is a pane with a divider between two child widgets. The children can be laid out horizontally or vertically. Use the **SplitPane** properties to configure the behavior and position of the divider between the two panes.

Figure 111 SplitPane Property Sheet



You access these properties by expanding **Bajoui** palette and navigate to **Pane**, double-click **SplitPane**

| Property | Value | Description |
|-------------------|-------------------------|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<code>true</code>) or not (<code>false</code>). |
| Enabled | true (default) false | When set to <code>true</code> , the table in Px page interface is commandable using the popup menu. When set to <code>false</code> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Widget 1 | additional properties | Configures additional properties for the widget. |
| Widget 2 | additional properties | Configures additional properties for the widget. |
| Divider Width | number | Sets the width of the split plane divider. |
| Moveable Divider | true (default) false | When set to <code>true</code> allows the divider to move. When set to <code>false</code> does not allow the divider to move. |
| Divider Position | number | Specifies the divider position. |
| Orientation | drop-down list | Specifies the orientation of the divider. |
| Continuous layout | true (default) false | When set to <code>true</code> allows continuous layout. When set to <code>false</code> does not allow continuous layout. |

TabbedPane

TabbedPane, in the `bajai` palette, is a container which displays a set of LabelPane children one at a time using a set of tabs to select the currently displayed child. The LabelPane's label is used to label each tab, and the content of the current selection fills the rest of the pane.

Figure 112 TabbedPane Property Sheet

Property Sheet

TabbedPane (Tabbed Pane)

Visible true

Enabled true

Layout

| | Value | Units | |
|--------|--------|-------|--|
| X | 0.00 | Abs | |
| Y | 0.00 | Abs | |
| Width | 100.00 | Abs | |
| Height | 100.00 | Abs | |

Fill

Tab Placement Top

Show Single Tab true

Paint Full Border true

Refresh Save

You access these properties by expanding `Bajai` palette and navigate to `Panes`, double-click `TabbedPane`

| Property | Value | Description |
|-------------------|--|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<code>true</code>) or not (<code>false</code>). |
| Enabled | true (default) false | When set to <code>true</code> , the table in Px page interface is commandable using the popup menu. When set to <code>false</code> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Tab Placement | Top (default) Left Right, Bottom, and Center | Specifies the location of the tab. |
| Show Single Tab | true (default) false | When set to <code>true</code> allows to show single tab. When set to <code>false</code> does not allow to show single tab. |
| Paint Full Border | true (default) false | When set to <code>true</code> allows to paint the full border. When set to <code>false</code> does not allow to paint the full border. |

TextEditorOptions

bajai-TextEditorOptions

The `TextEditorOptions`, in the `bajai` palette, stores the options used to configure text entry. These are stored under `/user/{user}/textEditor.options`. The Status Colors options allow you to view and change the following (default):

- Show Spaces (`false`)
- Show Tabs (`false`)

- Show Newlines (false)
- Tab To Space Conversion (2)
- Show Margin (0)
- Color Coding
 - Foreground (00 00 00 black)
 - Whitespace (c0 c0 c0 gray)
 - Number Literal (80 00 80 purple)
 - String Literal (80 00 80 purple)
 - Identifier (00 00 00 black)
 - Keyword (00 00 ff blue)
 - Preprocessor (80 00 00 maroon)
 - Bracket (ff 00 00 red)
 - Line Comment (00 80 00 green)
 - Multi Line Comment (00 80 00 green)
 - Non Javadoc Comment (80 80 80 dark gray)
- Key Bindings (with default)
 - Move Up Up
 - Move Down Down
 - Move Left Left
 - Move Right Right
 - Page Up PageUp
 - Page Down PageDown
 - Line Start Home
 - Line End End
 - Document Start Ctrl + Home
 - Document End Ctrl + End
 - Word Left Ctrl + Left
 - Word Right Ctrl + Right
 - Cut Ctrl + X
 - Copy Ctrl + C
 - Paste Ctrl + V
 - Undo Ctrl + Z
 - Redo Ctrl + Alt + Z
 - Delete Delete
 - Backspace Backspace
 - Cut Line Ctrl + Y
 - Delete Word Ctrl + Delete
 - Tab Forward Tab

- Tab Back Shift + Tab
- Toggle Slash Slash Esc
- Word Wrap Ctrl + W
- Goto Line Ctrl + G
- Find F5
- Find Next Ctrl + F
- Find Prev Ctrl + Shift + F
- Replace F6
- Reload Macros Ctrl + M
- Select All Ctrl + A
- Undo Navigation (true)
- Match Parens (true)
- Match Braces (true)
- Match Brackets (true)

ValueBinding

bajoui-ValueBinding

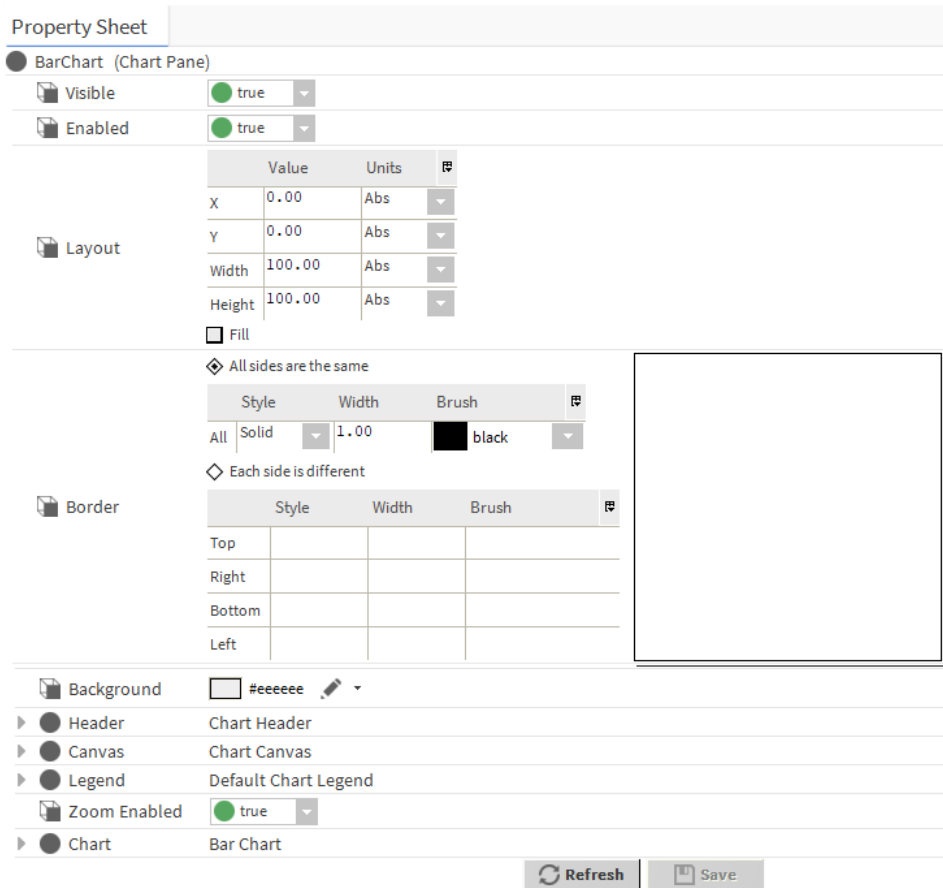
ValueBinding, in the `bajoui` palette, is used to bind to Values typically under a Component. Widget properties may be overridden by creating dynamic slots of the same name with a Converter that maps the bound target to the property value.

Components in chart module


BarChart

One of two chart types available (the other is Linechart).

Figure 113 BarChart Property Sheet



You access these properties by expanding **Chart** palette and double-click **BarChart**

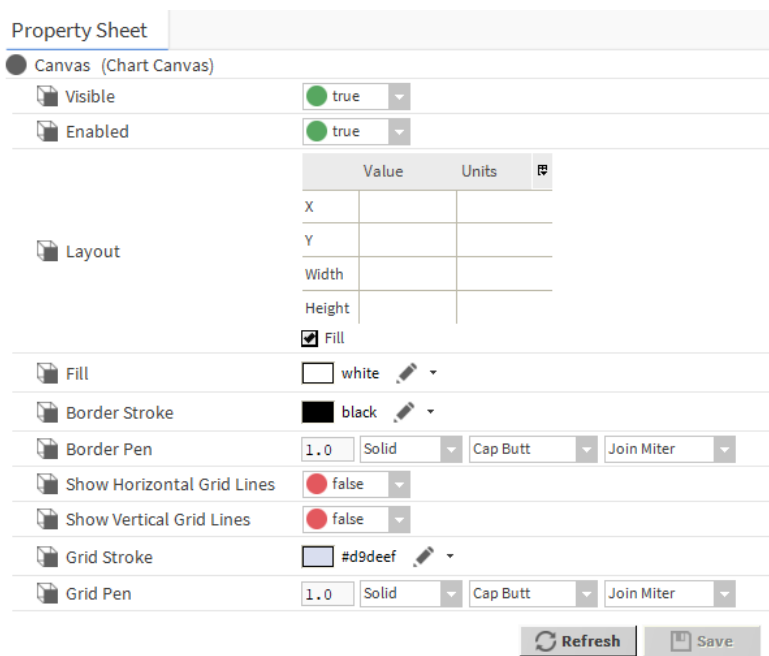
| Property | Value | Description |
|------------|--|---|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Border | text | Displays the border design. |
| Background | Solid, Gradient, Image, Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |

| Property | Value | Description |
|--------------|-------------------------|--|
| Header | additional properties | Configures additional parameters for header. |
| Canvas | additional properties | Configures additional parameters for canvas. |
| Legend | additional properties | Configures additional parameters for chart legend. |
| Zoom Enabled | true (default) false | When set to true allows the chart to zoom in. When set to false does not allows the charts to zoom in. |
| Chart | additional properties | Configures additional parameters for chart. |

ChartCanvas

Chart Canvas is the canvas widget under a ChartPane.

Figure 114 ChartCanvas Property Sheet



You access these properties by expanding **Bajau** palette and navigate to **Barchart**, double-click **Canvas**

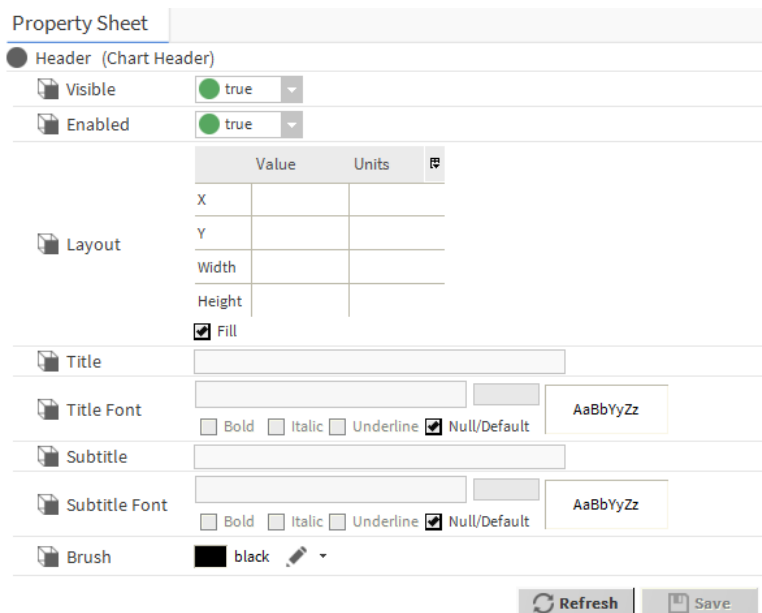
| Property | Value | Description |
|----------|-------------------------|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |

| Property | Value | Description |
|----------------------------|--|---|
| Fill | Solid, Gradient, Image ,Null (default) | Specifies color fill of the pane . Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no background fill. |
| Border Stroke | Solid, Gradient, Image ,Null (default) | Specifies color of the border. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Texture window, click the Browse icon to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no border fill. |
| Border Pen | additional properties | Configure additional properties for border. |
| Show Horizontal Grid Lines | true false (default) | When set to true horizontal grid lines are visible. When set to false horizontal grid lines are invisible. |
| Show Vertical Grid Lines | true false (default) | When set to true vertical grid lines are visible. When set to false vertical grid lines are invisible. |
| Grid Stroke | Solid, Gradient, Image ,Null (default) | Specifies color of the grid. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Texture window, click the Browse icon to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no grid fill. |
| Grid Pen | additional properties | configures additional parameters for grid pen. |

ChartHeader

ChartHeader is the header widget under a ChartPane.

Figure 115 ChartHeader Property Sheet



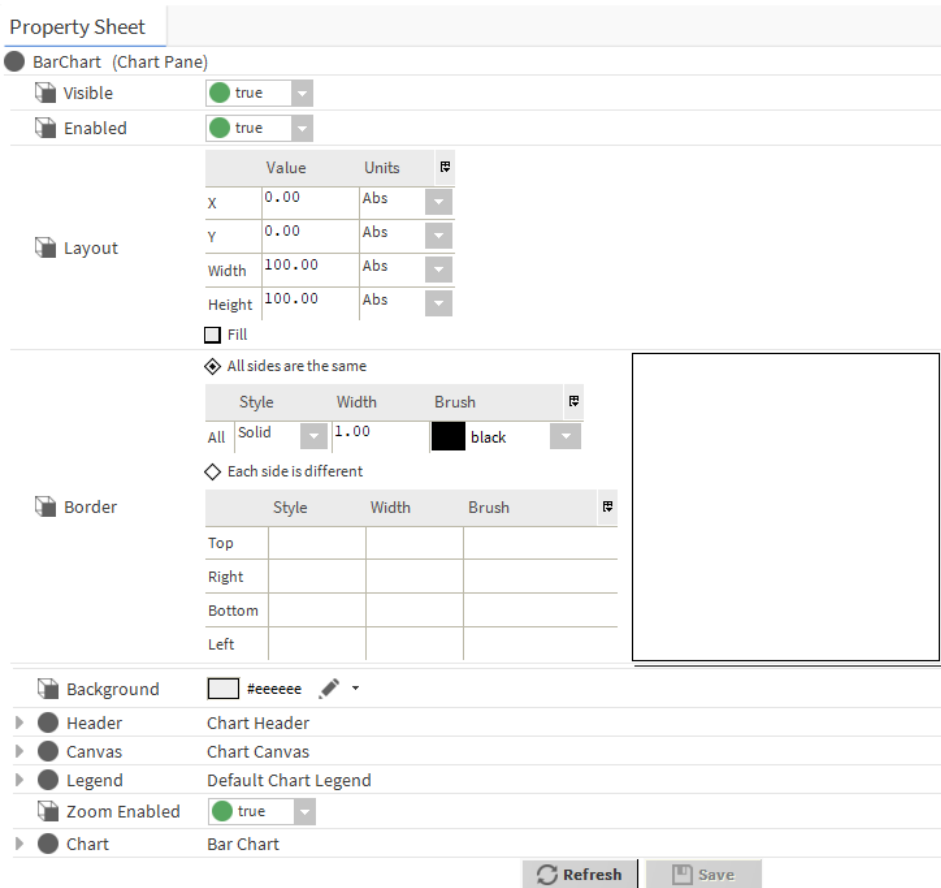
You access these properties by expanding **Chart** palette and navigate to **BarChart** or **LineChart**, double-click **Header**

| Property | Value | Description |
|---------------|--|---|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<code>true</code>) or not (<code>false</code>). |
| Enabled | true (default) false | When set to <code>true</code> , the table in Px page interface is commandable using the popup menu. When set to <code>false</code> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Title | Text | Specifies the title of the header. |
| Title Font | additional properties | Configures the font parameters for the title of the header. |
| Subtitle | Text | Specifies the subtitle of the header. |
| Subtitle Font | additional properties | Configures the font parameters for the subtitle of the header. |
| Brush | Solid, Gradient, Image, Null (default) | Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Texture window, click the Browse icon to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no fill. |


ChartPane

ChartPane is the container widget created when adding a Px chart.

Figure 116 BarChart Property Sheet



You access these properties by expanding **Chart** palette and double-click **BarChart**

| Property | Value | Description |
|------------|--|---|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Border | text | Displays the border design. |
| Background | Solid, Gradient, Image ,Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |

| Property | Value | Description |
|--------------|-------------------------|--|
| Header | additional properties | Configures additional parameters for header. |
| Canvas | additional properties | Configures additional parameters for canvas. |
| Legend | additional properties | Configures additional parameters for chart legend. |
| Zoom Enabled | true (default) false | When set to <code>true</code> allows the chart to zoom in. When set to <code>false</code> does not allows the charts to zoom in. |
| Chart | additional properties | Configures additional parameters for chart. |

DefaultChartLegend

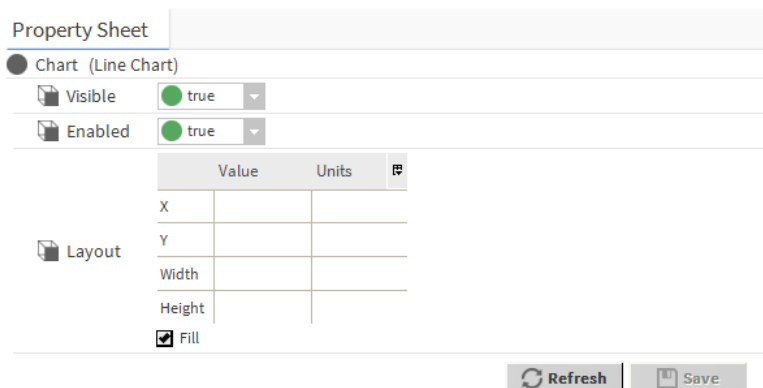
chart-DefaultChartLegend

DefaultChartLegend is the legend widget under a ChartPane.

LineChart

One of two chart types available (the other is BarChart).

Figure 117 LineChart Property Sheet



You access these properties by expanding **Chart** palette and navigate to **LineChart** and double-click **Chart**

| Property | Value | Description |
|----------|-------------------------|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<code>true</code>) or not (<code>false</code>). |
| Enabled | true (default) false | When set to <code>true</code> , the table in Px page interface is commandable using the popup menu. When set to <code>false</code> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |

Components in gx module

Brush

gx-Brush

Brush is used to change presentation. the Brush is available in the **gx** module.

Components in kitPx module

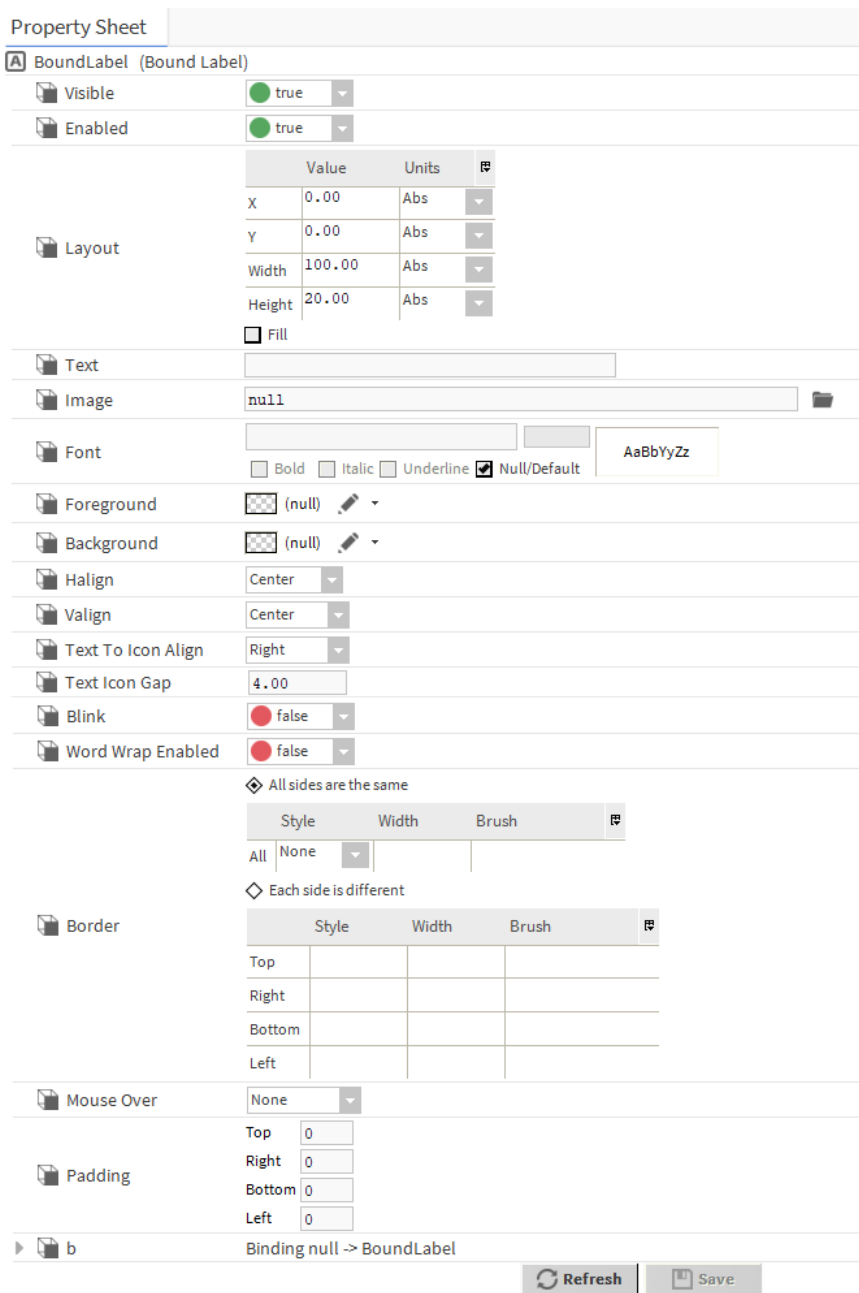
BoundLabel, BooleanImage, NumericImage, or EnumImage

This topic documents BoundLabel, BooleanImage, NumericImage, or EnumImage.

kitPx-BoundLabel


kitPx-BoundLabel is a widget that simply provides a label widget with a binding already available.

Figure 118 Boundlabel Property Sheet



You access these properties by expanding **KitPx** palette and double-click **BoundLabel**

| Property | Value | Description |
|----------|-------------------------|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |

| Property | Value | Description |
|--------------------|--|---|
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Text | text | Specifies the text on the label. |
| Image | file chooser | Allows to choose the image for the label. |
| Font | additional properties | Configures the additional parameters for the font on the label. |
| Foreground | Solid, Gradient, Image, Null (default) | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no foreground fill. |
| Background | Solid, Gradient, Image, Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |
| Halign | Left, Center (default), Right, Fill | Selects from among horizontal alignment options. |
| Valign | Left, Center (default), Right, Fill | Specifies how to vertically align the pane. |
| Text To Icon Align | Left, Center, Right (default), Top, Bottom | Defines the alignment of the text in relationship to the icon: Right, Top, Bottom, Left, Center |
| Text To Icon gap | number | Defines the gap between the text and icon. |
| Blink | true false (default) | Configures the background color to blink. |
| Word Wrap Enabled | true false (default) | Allows a long word on a button to wrap (<i>true</i>) or prevents the word from wrapping (<i>false</i>). |
| Border | additional properties | Configures additional parameters for the border of the label. |
| Mouse Over | drop-down | Configures what happens when a user passes the mouse cursor over the graphic. |
| Padding | additional properties | Configures the padding for the label. |
| b | additional properties | Configures the additional parameters for binding the label. |

kitPx-BooleanImage

kitPx-BooleanImage is a widget that simply provides a label widget with a binding already available. Refer to above mentioned properties for further configuration.

kitPx-NumericImage

kitPx-NumericImage is a widget that simply provides a label widget with a binding already available. Refer to above mentioned properties for further configuration.

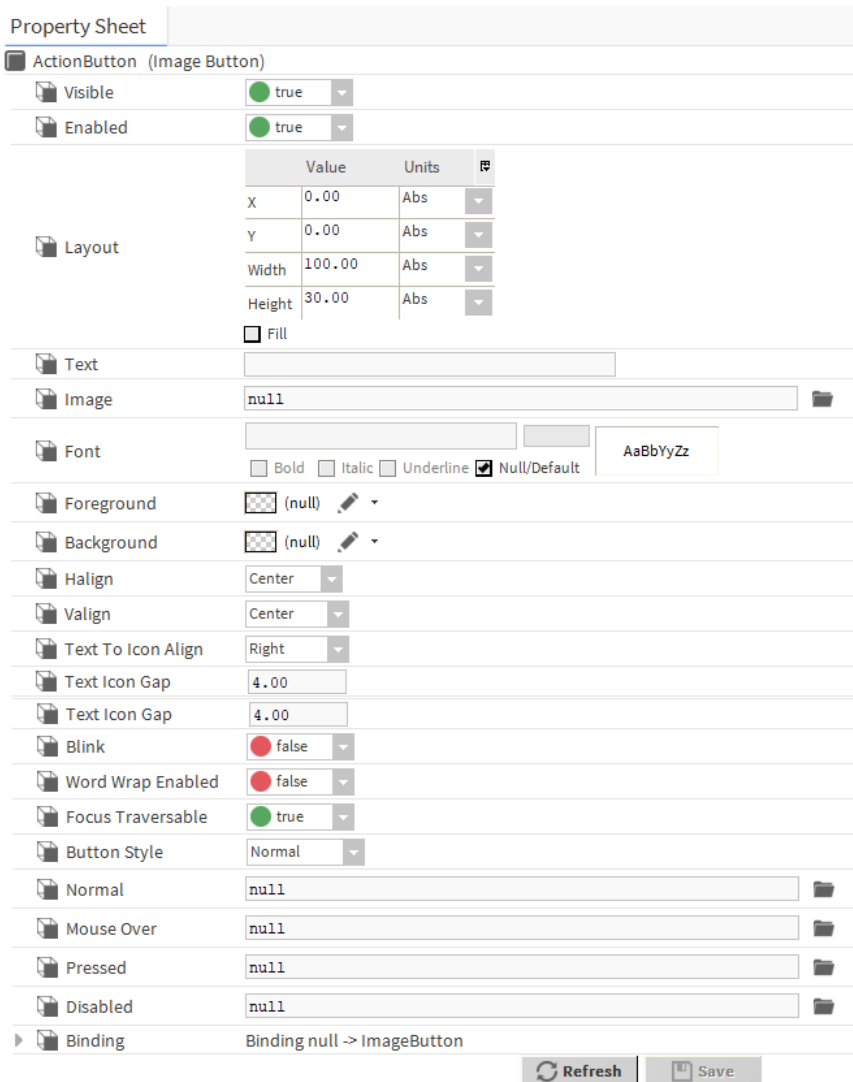
kitPx-EnumImage

kitPx-EnumImage is a widget that simply provides a label widget with a binding already available. Refer to above mentioned properties for further configuration.

ActionButton


This topic documents ActionButton,. kitPx-ActionButton is a button that uses images to display the **state** of the button. For example, the action button may have three separate images to indicate the `normal`, `mouse-Over`, and `pressed` states.

Figure 119 Action Button Property Sheet



You access these properties by expanding **KitPx** palette and double-click **ActionButton**

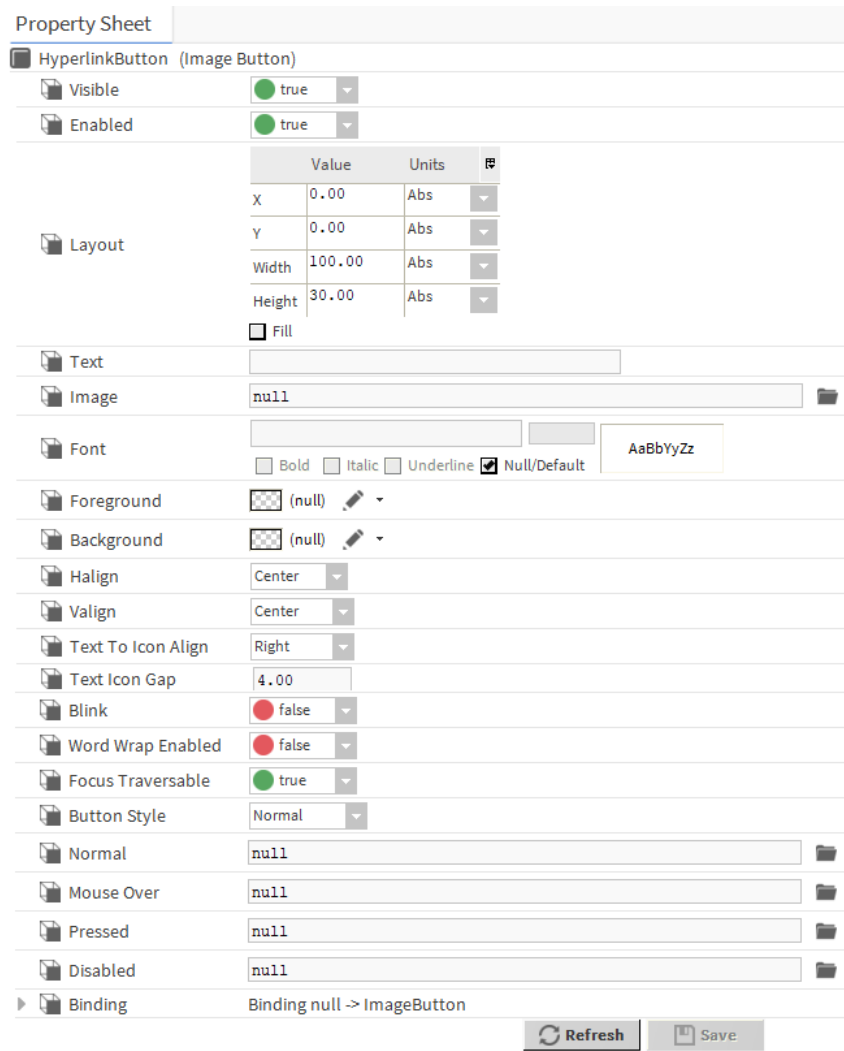
| Property | Value | Description |
|----------|-------------------------|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Text | text | Specifies the text on the label. |
| Image | file chooser | Allows to choose the image for the label. |
| Font | additional properties | Configures the additional parameters for the font on the label. |

| Property | Value | Description |
|--------------------|---|---|
| Foreground | Solid, Gradient, Image, Null (default) | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, or d chooser to select the image file. Null indicates no foreground fill. |
| Background | Solid, Gradient, Image, Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |
| Halign | Left, Center (default), Right, Fill | Selects from among horizontal alignment options. |
| Valign | Left, Center (default), Right, Fill | Specifies how to vertically align the pane. |
| Text To Icon Align | Left, Center, Right (default), Top, Bottom | Defines the alignment of the text in relationship to the icon: Right, Top, Bottom, Left, Center |
| Text To Icon gap | number | Defines the gap between the text and icon. |
| Blink | true false (default) | Configures the background color to blink. |
| Word Wrap Enabled | true false (default) | Allows a long word on a button to wrap (true) or prevents the word from wrapping (false). |
| Focus Trasversible | true false (default) | |
| Normal | files chooser | Specifies the image for Normal state. |
| Mouse Over | files chooser | Specifies the image for Mouse over state. |
| Pressed | files chooser | Specifies the image for Pressed state. |
| Disabled | files chooser | Specifies the image for Disabled state. |
| b | additional properties | Configures the additional parameters for binding the label. |
| Button Style | Normal (default), Tool Bar, Hyperlink, None | Selects the button style: Normal, Tool Bar, Hyperlink, None |

HyperLink Button


kitPx-HyperlinkButton is a button that uses images to display the **state** of the button. For example, the action button may have three separate images to indicate the `normal`, `mouseOver`, and `pressed` states.

Figure 120 HyperlinkButton Property Sheet



You access these properties by expanding **KitPx** palette and double-click **HyperLinkButton**

| Property | Value | Description |
|----------|-------------------------|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Text | text | Specifies the text on the label. |
| Image | file chooser | Allows to choose the image for the label. |
| Font | additional properties | Configures the additional parameters for the font on the label. |

| Property | Value | Description |
|--------------------|--|---|
| Foreground | Solid, Gradient, Image, Null (default) | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no foreground fill. |
| Background | Solid, Gradient, Image, Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |
| Halign | Left, Center (default), Right, Fill | Selects from among horizontal alignment options. |
| Valign | Left, Center (default), Right, Fill | Specifies how to vertically align the pane. |
| Text To Icon Align | Left, Center, Right (default), Top, Bottom | Defines the alignment of the text in relationship to the icon: Right, Top, Bottom, Left, Center |
| Text To Icon gap | number | Defines the gap between the text and icon. |
| Blink | true false (default) | Configures the background color to blink. |
| Word Wrap Enabled | true false (default) | Allows a long word on a button to wrap (true) or prevents the word from wrapping (false). |
| Focus Trasversible | true false (default) | |
| Normal | files chooser | Specifies the image for Normal state. |
| Mouse Over | files chooser | Specifies the image for Mouse over state. |
| Pressed | files chooser | Specifies the image for Pressed state. |
| Disabled | files chooser | Specifies the image for Disabled state. |
| Binding | additional properties | Configures the additional parameters for binding the label. |

IncrementSetPointButton

kitPx-IncrementSetPointButton is a button that uses images to display the **state** of the button. For example, the action button may have three separate images to indicate the `normal`, `mouseOver`, and `pressed` states.

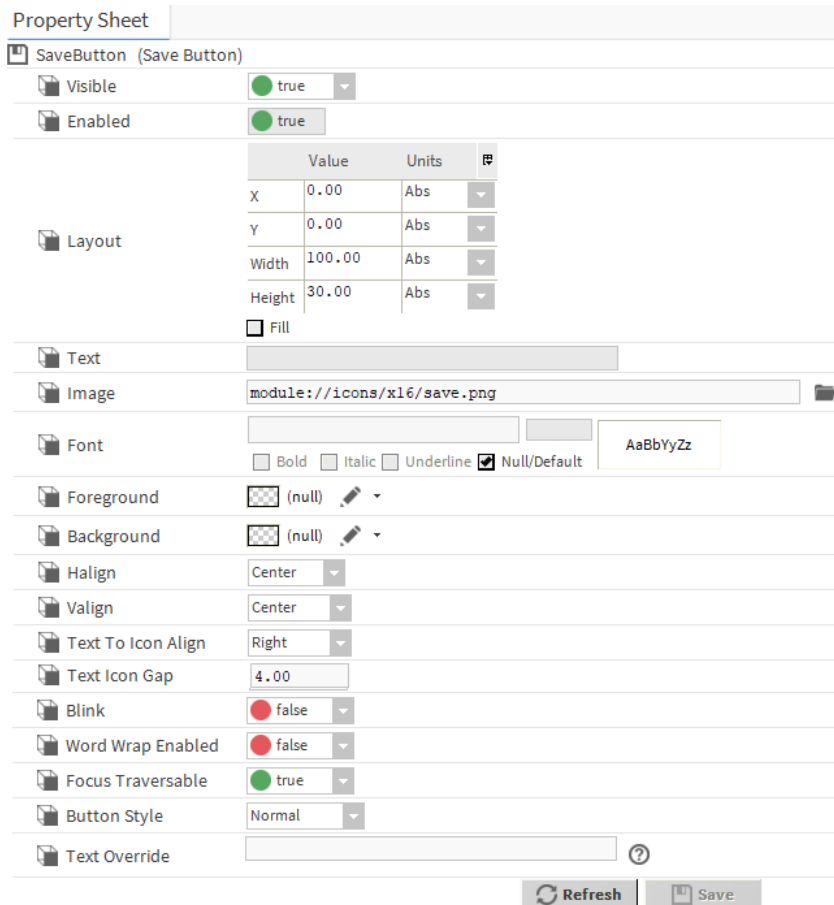
DecrementSetPointButton

kitPx-DecrementSetPointButton is a button that uses images to display the **state** of the button. For example, the action button may have three separate images to indicate the `normal`, `mouseOver`, and `pressed` states.

SaveButton


kitPx is used to issue a save command when it is clicked. The Save text that appears on the button is included by default in the Save property—but, like the other properties, it may be edited, as desired.

Figure 121 SaveButton Property Sheet



You access these properties by expanding **KitPx** palette and double-click **SaveButton**

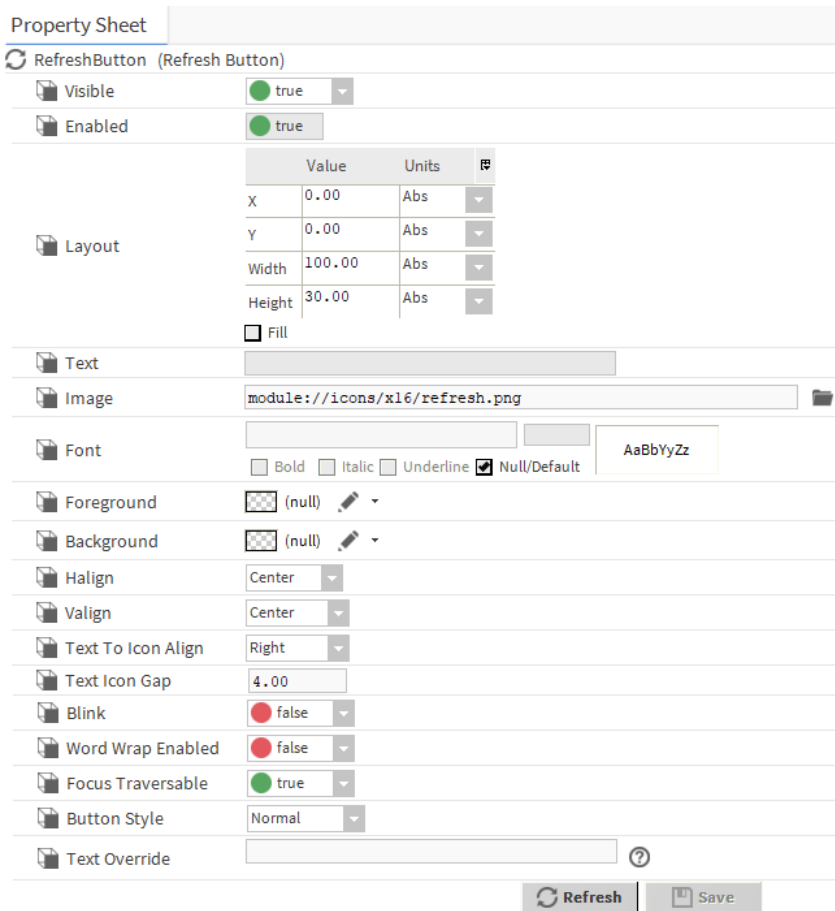
| Property | Value | Description |
|----------|-------------------------|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<code>true</code>) or not (<code>false</code>). |
| Enabled | true (default) false | When set to <code>true</code> , the table in Px page interface is commandable using the popup menu. When set to <code>false</code> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Text | text | Specifies the text on the label. |
| Image | file chooser | Allows to choose the image for the label. |
| Font | additional properties | Configures the additional parameters for the font on the label. |

| Property | Value | Description |
|--------------------|---|---|
| Foreground | Solid, Gradient, Image, Null (default) | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, or d chooser to select the image file. Null indicates no foreground fill. |
| Background | Solid, Gradient, Image, Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |
| Halign | Left, Center (default), Right, Fill | Selects from among horizontal alignment options. |
| Valign | Left, Center (default), Right, Fill | Specifies how to vertically align the pane. |
| Text To Icon Align | Left, Center, Right (default), Top, Bottom | Defines the alignment of the text in relationship to the icon: Right, Top, Bottom, Left, Center |
| Text To Icon gap | number | Defines the gap between the text and icon. |
| Blink | true false (default) | Configures the background color to blink. |
| Word Wrap Enabled | true false (default) | Allows a long word on a button to wrap (true) or prevents the word from wrapping (false). |
| Focus Trasversable | true false (default) | |
| Button Style | Normal (default), Tool Bar, Hyperlink, None | Selects the button style: Normal, Tool Bar, Hyperlink, None |
| Text Override | text | |

RefreshButton


kitPx-RefreshButton is used to issue a **refresh** command when it is clicked. The Refresh text that appears on the button is included by default in the Text property. This button is designed to work well with Touchscreen displays that require a larger button. The Refresh button is located in the kitPx palette and is pre-configured so that all you have to do is drag it onto a Px page and save the page for it for it to take effect.

Figure 122 Refresh button Property Sheet



You access these properties by expanding **KitPx** palette and double-click **RefreshButton**

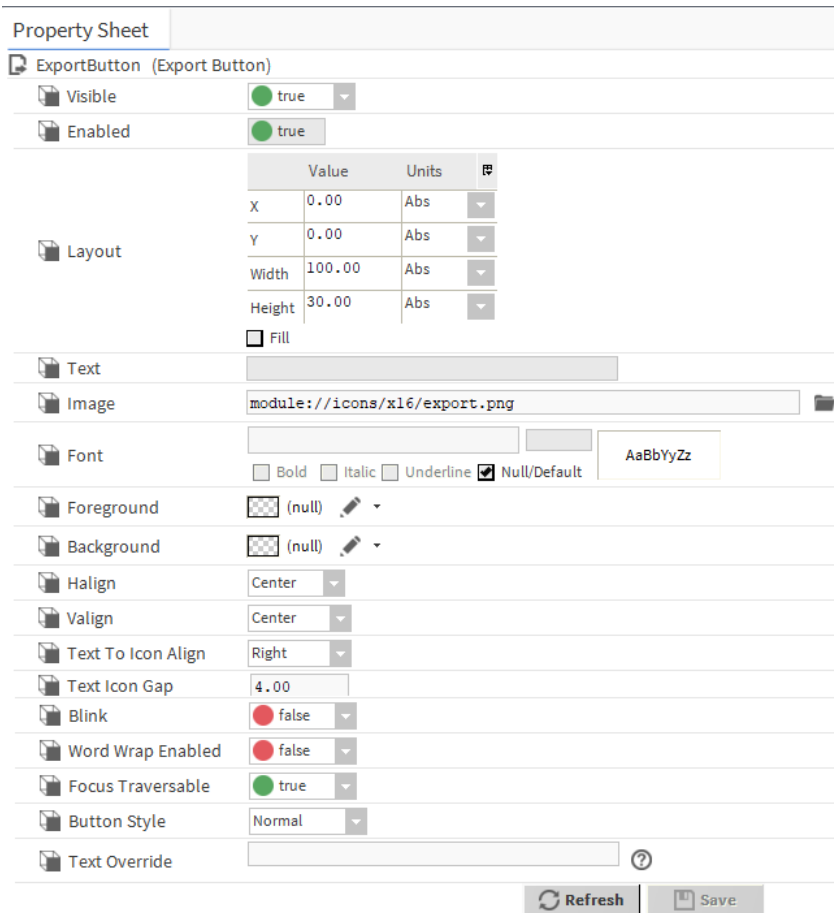
| Property | Value | Description |
|------------|--|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Text | text | Specifies the text on the label. |
| Image | file chooser | Allows to choose the image for the label. |
| Font | additional properties | Configures the additional parameters for the font on the label. |
| Foreground | Solid, Gradient, Image, Null (default) | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, or d chooser to select the image file. Null indicates no foreground fill. |

| Property | Value | Description |
|--------------------|--|---|
| Background | Solid, Gradient, Image, Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |
| Halign | Left, Center (default), Right, Fill | Selects from among horizontal alignment options. |
| Valign | Left, Center (default), Right, Fill | Specifies how to vertically align the pane. |
| Text To Icon Align | Left, Center, Right(default), Top, Bottom | Defines the alignment of the text in relationship to the icon: Right, Top, Bottom, Left, Center |
| Text To Icon gap | number | Defines the gap between the text and icon. |
| Blink | true false (default) | Configures the background color to blink. |
| Word Wrap Enabled | true false (default) | Allows a long word on a button to wrap (true) or prevents the word from wrapping (false). |
| Focus Trasversable | true false (default) | |
| Button Style | Normal(default), Tool Bar, Hyperlink, None | Selects the button style: Normal, Tool Bar, Hyperlink, None |
| Text Override | text | |

ExportButton


kitPx-ExportButton can be used in Touchscreen displays that require a larger button. It works the same way as the Export button that is located on the Workbench toolbar menu (opens the Export dialog box when clicked). The ExportButton widget is located in the kitPx palette and is pre-configured so that all you have to do is drag it onto a Px page and save the page for it for it to take effect.

Figure 123 ExportButton Property Sheet



You access these properties by expanding KitPx palette and double-click **ExportButton**

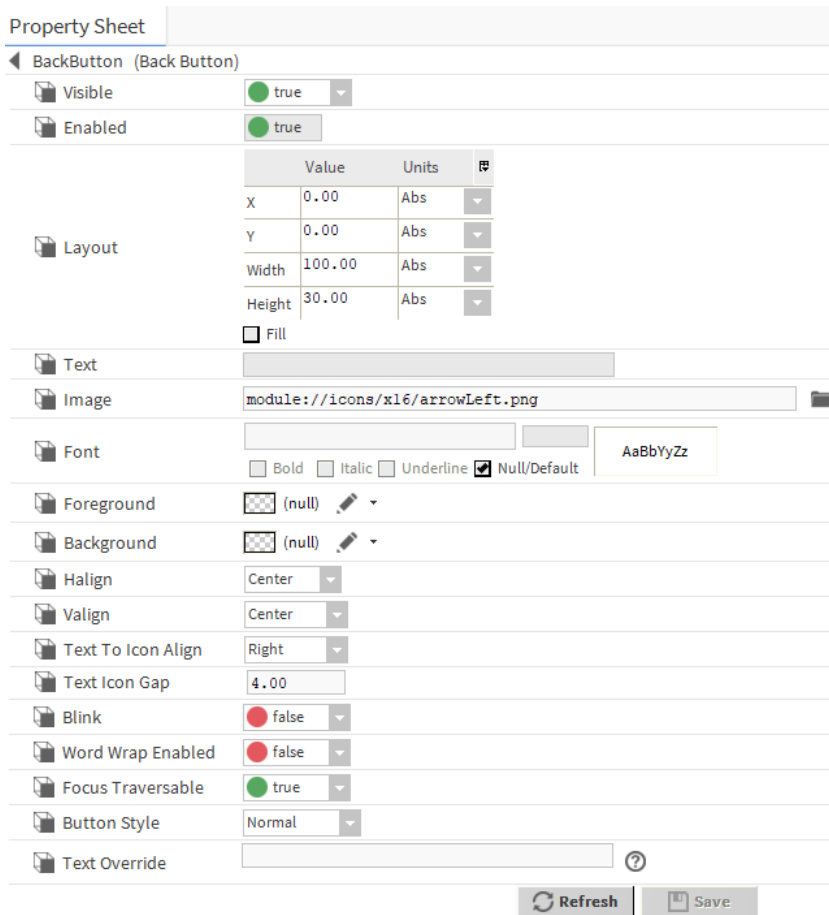
| Property | Value | Description |
|------------|--|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Text | text | Specifies the text on the label. |
| Image | file chooser | Allows to choose the image for the label. |
| Font | additional properties | Configures the additional parameters for the font on the label. |
| Foreground | Solid, Gradient, Image, Null (default) | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, or d chooser to select the image file. Null indicates no foreground fill. |

| Property | Value | Description |
|--------------------|---|---|
| Background | Solid, Gradient, Image, Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |
| Halign | Left, Center (default), Right, Fill | Selects from among horizontal alignment options. |
| Valign | Left, Center (default), Right, Fill | Specifies how to vertically align the pane. |
| Text To Icon Align | Left, Center, Right (default), Top, Bottom | Defines the alignment of the text in relationship to the icon: Right, Top, Bottom, Left, Center |
| Text To Icon gap | number | Defines the gap between the text and icon. |
| Blink | true false (default) | Configures the background color to blink. |
| Word Wrap Enabled | true false (default) | Allows a long word on a button to wrap (true) or prevents the word from wrapping (false). |
| Focus Trasversable | true false (default) | |
| Button Style | Normal (default), Tool Bar, Hyperlink, None | Selects the button style: Normal, Tool Bar, Hyperlink, None |
| Text Override | text | |

BackButton


kitPx-BackButton is designed for Touchscreen displays that require a larger button. This widget works like the standard Workbench Back button that is located in the top left corner of the window. The button is dimmed when no previous page has been visited and is enabled when a previous page is available to go back to when you click it. It is located in the kitPx palette and is pre-configured so that all you have to do is drag it onto a Px page and save the page for it for it to take effect.

Figure 124 BackButton Property Sheet



You access these properties by expanding KitPx palette and double-click **BackButton**

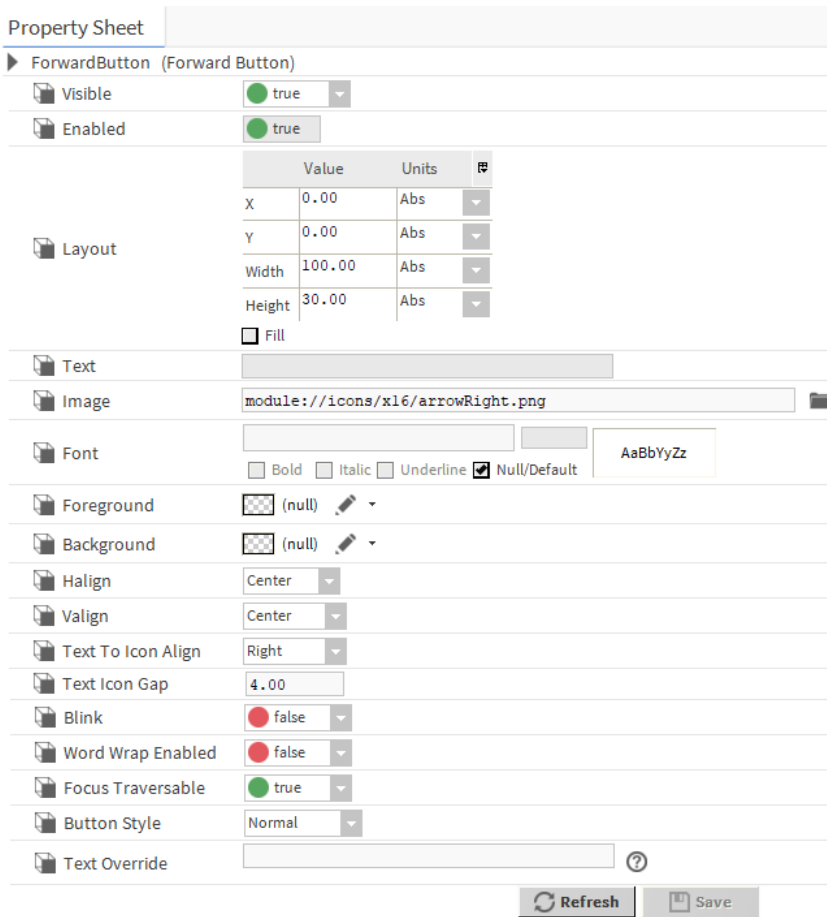
| Property | Value | Description |
|------------|--|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<i>true</i>) or not (<i>false</i>). |
| Enabled | true (default) false | When set to <i>true</i> , the table in Px page interface is commandable using the popup menu. When set to <i>false</i> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Text | text | Specifies the text on the label. |
| Image | file chooser | Allows to choose the image for the label. |
| Font | additional properties | Configures the additional parameters for the font on the label. |
| Foreground | Solid, Gradient, Image, Null (default) | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, or d chooser to select the image file. Null indicates no foreground fill. |

| Property | Value | Description |
|--------------------|--|---|
| Background | Solid, Gradient, Image, Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |
| Halign | Left, Center (default), Right, Fill | Selects from among horizontal alignment options. |
| Valign | Left, Center (default), Right, Fill | Specifies how to vertically align the pane. |
| Text To Icon Align | Left, Center, Right(default), Top, Bottom | Defines the alignment of the text in relationship to the icon: Right, Top, Bottom, Left, Center |
| Text To Icon gap | number | Defines the gap between the text and icon. |
| Blink | true false (default) | Configures the background color to blink. |
| Word Wrap Enabled | true false (default) | Allows a long word on a button to wrap (true) or prevents the word from wrapping (false). |
| Focus Trasversable | true false (default) | |
| Button Style | Normal(default), Tool Bar, Hyperlink, None | Selects the button style: Normal, Tool Bar, Hyperlink, None |
| Text Override | text | |

ForwardButton


kitPx-ForwardButton is designed for Touchscreen displays that require a larger button. This widget works like the standard Workbench Forward button that is located in the top left corner of the toolbar. The button is dimmed when no forward page is available and is enabled when a page is available to go forward to when you click it. It is located in the kitPx palette and is pre-configured so that all you have to do is drag it onto a Px page and save the page for it for it to take effect.

Figure 125 ForwardButton Property Sheet



You access these properties by expanding **KitPx** palette and double-click **ForwardButton**

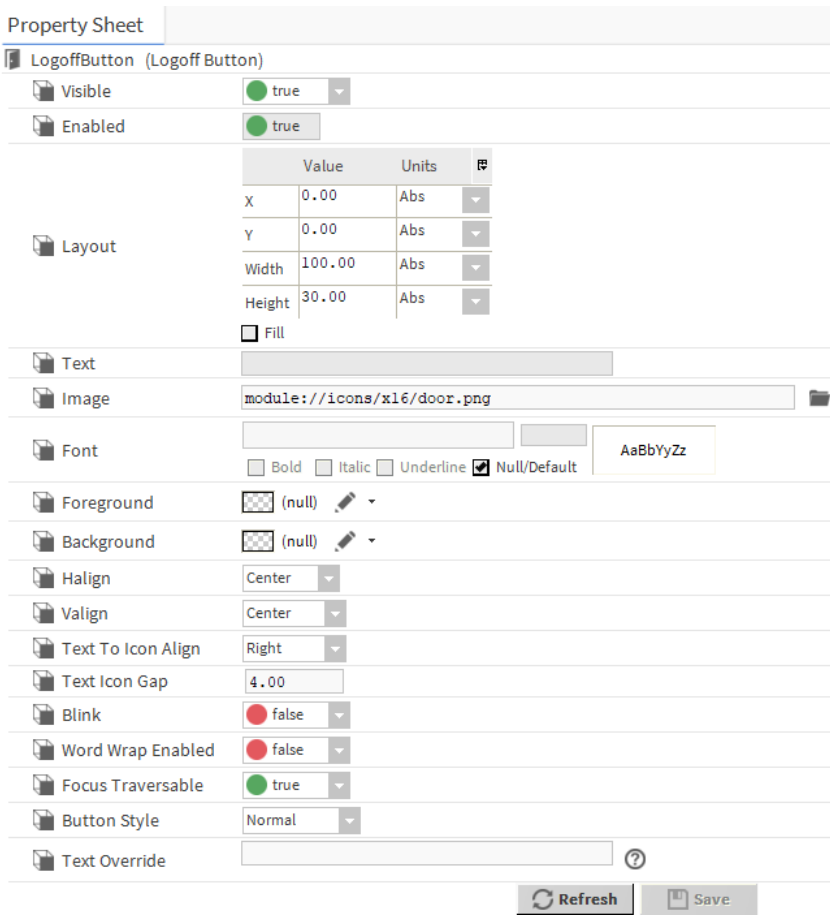
| Property | Value | Description |
|------------|--|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<code>true</code>) or not (<code>false</code>). |
| Enabled | true (default) false | When set to <code>true</code> , the table in Px page interface is commandable using the popup menu. When set to <code>false</code> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Text | text | Specifies the text on the label. |
| Image | file chooser | Allows to choose the image for the label. |
| Font | additional properties | Configures the additional parameters for the font on the label. |
| Foreground | Solid, Gradient, Image, Null (default) | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File |

| Property | Value | Description |
|--------------------|---|---|
| | | chooser, ord chooser to select the image file. Null indicates no foreground fill. |
| Background | Solid, Gradient, Image, Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |
| Halign | Left, Center (default), Right, Fill | Selects from among horizontal alignment options. |
| Valign | Left, Center (default), Right, Fill | Specifies how to vertically align the pane. |
| Text To Icon Align | Left, Center, Right (default), Top, Bottom | Defines the alignment of the text in relationship to the icon: Right, Top, Bottom, Left, Center |
| Text To Icon gap | number | Defines the gap between the text and icon. |
| Blink | true false (default) | Configures the background color to blink. |
| Word Wrap Enabled | true false (default) | Allows a long word on a button to wrap (true) or prevents the word from wrapping (false). |
| Focus Trasversable | true false (default) | |
| Button Style | Normal (default), Tool Bar, Hyperlink, None | Selects the button style: Normal, Tool Bar, Hyperlink, None |
| Text Override | text | |

LogoffButton


kitPx-LogoffButton is designed for Touchscreen displays that require a larger button. When clicked, this widget automatically initiates a logoff command and disconnects the current user from any active session. The Logoff button is located in the kitPx palette and is pre-configured so that all you have to do is drag it on to a Px page and save the page for it for it to take effect.

Figure 126 LogOffButton Property Sheet



You access these properties by expanding KitPx palette and double-click LogOffButton

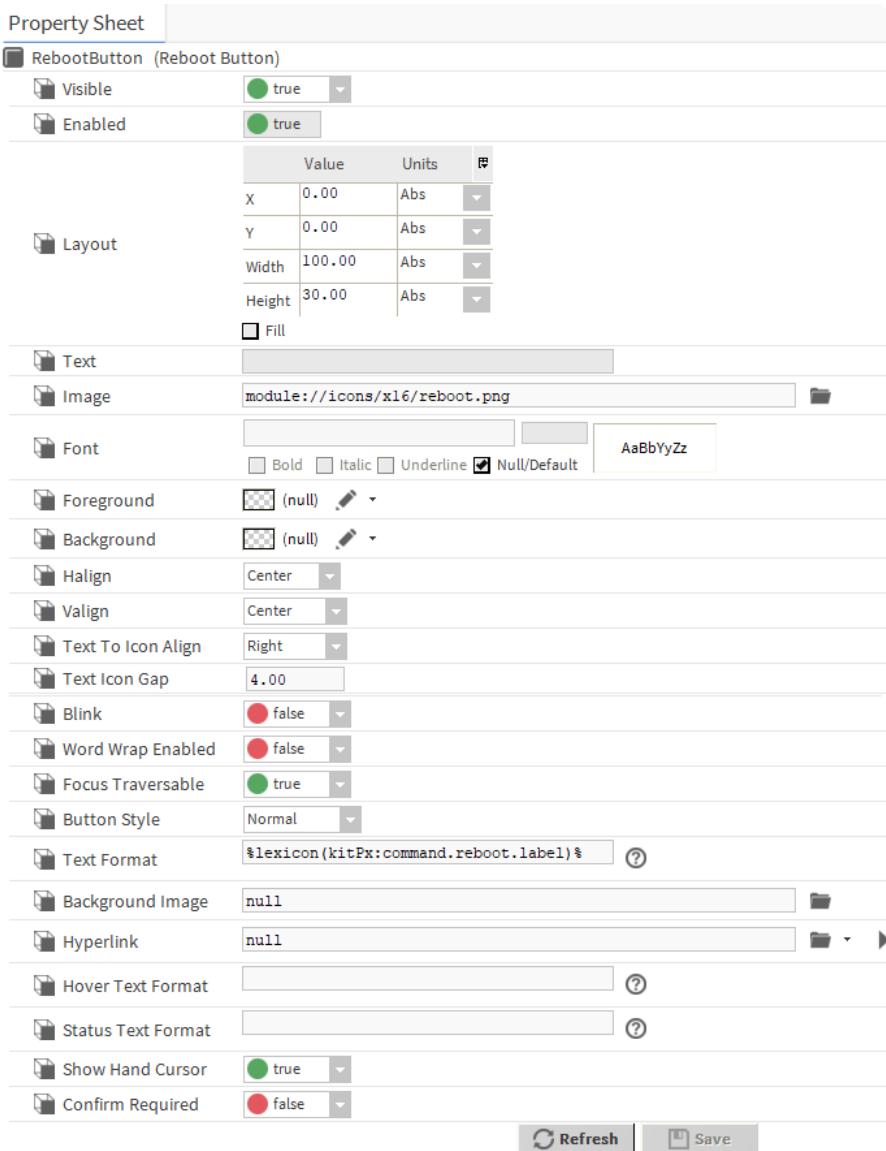
| Property | Value | Description |
|------------|--|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<i>true</i>) or not (<i>false</i>). |
| Enabled | true (default) false | When set to <i>true</i> , the table in Px page interface is commandable using the popup menu. When set to <i>false</i> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Text | text | Specifies the text on the label. |
| Image | file chooser | Allows to choose the image for the label. |
| Font | additional properties | Configures the additional parameters for the font on the label. |
| Foreground | Solid, Gradient, Image, Null (default) | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, or d chooser to select the image file. Null indicates no foreground fill. |

| Property | Value | Description |
|--------------------|---|---|
| Background | Solid, Gradient, Image, Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |
| Halign | Left, Center (default), Right, Fill | Selects from among horizontal alignment options. |
| Valign | Left, Center (default), Right, Fill | Specifies how to vertically align the pane. |
| Text To Icon Align | Left, Center, Right (default), Top, Bottom | Defines the alignment of the text in relationship to the icon: Right, Top, Bottom, Left, Center |
| Text To Icon gap | number | Defines the gap between the text and icon. |
| Blink | true false (default) | Configures the background color to blink. |
| Word Wrap Enabled | true false (default) | Allows a long word on a button to wrap (true) or prevents the word from wrapping (false). |
| Focus Trasversable | true false (default) | |
| Button Style | Normal (default), Tool Bar, Hyperlink, None | Selects the button style: Normal, Tool Bar, Hyperlink, None |
| Text Override | text | |

RebootButton


When clicked, this widget automatically initiates a Reboot command and restart the running station. The Reboot button is located in the kitPx palette and is pre-configured so that all you have to do is drag it onto a Px page and save the page for it for it to take effect.

Figure 127 RebootButton Property Sheet



You access these properties by expanding **KitPx** palette and double-click **RebootButton**

| Property | Value | Description |
|----------|-------------------------|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Text | text | Specifies the text on the label. |
| Image | file chooser | Allows to choose the image for the label. |

| Property | Value | Description |
|--------------------|---|---|
| Font | additional properties | Configures the additional parameters for the font on the label. |
| Foreground | Solid, Gradient, Image, Null (default) | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no foreground fill. |
| Background | Solid, Gradient, Image, Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |
| Halign | Left, Center (default), Right, Fill | Selects from among horizontal alignment options. |
| Valign | Left, Center (default), Right, Fill | Specifies how to vertically align the pane. |
| Text To Icon Align | Left, Center, Right (default), Top, Bottom | Defines the alignment of the text in relationship to the icon: Right, Top, Bottom, Left, Center |
| Text To Icon gap | number | Defines the gap between the text and icon. |
| Blink | true false (default) | Configures the background color to blink. |
| Word Wrap Enabled | true false (default) | Allows a long word on a button to wrap (true) or prevents the word from wrapping (false). |
| Focus Trasversable | true false (default) | |
| Button Style | Normal (default), Tool Bar, Hyperlink, None | Selects the button style: Normal, Tool Bar, Hyperlink, None |
| Text Format | text | Defines the text format for the reboot button. |
| Background image | file chooser | Allows to add background image for the button. |
| Hyperlink | drop-down list | Allows to select the option to insert hyperlink for the button. |
| Hover Text Format | text | Specifies the text format for hover text on the button. |
| Status Text Format | text | |
| Show Hand Cursor | true (default) false | When set to true shows the hand cursor. When set to false does not show the hand cursor. |
| Confirm Required | true false (default) | When set to true asks for confirmation for reboot. When set to false does not asks for confirmation for reboot. |

ButtonGroup

kitPx-ButtonGroup is a widget that provides a convenient way to bind a group of buttons to a common boolean or enum point. This widget is comprised of a standard GridPane with a Button-GroupBinding already available. The ButtonGroup properties are editable in the Px Editor properties side bar.

Figure 128 ButtonGroup Property Sheet



You access these properties by expanding **KitPx** palette and double-click **ButtonGroup**

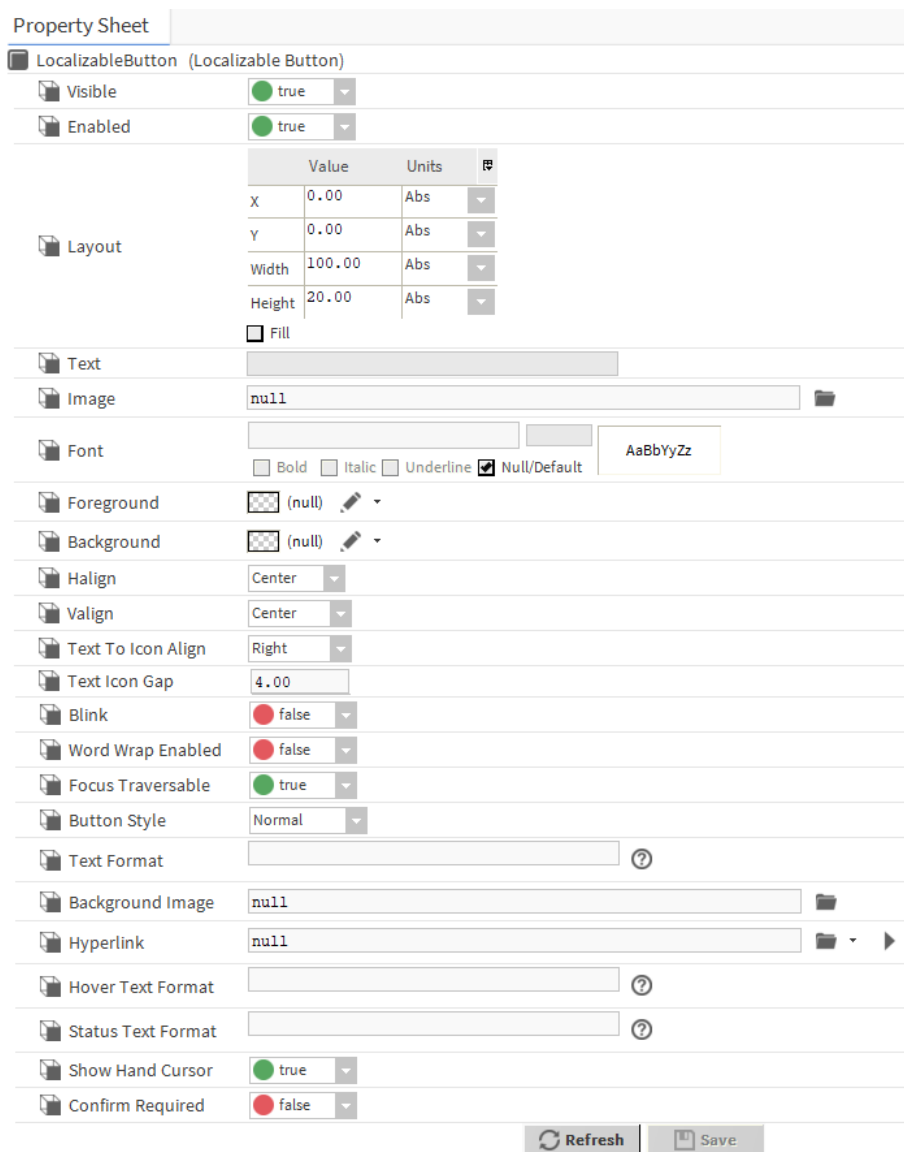
| Property | Value | Description |
|----------------------|-------------------------------------|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Column Count | number | Specifies the number of column in the button group. |
| Halign | Left, Center (default), Right, Fill | Selects from among horizontal alignment options. |
| Valign | Left, Center (default), Right, Fill | Specifies how to vertically align the pane. |
| Row Align | Top, Center (default), Bottom, Fill | Specifies the alignment of the row. |
| Column Align | Top, Center, Bottom, Fill(default) | Specifies the alignment of the column. |
| Row Gap | number | Specifies the gap between the rows. |
| Column Gap | number | Specifies the gap between the columns. |
| Uniform Row Height | true false (default) | When set to true sets uniform row height. When set to false does not have uniform row height. |
| Uniform Column Width | true false (default) | When set to true sets uniform column width. When set to false does not have uniform column width. |
| Stretch Row | number | Allows to stretch the row. |
| Stretch Column | number | Allows to stretch the column. |
| Color Rows | true false (default) | When set to true allows to color the rows. When set to false does not allows to color the rows. |

| Property | Value | Description |
|------------|--|---|
| Band Brush | Solid, Gradient, Image ,Null (default) | Specifies row fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no row fill. |
| b | additional properties | Configures additional parameters for the binding for the button group. |


LocalizableButton

LocalizableButton allows localizable text to be used without requiring a binding. Animate the text property if you want the button bound to a value. This is an extension to the existing BButton and a few more properties such as there is no need to bind to provide localized text.

Figure 129 LocalizableButton Property Sheet



You access these properties by expanding **KitPx** palette and double-click **LocalizableButton**

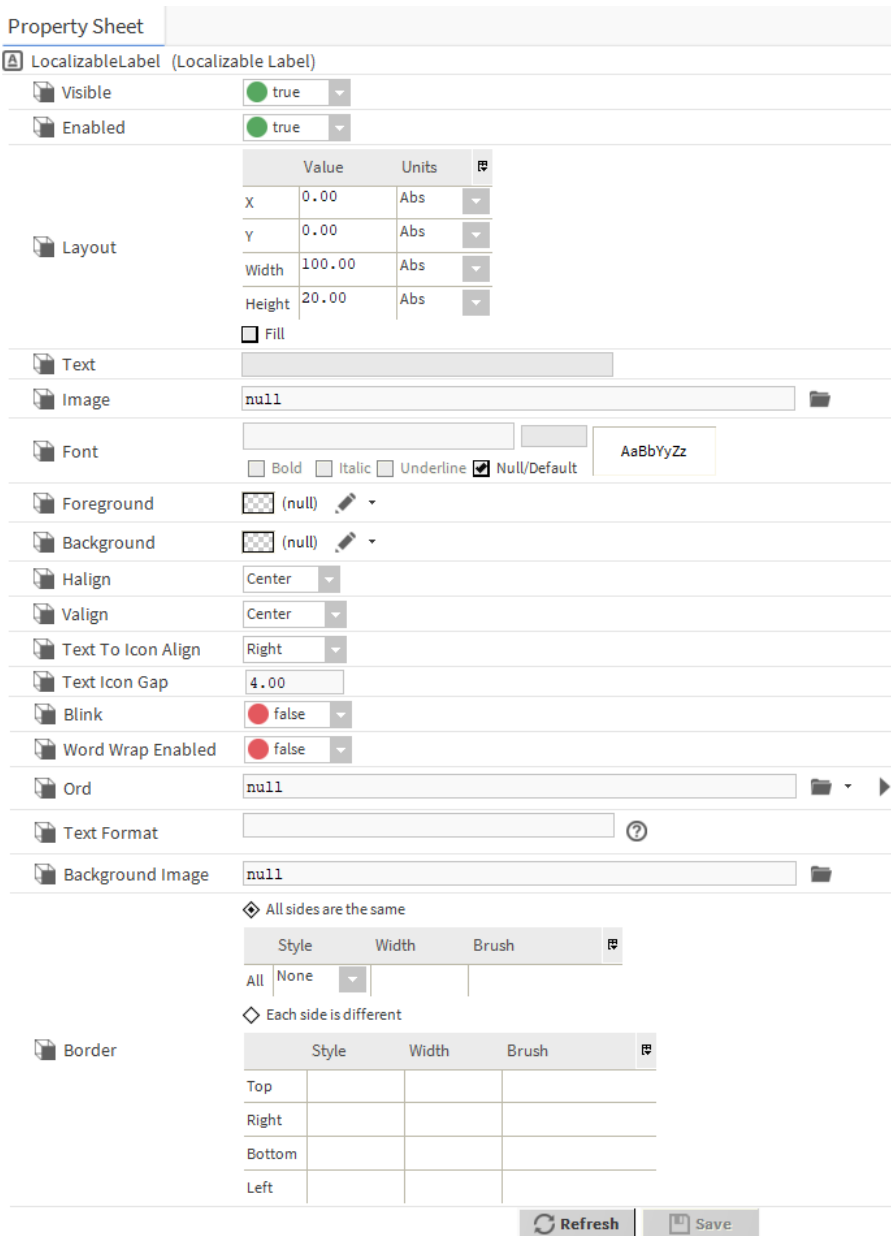
| Property | Value | Description |
|--------------------|---|---|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Text | text | Specifies the text on the label. |
| Image | file chooser | Allows to choose the image for the label. |
| Font | additional properties | Configures the additional parameters for the font on the label. |
| Foreground | Solid, Gradient, Image, Null (default) | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, or d chooser to select the image file. Null indicates no foreground fill. |
| Background | Solid, Gradient, Image, Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |
| Halign | Left, Center (default), Right, Fill | Selects from among horizontal alignment options. |
| Valign | Left, Center (default), Right, Fill | Specifies how to vertically align the pane. |
| Text To Icon Align | Left, Center, Right (default), Top, Bottom | Defines the alignment of the text in relationship to the icon: Right, Top, Bottom, Left, Center |
| Text To Icon gap | number | Defines the gap between the text and icon. |
| Blink | true false (default) | Configures the background color to blink. |
| Word Wrap Enabled | true false (default) | Allows a long word on a button to wrap (true) or prevents the word from wrapping (false). |
| Focus Trasversable | true false (default) | |
| Button Style | Normal (default), Tool Bar, Hyperlink, None | Selects the button style: Normal, Tool Bar, Hyperlink, None |
| Text Format | text | Defines the text format for the reboot button. |

| Property | Value | Description |
|--------------------|-------------------------|---|
| Background image | file chooser | Allows to add background image for the button. |
| Hyperlink | drop-down list | Allows to select the option to insert hyperlink for the button. |
| Hover Text Format | text | Specifies the text format for hover text on the button. |
| Status Text Format | text | |
| Show Hand Cursor | true (default) false | When set to <code>true</code> shows the hand cursor. When set to <code>false</code> does not show the hand cursor. |
| Confirm Required | true false (default) | When set to <code>true</code> asks for confirmation for reboot. When set to <code>false</code> does not asks for confirmation for reboot. |

LocalizableLabel


LocalizableLabel allows localizable text to be used without requiring a binding. Animate the text property if you want the label bound to a value. This is an extension to the existing BButton and a few more properties such as there is no need to bind to provide localized text.

Figure 130 LocalizableLabel Property Sheet



You access these properties by expanding **KitPx** palette and double-click **Localizablelabel**

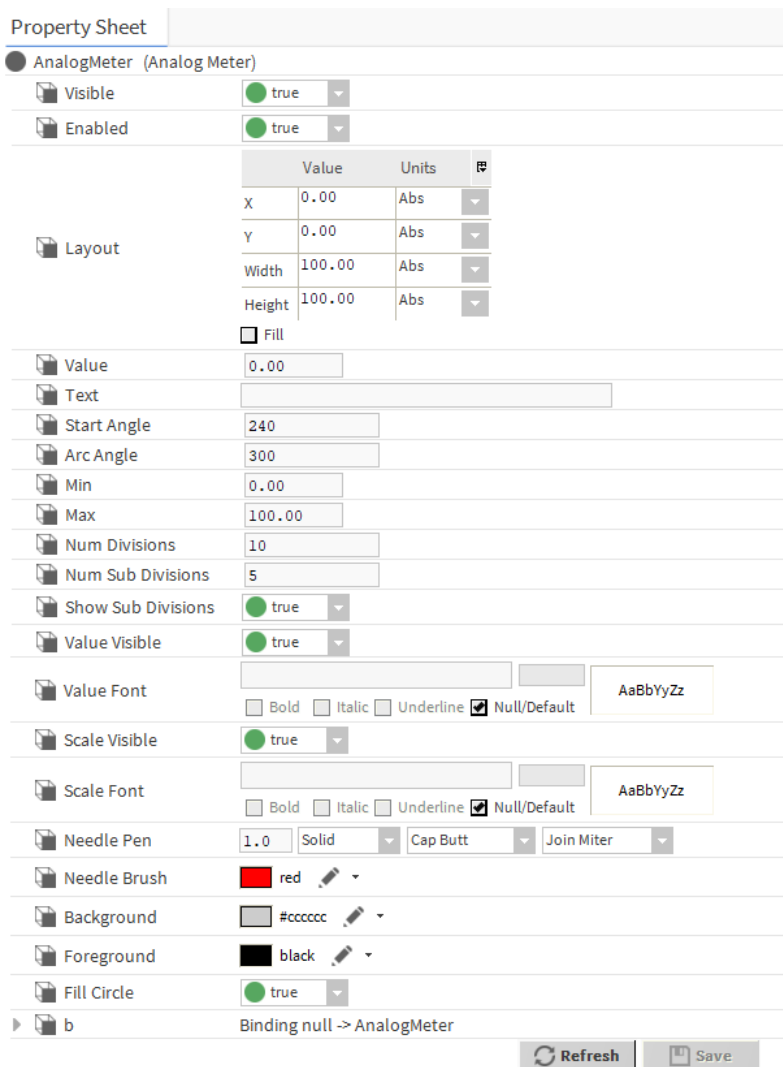
| Property | Value | Description |
|----------|-------------------------|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<i>true</i>) or not (<i>false</i>). |
| Enabled | true (default) false | When set to <i>true</i> , the table in Px page interface is commandable using the popup menu. When set to <i>false</i> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |

| Property | Value | Description |
|--------------------|--|---|
| Text | text | Specifies the text on the label. |
| Image | file chooser | Allows to choose the image for the label. |
| Font | additional properties | Configures the additional parameters for the font on the label. |
| Foreground | Solid, Gradient, Image, Null (default) | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no foreground fill. |
| Background | Solid, Gradient, Image, Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |
| Halign | Left, Center (default), Right, Fill | Selects from among horizontal alignment options. |
| Valign | Left, Center (default), Right, Fill | Specifies how to vertically align the pane. |
| Text To Icon Align | Left, Center, Right (default), Top, Bottom | Defines the alignment of the text in relationship to the icon: Right, Top, Bottom, Left, Center |
| Text To Icon gap | number | Defines the gap between the text and icon. |
| Blink | true false (default) | Configures the background color to blink. |
| Word Wrap Enabled | true false (default) | Allows a long word on a button to wrap (true) or prevents the word from wrapping (false). |
| ord | File chooser | Select the ord for the label. |
| Background image | file chooser | Allows to add background image for the button. |
| Text Format | text | Defines the text format for the reboot button. |
| Border | additional properties | Configures additional parameters for the border of the label. |

AnalogMeter


kitPx-AnalogMeter is a widget that is designed to look and act like a analog meter. This component comes with a value binding that should be bound to a value by an ORD.

Figure 131 AnalogMeter Property Sheet



You access these properties by expanding **KitPx** palette and double-click **AnalogMeter**

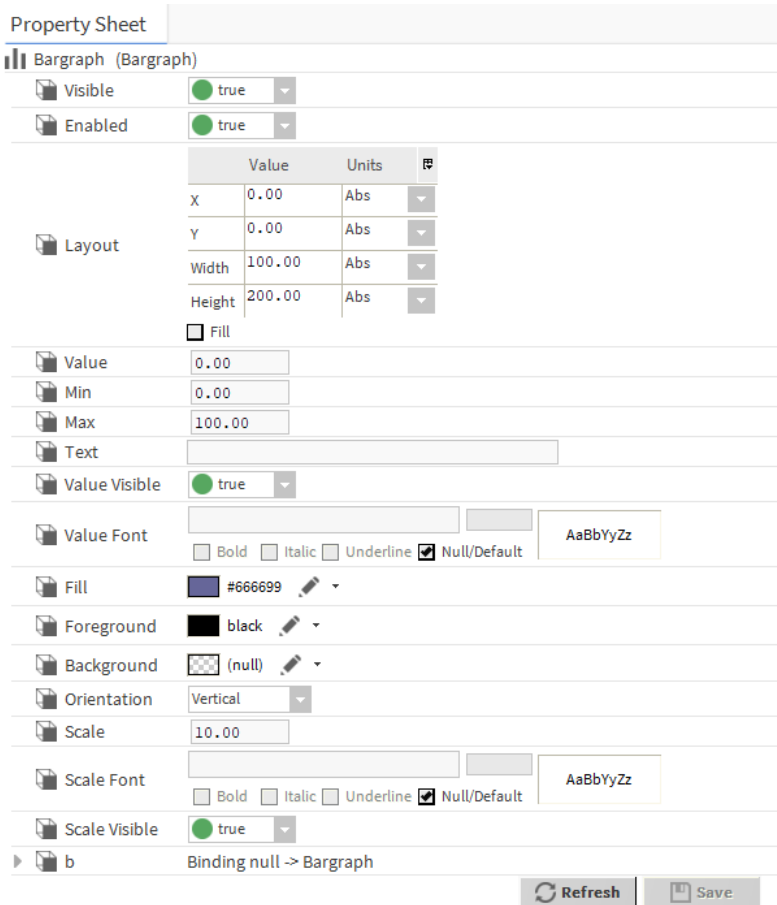
| Property | Value | Description |
|-------------|-------------------------|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<i>true</i>) or not (<i>false</i>). |
| Enabled | true (default) false | When set to <i>true</i> , the table in Px page interface is commandable using the popup menu. When set to <i>false</i> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Value | number | Displays the Number on analog meter |
| Text | text | Specifies the text on the label. |
| Start Angle | number | Defines the start angle dimension. |

| Property | Value | Description |
|--------------------|---|---|
| Arc Angle | number | Defines the arc angle dimension. |
| Min | number | Shows the minimum value that can be entered for the analog meter |
| Max | number | Shows the maximum value that can be entered for the analog meter |
| Num Divisions | number | Defines the number of divisions. |
| Num Sub Divisions | number | Defines the number os subdivisions. |
| Show Sub Divisions | true (default) false | When set to <code>true</code> the sub divisions are visible on the meter. When set to <code>false</code> the sub divisions are not visible on the meter. |
| Value Visible | true (default) false | When set to <code>true</code> the value is visible on the meter. When set to <code>false</code> the value is not visible on the meter. |
| Value Font | additional properties | Configures the additional parameters for the value font on the label. |
| Scale Visible | true (default) false | When set to <code>true</code> the scale is visible on the meter. When set to <code>false</code> the value is not scale on the meter. |
| Scale Font | additional properties | Configures the additional parameters for the scale font on the label. |
| Needle Pen | Additional properties | Configures additional parameters for the needle pen on the meter label. |
| Needle Brush | Solid, Gradient, Image, Null (default) | Specifies a needle color fill for the meter label. <code>Solid</code> opens the Color Chooser window. <code>Gradient</code> opens the Gradient Editor window. <code>Image</code> opens the Texture window, click the Browse icon to open the File Chooser, Ord Chooser, or other method of selecting an image file. <code>Null</code> indicates no fill. |
| Foreground | Solid (default), Gradient, Image, Null(default) | Specifies foreground fill. <code>Solid</code> opens the color chooser window. <code>Gradient</code> opens the gradient editor window. <code>Image</code> opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. <code>Null</code> indicates no foreground fill. |
| Background | Solid, Gradient, Image, Null (default) | Specifies background fill color. <code>Solid</code> opens the Color Chooser window. <code>Gradient</code> opens the Gradient Editor window. <code>Image</code> opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. <code>Null</code> indicates no color (white). |
| Fill Circle | true (default) false | When set to <code>true</code> the circle is filled with the selected color on the meter label. When set to <code>false</code> the circle is not filled with the selected color on the meter label. |
| b | Additional Properties | Configures the additional parameters for the binding on the label. |


Bargraph

kitPx-Bargraph is a widget that is designed to display a single value in a bar graph. In addition to action bindings, value and setpoint bindings are available. To plot more than one value using a single widget (using a line chart) . Or, you can use a chart widget for history tables.

Figure 132 BarGraph Property Sheet



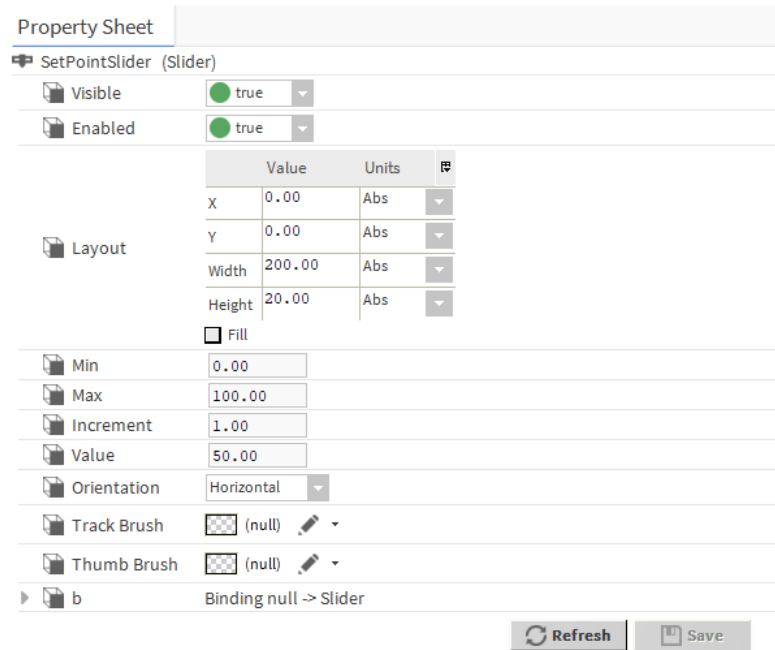
| Property | Value | Description |
|----------|-------------------------|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Value | number | Displays the Number on analog meter |
| Text | text | Specifies the text on the label. |
| Min | number | Shows the minimum value that can be entered for the analog meter |

| Property | Value | Description |
|---------------|---|---|
| Max | number | Shows the maximum value that can be entered for the analog meter |
| Value Visible | true (default) false | When set to <code>true</code> the value is visible on the meter. When set to <code>false</code> the value is not visible on the meter. |
| Foreground | Solid (default), Gradient, Image ,Null(default) | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no foreground fill. |
| Background | Solid, Gradient, Image ,Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |
| Fill | Solid, Gradient, Image ,Null (default) | Specifies color fill of the pane . Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no background fill. |
| Orientation | Vertical (de- fault),Horizontal | Specifies the orientation of the graph. |
| Scale | number | Defines the scale of the graph. |
| Scale Visible | true (default) false | When set to <code>true</code> the scale is visible on the meter. When set to <code>false</code> the value is not scale on the meter. |
| Scale Font | additional properties | Configures the additional parameters for the scale font on the label. |
| b | Additional Properties | Configures the additional parameters for the binding on the label. |

SetPointSlider

SetPointSlider, in the `kitPx` palette, provides a visual slider which is used to select and integer or floating point value between a fixed range.

Figure 133 SetPointSlider Property Sheet



You access these properties by expanding **KitPx** palette and double-click **SetPointSlider**

| Property | Value | Description |
|-------------|--|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Min | number | Shows the minimum value that can be entered for the analog meter |
| Max | number | Shows the maximum value that can be entered for the analog meter |
| Value | number | Displays the Number on slider |
| Increment | number | Increases the size of the slider by the dimension provided. |
| Orientation | Vertical (default),Horizontal | Specifies the orientation of the graph. |
| Track Brush | Solid, Gradient, Image ,Null (default) | Specifies a track color fill for the slider. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Texture window, click the Browse icon to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no fill. |

| Property | Value | Description |
|-------------|--|--|
| Thumb Brush | Solid, Gradient, Image, Null (default) | Specifies a thumb color fill for the slider. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Texture window, click the Browse icon to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no fill. |
| b | Additional Properties | Configures the additional parameters for the binding on the label. |

SetPointToggleButton

SetPointToggleButton, in the `kitPx` palette, provides a two-state widget which may be selected and unselected.

Figure 134 SetPointToggleButton Property Sheet

Property Sheet


SetPointToggleButton (Toggle Button)

- Visible: true
- Enabled: true
- Layout:

| | Value | Units | # |
|--------|--------|-------|---|
| x | 0.00 | Abs | |
| y | 0.00 | Abs | |
| Width | 100.00 | Abs | |
| Height | 40.00 | Abs | |
- Fill:
- Text:
- Image: null
- Font: AaBbYyZz
 - Bold
 - Italic
 - Underline
 - Null/Default
- Foreground: (null)
- Background: (null)
- Halign: Center
- Valign: Center
- Text To Icon Align: Right
- Text Icon Gap: 4.00
- Blink: false
- Word Wrap Enabled: false
- Focus Traversable: true
- Button Style: Normal
- Selected: false
- b: Binding null -> ToggleButton

Refresh Save

You access these properties by expanding **KitPx** palette and double-click **SetPointToggleButton**.

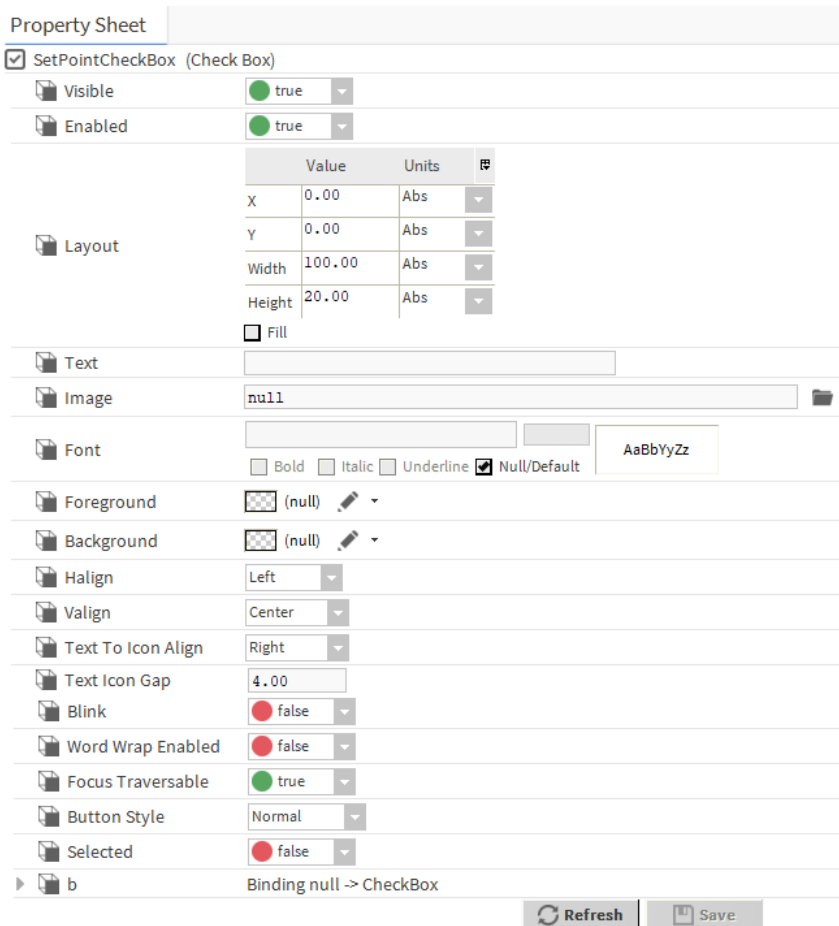
| Property | Value | Description |
|--------------------|---|---|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Text | text | Specifies the text on the label. |
| Image | file chooser | Allows to choose the image for the label. |
| Font | additional properties | Configures the additional parameters for the font on the label. |
| Foreground | Solid, Gradient, Image, Null (default) | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, or Ord Chooser to select the image file. Null indicates no foreground fill. |
| Background | Solid, Gradient, Image, Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |
| Halign | Left, Center (default), Right, Fill | Selects from among horizontal alignment options. |
| Valign | Left, Center (default), Right, Fill | Specifies how to vertically align the pane. |
| Text To Icon Align | Left, Center, Right (default), Top, Bottom | Defines the alignment of the text in relationship to the icon: Right, Top, Bottom, Left, Center |
| Text To Icon gap | number | Defines the gap between the text and icon. |
| Blink | true false (default) | Configures the background color to blink. |
| Word Wrap Enabled | true false (default) | Allows a long word on a button to wrap (true) or prevents the word from wrapping (false). |
| Focus Trasversable | true false (default) | |
| Button Style | Normal (default), Tool Bar, Hyperlink, None | Selects the button style: Normal, Tool Bar, Hyperlink, None |

| Property | Value | Description |
|----------|-----------------------|--|
| Selected | true false (default) | |
| b | Additional Properties | Configures the additional parameters for the binding on the label. |

SetPointCheckBox


SetPointCheckBox, in the `KitPx` palette, is a specialized `ToggleButton` which displays its label next to a box which can be checked and unchecked.

Figure 135 SetPointCheckBox Property Sheet



You access these properties by expanding `KitPx` palette and double-click `SetPointCheckBox`

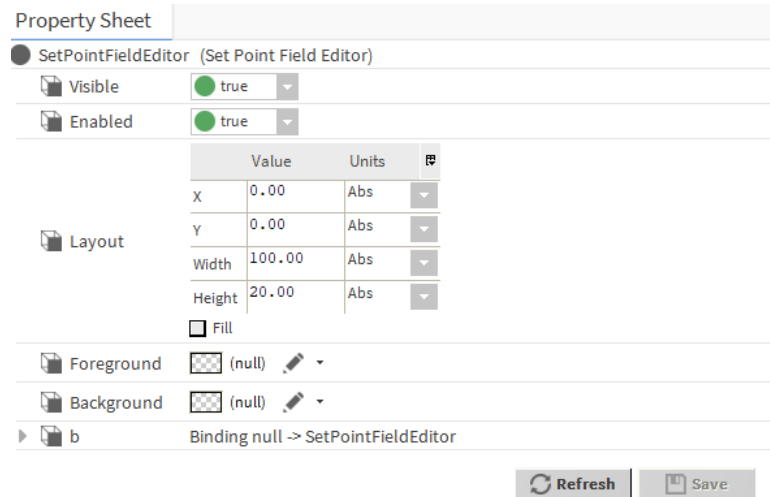
| Property | Value | Description |
|----------|----------------------|--|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<code>true</code>) or not (<code>false</code>). |
| Enabled | true (default) false | When set to <code>true</code> , the table in Px page interface is commandable using the popup menu. When set to <code>false</code> , the display is still visible but not commandable. |

| Property | Value | Description |
|--------------------|---|---|
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Text | text | Specifies the text on the label. |
| Image | file chooser | Allows to choose the image for the label. |
| Font | additional properties | Configures the additional parameters for the font on the label. |
| Foreground | Solid, Gradient, Image, Null (default) | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no foreground fill. |
| Background | Solid, Gradient, Image, Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |
| Halign | Left, Center (default), Right, Fill | Selects from among horizontal alignment options. |
| Valign | Left, Center (default), Right, Fill | Specifies how to vertically align the pane. |
| Text To Icon Align | Left, Center, Right (default), Top, Bottom | Defines the alignment of the text in relationship to the icon: Right, Top, Bottom, Left, Center |
| Text To Icon gap | number | Defines the gap between the text and icon. |
| Blink | true false (default) | Configures the background color to blink. |
| Word Wrap Enabled | true false (default) | Allows a long word on a button to wrap (<i>true</i>) or prevents the word from wrapping (<i>false</i>). |
| Focus Trasversable | true false (default) | |
| Button Style | Normal (default), Tool Bar, Hyperlink, None | Selects the button style: Normal, Tool Bar, Hyperlink, None |
| Selected | true false (default) | |
| b | Additional properties | Configures the additional parameters for the binding on the label. |


SetPointFieldEditor

kitPx-SetPoinFieldEditor is a special field editor that is designed to allow you to edit setpoint values from a graphic view. It is designed for layout on panes and is comprised of standard widgets like buttons, text fields, and check boxes. For editing non-specific properties, use the kitPx GenericField-Editor.

Figure 136 SetPointFieldEditor Property Sheet



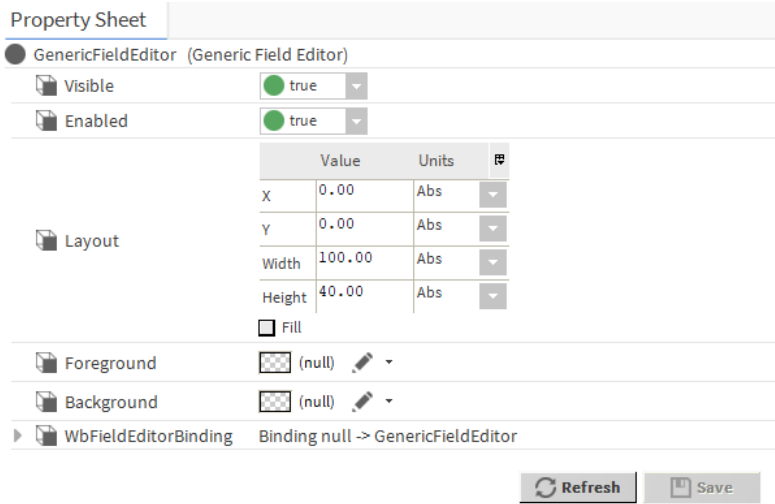
You access these properties by expanding **KitPx** palette and double-click **SetPointFieldEditor**

| Property | Value | Description |
|------------|--|---|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (<i>true</i>) or not (<i>false</i>). |
| Enabled | true (default) false | When set to <i>true</i> , the table in Px page interface is commandable using the popup menu. When set to <i>false</i> , the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Foreground | Solid, Gradient, Image, Null (default) | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, ord chooser to select the image file. Null indicates no foreground fill. |
| Background | Solid, Gradient, Image, Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |
| b | Additional Properties | Configures the additional parameters for the binding on the label. |


GenericFieldEditor

kitPx-GenericFieldEditor is a field editor that allows you to edit properties from a graphic view on non-specific fields. It is designed for layout on panes and is comprised of standard widgets like buttons, text fields, and check boxes. Specialized field editor widgets include: SetPointFieldEditor.

Figure 137 GenericFieldEditor Property Sheet



You access these properties by expanding **KitPx** palette and double-click **GenericFieldEditor**

| Property | Value | Description |
|-----------------------|--|---|
| Visible | true (default) false | Sets the table to be visible in the Px page interface (true) or not (false). |
| Enabled | true (default) false | When set to true, the table in Px page interface is commandable using the popup menu. When set to false, the display is still visible but not commandable. |
| Layout | additional properties | Opens a layout window for specifying the size and location of the table in relation to its parent (X, Y coordinates, height and width). |
| Foreground | Solid, Gradient, Image, Null (default) | Specifies foreground fill. Solid opens the color chooser window. Gradient opens the gradient editor window. Image opens the texture window, click the browser icon to open File chooser, or d chooser to select the image file. Null indicates no foreground fill. |
| Background | Solid, Gradient, Image, Null (default) | Specifies background fill color. Solid opens the Color Chooser window. Gradient opens the Gradient Editor window. Image opens the Image Brush Editor window. Click the Browse icon () to open the File Chooser, Ord Chooser, or other method of selecting an image file. Null indicates no color (white). |
| WbFieldEditor-Binding | Additional Properties | Configures the additional parameters for the binding on the label. |

ActionBinding

kitPx-ActionBinding invokes an action on the binding target component when an event is fired by the parent widget. The ORD of an action binding must resolve down to a specific action within a component.

Figure 138 ActionBinding Property Sheet



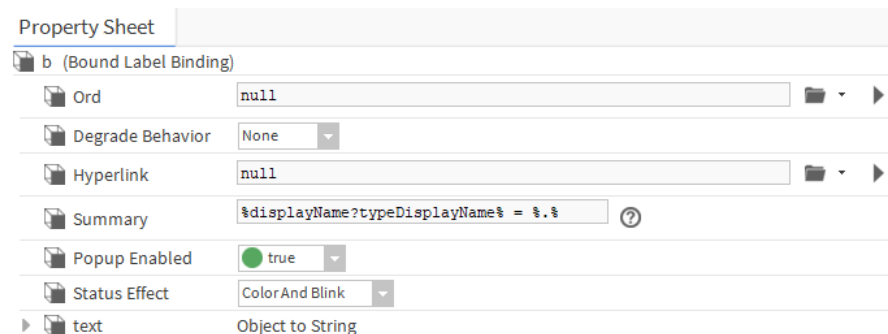
You access these properties by expanding **KitPx** palette and right-click **LocalizableButton**, double click **ActionBinding**

| Property | Value | Description |
|------------------|----------------|--|
| ord | Browse | Select the folder path for the binding. |
| Degrade Behavior | Drop-down list | Select from the drop- down to specify the degrade behaviour. |
| Widget Event | text | Specifies the widget event. |
| Action Arg | text | |

BoundLabelBinding

kitPx-BoundLabelBinding is used exclusively for connecting a value to a bound label widget. Bound labels, which are located in the kitPx module, have properties that you can edit and access from the PxBrowser properties side bar.

Figure 139 BoundLabelBinding Property Sheet



You access these properties by expanding **KitPx** palette and double-click **LocalizableButton** double click **BoundLabelBinding**

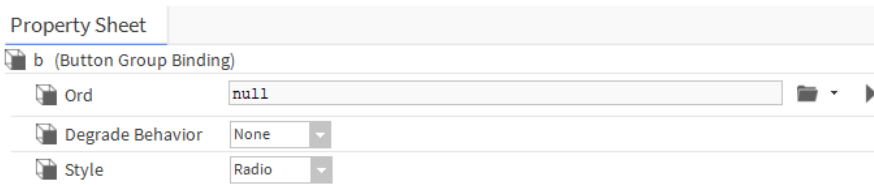
| Property | Value | Description |
|------------------|----------------|--|
| ord | Browse | Select the folder path for the binding. |
| Degrade Behavior | Drop-down list | Select from the drop- down to specify the degrade behaviour. |
| Hyperlink | Browse | Select the folder path for the hyperlink. |

| Property | Value | Description |
|---------------|-------------------------|---|
| Summary | text | Displays the summary of the binding. |
| Popup Enabled | true(default), or false | When set to true allows the binding popup to populate. When set to false disallows the binding popup to populate. |
| Status Effect | drop-down list | Select the status of effect from the list to be displayed. |
| Text | additional properties | |

ButtonGroupBinding

kitPx-ButtonGroupBinding is used exclusively for connecting a value to a ButtonGroup widget. Button-Groups, which are located in the kitPx module, have properties that you can edit and access from the Px Editor properties side bar.

Figure 140 ButtonGroupBinding Property Sheet



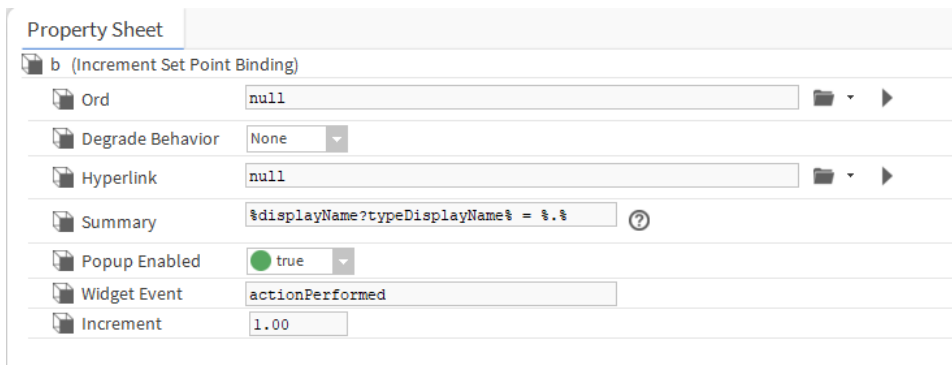
You access these properties by expanding KitPx palette and expand ButtonGroupdouble click **b (Button Group Binding)**

| Property | Value | Description |
|------------------|----------------|--|
| ord | Browse | Select the folder path for the binding. |
| Degrade Behavior | Drop-down list | Select from the drop- down to specify the degrade behaviour. |
| Style | Drop-down list | Select the style of the button from the drop-down list. |

IncrementSetPointBinding

kitPx-IncrementSetPointBinding is used to increment (increase) or decrement (decrease) a setpoint value.

Figure 141 IncrementSetPointBinding Property Sheet



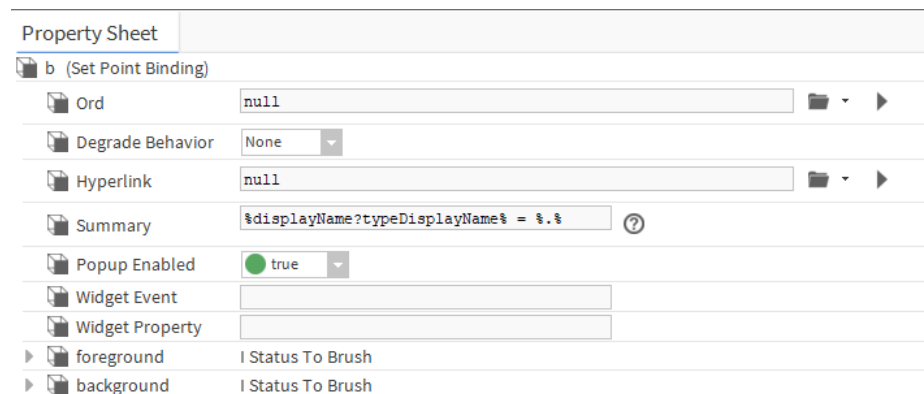
You access these properties by expanding **KitPx** palette and expand **IncrementSetPointButton** double click **Increment Set Point Binding**

| Property | Value | Description |
|------------------|-------------------------|---|
| ord | Browse | Select the folder path for the binding. |
| Degrade Behavior | Drop-down list | Select from the drop- down to specify the degrade behaviou. |
| Hyperlink | Browse | Select the folder path for the hyperlink. |
| Summary | text | Displays the summary of the binding. |
| Popup Enabled | true(default), or false | When set to true allows the binding popup to populate. When set to false disallows the binding popup to populate. |
| widget | text | |
| Increment | number | |

SetPointBinding

kitPx-SetPointBinding is used to display the current value of a setpoint and also to provide the ability to modify it. The setpoint binding ORD must resolve down to the specific property that is being manipulated.

Figure 142 SetPointBinding Property Sheet



You access these properties by expanding **KitPx** palette and expand **SetPointFieldEditor** double click **Set Point Binding**

| Property | Value | Description |
|------------------|-------------------------|---|
| ord | Browse | Select the folder path for the binding. |
| Degrade Behavior | Drop-down list | Select from the drop- down to specify the degrade behaviou. |
| Hyperlink | Browse | Select the folder path for the hyperlink. |
| Summary | text | Displays the summary of the binding. |
| Popup Enabled | true(default), or false | When set to true allows the binding popup to populate. When set to false disallows the binding popup to populate. |
| Widget Event | text | |
| Widget Property | text | |

| Property | Value | Description |
|------------|----------------|--|
| foreground | drop-down list | Select mode from drop-down list whether to display the current value of a setpoint in foreground or in background. |
| background | drop-down list | Select mode from drop-down list whether to display the current value of a setpoint in foreground or in background. |

Components in pxeditor module

NewPxViewDialog

pxeditor-NewPxViewDialog

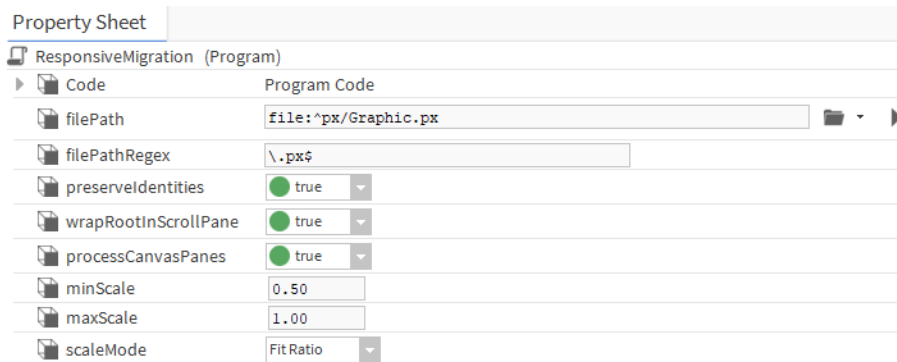
NewPxViewDialog allows you to create a new Px view.

ResponsiveMigration

If you have existing Px graphics, running this program object will make them mobile friendly.

Available in Niagara 4.6 and later, this programObject can process a folder full of existing Px files (or a single existing Px file), and apply responsive rules to the graphics. This process optimizes the display when the graphics are viewed on mobile devices using HxPx.

Figure 143 ResponsiveMigration properties



This programObject is available in the `pxEditor` palette.

Properties

| Name | Value | Description |
|---------------------------------|--------------------------|--|
| <code>filePath</code> | <code>file:^px</code> | Indicates the folder of Px files (or an individual Px file) to be optimized. |
| <code>filePathRegex</code> | <code>.px\$</code> | Filters all but the Px files for indicated folders |
| <code>preserveIdentities</code> | true (default), false | Enable/disable this property. Equivalent to the Preserve Identities setting under the Px Editor section in Workbench Options . If true, the names of the widgets in your Px pages will be preserved in the Px file. If false, they will be left out. In general, this should be set to the same value with which the Px pages were originally saved. Note that any pages that include Px Properties will automatically have identities preserved. If unsure, leave set to true. |

| Name | Value | Description |
|-----------------------------|--|--|
| wrapRootInScrollPane | true (default), false | Enable/disable this property. Ensures that the root element of your Px page is a ScrollPane, which will allow the CanvasPane root to resize down to a smaller form factor, but still scroll to allow the entire graphic to be viewed without shrinking the graphic down beyond a reasonable limit. |
| processCanvasPanes | true (default), false | Enable/disable this property. Sets the min and max scale factors of the main CanvasPane, which defines how small or large the CanvasPane may scale in order to fit different screen sizes. |
| minScale | 0.50 (default) | Range is 0.0–1.0. Typically between 0 (allow it to shrink to nothing) and 1 (do not allow it to shrink at all). |
| maxScale | 1.0 (default) | Range is 1.0-2.0, although the upper limit is not enforced. Use a value of 1.0 (do not allow it to grow at all) or up to some reasonable upper limit (for example, 2.0 would allow it to grow up to twice its defined size). |
| scaleMode | None, Fit, Fit Ratio (default), Fit Width, Fit Height | Defines how the aspect ratio of the Px graphic will change as it grows or shrinks. This would most commonly be Fit Ratio, which will preserve the graphic's aspect ratio. Another option is Fit, which would cause the graphics to fill the entire available screen space, at the cost of some stretching. |

Actions

- **Execute**

Executes the programObject.

- **Dry Run**

list out all the matching files in the target directory and describe exactly what changes will be made. This information will be logged to the `com.tridium.px.editor.util` log, so use the Logger Configuration view for the DebugService to add log and adjust the log level, if needed. Execute the programObject a second time with Dry Run turned Off to commit the changes to the file system.

Command-line Interface

Migration can also be performed from the command line, to avoid having to do an unsafe migration against a running station. Enter the following command to invoke it (line breaks added for clarity):

```
nre.exe workbench:com.tridium.workbench.px.PxFileMigrationUtil
-preserve-identities -process-canvas-panes -wrap-in-scroll-pane
-file-path:c:/myDevDirectory/myPxFiles -max-scale:1.5 -min-scale:0.5
-scale:fitRatio "-regex:\.px$"
```

To commit changes, add the `-overwrite-files` flag; otherwise, it defaults to a Dry Run.

All the command-line flags correspond to the different parameters that can be configured on this program object; for the boolean parameters, omitting them entirely will correspond to false.

Components in report module

BqlGrid

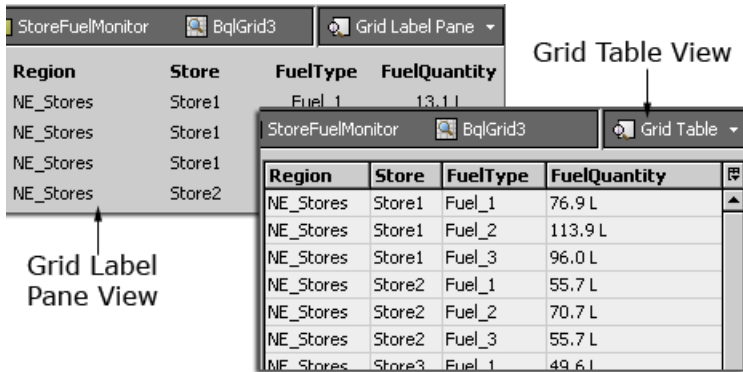
The BqlGrid component provides a means for using a Bql Query and the BFormat syntax to define a dynamic table layout.

Use the BqlGrid component to specify an OrdTarget and columns of values to build a dynamic Grid Table or Grid Label Pane view. The BqlGrid component subscribes to the specified values for dynamic updating. The BqlGrid component allows you to create a Grid Table or Grid Label Pane view of any collection of data that you can access with a Bql query and the BFormat syntax. The BqlGrid component is available in the Reporting folder of the **report** palette.

NOTE: This does not work with histories.

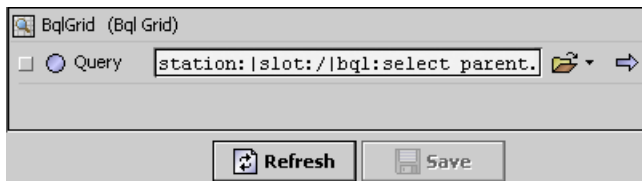
In addition to the standard wire sheet, category sheet, slot sheet, and property sheet views, the BqlGrid-Component property includes a Grid Table and Grid Label Pane view, as shown here:

Figure 144 BqlGrid Label Pane view



The BqlGrid component property sheet view is shown here and described below:

Figure 145 BqlGrid property sheet view



- **Query**

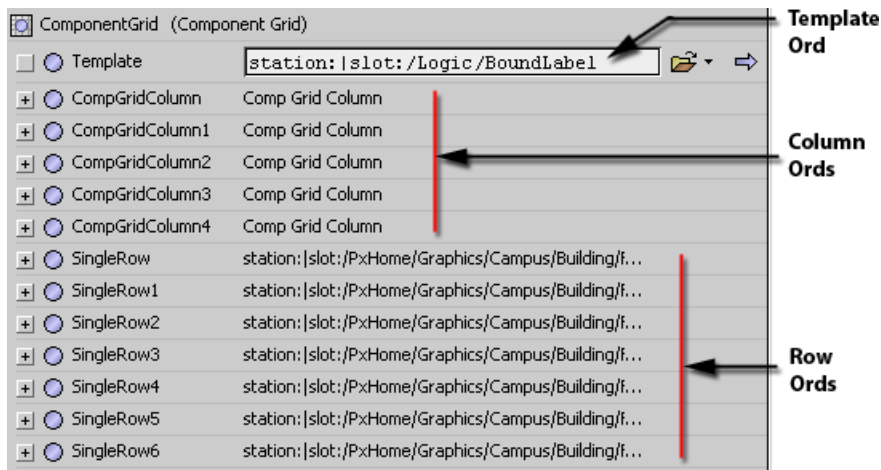
This single property is used to define a TargetOrd using a Bql query. Every column in the resultant row points to the same Ord. A value for each desired column in the row is provided by using BFormat syntax to point to the corresponding column from the Bql query. This is different from the ComponentGrid component where each table cell is its own OrdTarget.

ComponentGrid

The component grid allows you to define and design your report content (typically in the Grid Editor view).

The ComponentGrid allows you to define and design your report content (typically in the Grid Editor view). Use the component grid to link and layout the data that you want to put into your report. The Component-Grid component is available in the Reporting folder of the **report** palette.

Figure 146 ComponentGrid property sheet view



The component grid has most of the standard Workbench views, such as Property Sheet view (shown above), wire sheet, category sheet, slot sheet, and link sheet. Unique ComponentGrid views include the following:

- **Grid Table**

This view displays your linked data in a typical table format.

- **Grid Label pane**

This view displays your linked data in a tabular view that uses the GridLabelPane widget for displaying the report data.

- **Grid Editor**

This is the view that you use to specify row and column data for your report.

EmailRecipient

The **EmailRecipient** component contains properties that allow you to specify where the report is to be emailed.

report-EmailRecipient

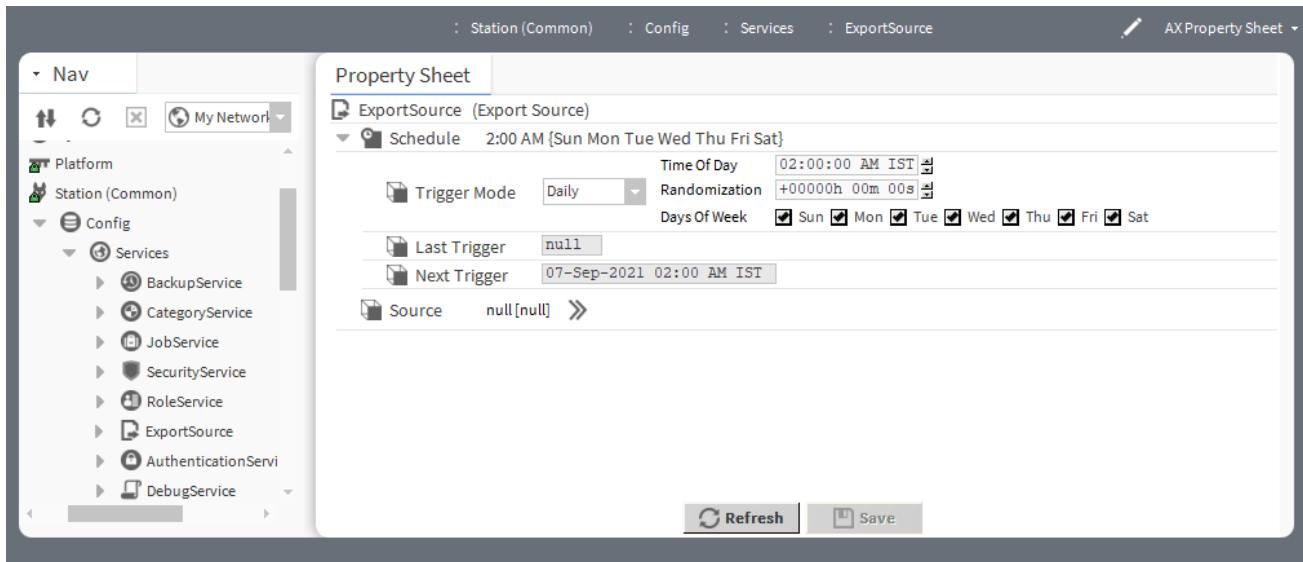
The property sheet view displays the properties that you configure for this function. The **EmailRecipient** component is available in the Reports folder in the **report** palette.

ExportSource

The Export Source component should be located in the ReportService container. It allows you to define a report display source (typically a Px file) for exporting and it allows you to schedule a time that you want to export it.

The property sheet view, shown below, contains the properties that you configure for these functions. This component is available in the **report** palette.

Figure 147 Export Source component property sheet view



- **Schedule**

This property contains standard scheduling options that allow you to choose different options for timing the delivery of your report. It also displays a **Last Trigger** and **Next Scheduled Trigger** field.

- **Source**

This property specifies the report display that you want to email. An arrow button located at the right end of the property display opens the **Export Source Wizard** window. This window provides two steps that assist you in assigning source location and selecting a PDF export or an oBix driver export for your report.

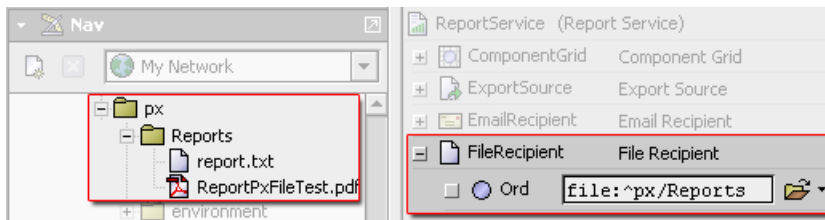
FileRecipient

The file recipient component allows you to save a report to a location under a station file system.

This component contains a single property that allows you to specify a location (file directory) for saving the report. The property sheet view, shown below, displays the property that you configure for this function.

CAUTION: Saving excessively to a flash drive can reduce drive life prematurely. Do not overschedule report saves to a flash drive. Saving to a hard drive does not have such limitations.

Figure 148 FileRecipient component specifying location for generated report file



FileRecipient component properties include the following:

- **Ord**

This property provides a text field that allows you to specify the directory for saving the generated report file.

ReportPane

The ReportPane widget is located in the ReportWidgets folder of the **report** palette and provides a container for layout of Px page report views.

The ReportPane has the standard Visible, Enabled, and Layout properties, but also has the following unique properties:

Figure 149 ReportPane property sheet view

Property Sheet

ReportPane (Report Pane)

Visible true

Enabled true

| | Value | Units |
|--------|-------|-------|
| X | | |
| Y | | |
| Width | | |
| Height | | |

Fill

Logo

Page # true

Timestamp true

Row Gap

- **Logo**

This property supports the linking of a graphic to the ReportPane component. When displayed in a Px page, or a Report, the graphic originates at the top left corner of the page. Type or browse to the desired graphic to set the file path in the property field. If no logo is used, the default value of null should be set in the property field.

- **Page#**

When this property value is set to `True`, page numbering displays on each Report page.

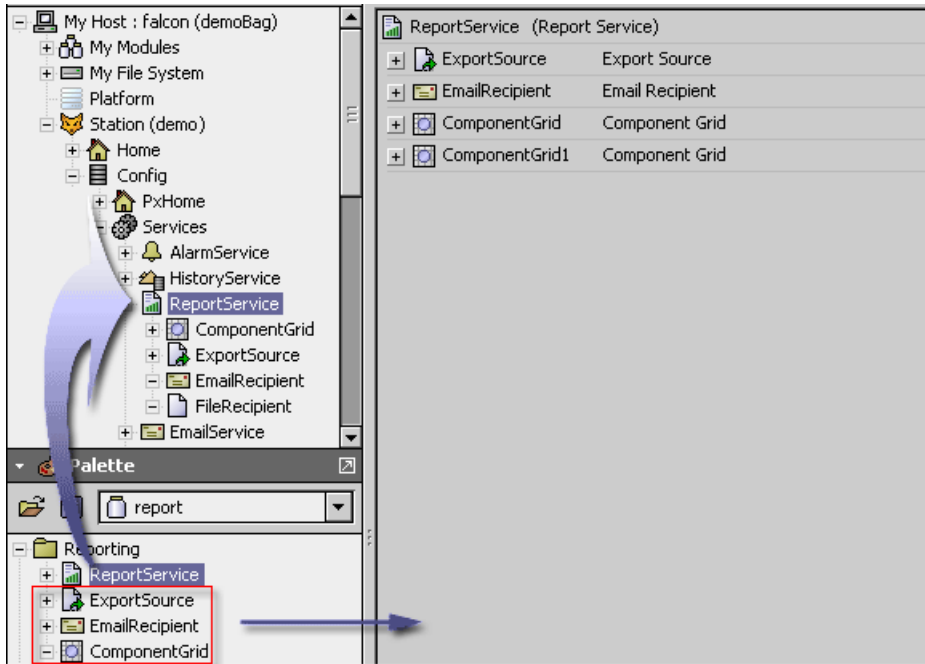
- **Timestamp**

When this property value is set to `True`, a timestamp is displayed in the bottom right corner of the report pages.

ReportService

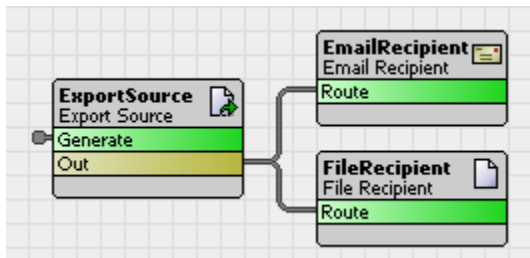
ReportService is the parent, or container, for the ExportSource, EmailRecipient, ComponentGrid, and FileRecipient components. In order to use the report service, copy the ReportService component from the **report** palette to the Services folder in the Nav Tree, as shown.

Figure 150 Copy the report service to the services directory and add report components



You can use the ReportService wire sheet view to link generated reports to the desired recipients, as shown in here.

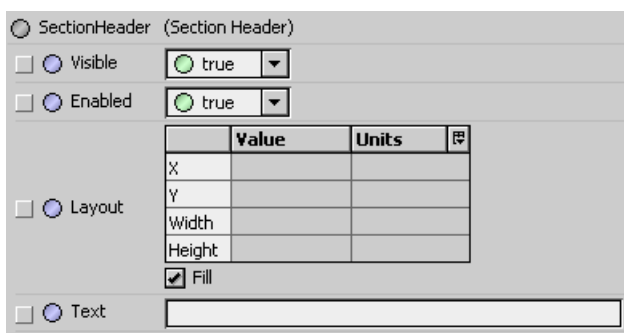
Figure 151 ReportService component Wire Sheet view



SectionHeader

The SectionHeader component is used to put a title or heading at the top of a report. It is located in the ReportWidgets folder in the report palette.

Figure 152 Section header property sheet



Plugins in pxEditor module

Summary information on views of the PxEditor component.

pxEditor

The PxEditor provides tools for creating and editing Px files.

pxEditor-PxEditor

The **Px Editor** is the view that is used to create graphic presentations in Niagara.

Plugins in report module

Summary information on views of ComponentGrid component.

GridTable

The **Grid Table** view is a view of the **ComponentGrid** component.

report-GridTable

This view provides a tabular display of all the points that are assigned to the ComponentGrid.

GridLabelPane

The Grid Label Pane view is a view of the **ComponentGrid** component.

report-GridLabelPane

This view provides a tabular display of all the points that are assigned to the ComponentGrid. In the Grid Table Plane view, you can write to points (issue commands) using the popup menu, if you have write permissions.

ComponentGridEditor

The Grid Editor view is a view of the **ComponentGrid** component.

report-ComponentGridEditor

This view provides a layout display that you use to assign points that you are going to use in a report.

Index

>binding properties
types 54

A

aboutexpandablePane 112
action bindings 59
addanadminrole 92
Alignment 104
animate
 using Popup Binding 32
Animate
 using static SVGs 51
Animating graphics
 about 49

B

bajau
 components 137
 bajau widgets 17
Basic Kiosk 98
bindings 32
 field editor 60
Bindings
 action 59
 bound label 56
 increment setpoint 58
 Popup 60
 relative vs. absolute 61
 spectrum 57
 spectrum setpoint 59
 table 60
 types 53
 value 56
 converters 57
Bound label binding
 about 56
BqlGrid
 component 219

C

Canvas
 grid
 show hatch
 snap 103
chart
 components 169
ComponentGrid
 component 220
components 137
 bajau 137

bajau-BoundTable 137
EmailRecipient (alarm) 83
Components 199, 201
 bajau 137, 140, 148, 154, 157–158
 CanvasPane 155
 ExpandablePane 158
 FlowPane 159
 GridPane 162
 HyperlinkLabel 145
 Label 143
 Line 149
 Path 150
 Picture 147
 Polygon 152
 PxInclude 148
 RadioButton 141
 Rect 153
 ResponsivePane 160
 ScrollPane 164
 Separator 147
 SplitPane 165
 TabbedPane 167
 TextEditorOptions 167
 ValueBinding 169
 BorderPane 154
 Button 140
 chart 169
 BarChart 169
 ChartCanvas 171
 ChartHeader 172
 ChartPane 173
 DefaultChartLegend 175
 LineChart 175
 ConstrainedPane 157
 EdgePane 158
 Ellipse 148
 gx 176
 Brush 176
 kitPx 176, 211
 ActionBinding 215
 ActionButton 179
 AnalogMeter 203
 BackButton 189
 Bargraph 206
 BooleanImage 176
 BoundLabel 176
 BoundLabelBinding 215
 ButtonGroup 198
 ButtonGroupBinding 216
 DecrementSetPointButton 179
 EnumImage 176
 ExportButton 187
 ForwardButton 191
 GenericFieldEditor 214
 IncrementSetPointBinding 216

- LogoffButton 193
 - NumericImage 176
 - Rebootbutton 195
 - RefreshButton 185
 - SaveButton 184
 - SetPointBinding 217
 - SetPointFieldEditor 213
 - SetPointSlider 207
 - SetPointToggleButton 209
 - pxeditor 218
 - NewPxViewDialog 218
 - report 219, 223
 - BqlGrid 219
 - ComponentGrid 220
 - EmailRecipient 221
 - Export Source 221
 - File Recipient 222
 - ReportPane 223
 - SectionHeader 224
 - ReportService 223
 - SetPointCheckBox 211
 - Converters
 - types 57
 - createanhtml5graphic 90
 - createuseraccount 93
 - customizations 69
- D**
- Data binding 49
 - add 50
 - types 53
 - Distribution 104
- E**
- EdgePane 158
 - email
 - EmailRecipient 83
 - EmailRecipient (report) 83
 - Export Source 221
- F**
- field editor bindings 60
 - File Recipient
 - component 222
 - FlowPane 159
- G**
- Grid Editor view 131
 - assign components 133
 - Edit Column 134
 - Edit Row 134
 - workflow 133
 - Grid Label Pane view 129
 - Grid Table view 128
 - GridPane 162
 - gx
 - components 176
- H**
- History Chart Builder
 - launch using a Popup binding 68
 - Hx profiling visible property
 - u 72
- I**
- Increment setpoint binding 58
- K**
- Kiosk Profiles
 - Default Kiosk 98
 - Handheld 98
 - kitpx 201
 - kitPx
 - components 176
 - Kitpx 199
 - kitPX palette 113
 - kitPx widgets 17
 - kitPxGraphics 114
 - kitPxGraphics widgets 17
 - kitPxHvac palette 113
 - kitPxHvac widgets 17
 - kitPxN4svg 115
 - kitPxN4svg widgets 17
- L**
- localizable label 201
 - Localizable Button 199
 - login screen
 - create custom 69
- M**
- make widget wizard 41
 - Action widget 44
 - bound label widget 25
 - chart widget 29
 - component properties-based 33
 - palette kit-based 35
 - Time Plot widget 39
 - Make Widget wizard 23
 - menu actions 25
 - Mobile Px app 135

N

| | |
|-------------------------------|-----|
| Nav file | |
| add nodes..... | 76 |
| building nav files | 75 |
| create | 75 |
| delete nodes..... | 76 |
| delete using popup menu | 75 |
| edit node hierarchy | 77 |
| edit nodes..... | 77 |
| elements | |
| <nav>..... | 125 |
| <node> | 125 |
| open a nav file..... | 76 |
| rename | 75 |
| Nav File Editor | |
| buttons..... | 127 |
| controls | 127 |
| result tree | 126 |
| source objects..... | 126 |
| Nav file, editing..... | 75 |
| new in N4 graphics | 12 |

O

| | |
|------------------------------------|-----|
| optimize Px files for mobile | 70 |
| ord | |
| relativizing | 52 |
| ORD | |
| absolute vs. relative..... | 116 |

P

| | |
|----------------------------|-------|
| Palette | |
| bajoui..... | 120 |
| kitPxGraphics..... | 114 |
| kitPxN4svg | 115 |
| Px Layers Palette..... | 108 |
| Px Property..... | 106 |
| Plugins | |
| ComponentGrid..... | 129 |
| Grid Editor view..... | 131 |
| plugins | 137 |
| Plugins | |
| pxEditor | 225 |
| Plugins module | |
| report | 225 |
| Popup Bindings | 60 |
| Portability | |
| examples | 116 |
| Presentation Xml (Px)..... | 11 |
| Profiles | |
| about..... | 85 |
| Kiosk | 97–98 |
| Web | |
| about | 85 |
| Basic Hx | 86 |
| Basic Wb | 96 |

| | |
|---------------------------|--------|
| Default Hx | 87 |
| Default Mobile..... | 94 |
| Default Wb | 96 |
| Handheld Hx..... | 88 |
| Handheld Wb | 97 |
| HTML5 Hx Profile..... | 88 |
| Hx | 72 |
| Simple Admin Wb | 97 |
| Velocity Doc | 86 |
| property parameters | |
| linking | |
| unlinking..... | 105 |
| Px Editor | |
| Canvas..... | 103 |
| zoom | 15–16 |
| Px file..... | 12, 99 |
| add comment box | 65–67 |
| embed history chart | 67 |
| Px files | 16 |
| add comments box..... | 65 |
| Px files, shared | 17 |
| Px graphics..... | 99 |
| about..... | 11 |
| Px Layer | |
| adding | 21 |
| assign objects | 21 |
| remove layer | 21 |
| rename | 21 |
| unassign objects..... | 21 |
| Px layers | |
| about..... | 20 |
| Px Layers..... | 20 |
| about..... | 107 |
| Px Layers Palette..... | 108 |
| Px page | |
| zoom | 15–16 |
| Px properties..... | 104 |
| Px Properties | |
| concepts | |
| property parameters..... | 105 |
| Px Property palette..... | 106 |
| Px target media | |
| HxPx Media | 13 |
| Mobile PxMedia | 13 |
| Report Px Media | 13 |
| types | 13 |
| Ux PxMedia | 13 |
| Workbench PxMedia | 13 |
| Px View | |
| create on component | 19 |
| Px Viewer | 14 |
| Px views | 12, 99 |
| Px Views | |
| visual styles..... | 69 |
| pxeditor | |
| components..... | 218 |
| PxEditor | |
| about | |
| canvas | |

| | |
|---------------------------------------|-----|
| side bar pane..... | 102 |
| zooming..... | 15 |
| pxEditor palette | 218 |
| PxInclude | 77 |
| embed Px file with ord variables..... | 78 |
| PxInclude widget | |
| multiple ORD variables..... | 48 |
| single ORD variable..... | 47 |
| workflow..... | 46 |
| PxInclude Widget | 120 |
| concepts..... | 121 |
| ORD variables..... | 124 |
| properties..... | 122 |
| using..... | 123 |
| PxMedia type | |
| choosing..... | 100 |

R

| | |
|-----------------------------------|---------|
| reference | 99 |
| relativizing ORDs..... | 61 |
| report | |
| components..... | 219 |
| Reporting | |
| EmailRecipient component | 83 |
| ExportSource..... | 83 |
| Reporting process | |
| workflow..... | 81 |
| ReportingService | |
| setup the | 81 |
| ReportPane | |
| component | 223 |
| ReportPxFile | 82, 127 |
| Reports | |
| generating..... | 81 |
| layout data in ComponentGrid..... | 82 |
| ReportService..... | 223 |
| ResponsiveMigration | 218 |
| ResponsivePane | 160 |

S

| | |
|----------------------------------|-------------|
| SectionHeader | |
| component | 224 |
| Setpoint binding | |
| about..... | 58 |
| Slot properties | |
| Px views as..... | 20 |
| Spectrum bindings | |
| about..... | 57 |
| sSpectrum setpoint bindings..... | 59 |
| SVG | |
| rendering..... | 109 |
| SVG support..... | 32, 51, 108 |
| browser compatible SVGs..... | 109 |

T

| | |
|--------------------------------|-----|
| table bindings | 60 |
| tag-based NEQL bindings | |
| about..... | 62 |
| converting bound ORDs to | 71 |
| target media | 13 |
| Third-party graphics..... | 116 |

V

| | |
|--------------------------|----------|
| Value bindings | |
| about..... | 56 |
| View Source XML..... | 109 |
| views..... | 137 |
| Views | |
| ComponentGridEditor..... | 225 |
| Grid Editor view | 129, 131 |
| Grid Table view | 128 |
| GridLabelPane | 225 |
| GridTable..... | 225 |
| pxEditor | 225 |
| PxEditor..... | 225 |
| report | 225 |
| report module..... | 225 |
| reportGridLabelPane..... | 225 |

W

| | |
|---|-----|
| widget | |
| Action..... | 44 |
| adding using Make New Widget wizard | 23 |
| chart..... | 29 |
| commands..... | 25 |
| component properties-based..... | 33 |
| make bound label widget | 25 |
| make view-based widgets..... | 41 |
| painting | 24 |
| palette kit-based module..... | 35 |
| PxInclude..... | 46 |
| sizing..... | 46 |
| Time Plot | 39 |
| Widget | |
| add binding | 50 |
| animate property | 50 |
| layout | 110 |
| panes..... | 111 |
| Picture | 32 |
| properties..... | 112 |
| relationships | 110 |
| scalable..... | 32 |
| Widget widget | |
| add a weather report..... | 68 |
| widgets | |
| about..... | 17 |
| Widgets | |
| Picture..... | 51 |
| PxInclude..... | 120 |

Widgets, Picture 108
windows..... 137

Z

Zoom

about..... 15
capability 15
reset zoom..... 16
zoom-in 15
zoom-out 16