1. Vision Components . . . 3
1.1 Vision - Input Palette . . . 4
1.1.1 Vision - Text Field . . . 5
1.1.2 Vision - Numeric Text Field . . . 16
1.1.3 Vision - Spinner . . . 27
1.1.4 Vision - Formatted Text Field . . . 30
1.1.5 Vision - Password Field . . . 42
1.1.6 Vision - Text Area . . . 53
1.1.7 Vision - Dropdown List . . . 64
1.1.8 Vision - Slider . . . 76
1.1.9 Vision - Language Selector . . . 88
1.2 Vision - Buttons Palette . . . 94
1.2.1 Vision - Button . . . 95
1.2.2 Vision - 2 State Toggle . . . 106
1.2.3 Vision - Multi-State Button . . . 117
1.2.4 Vision - One-Shot Button . . . 129
1.2.5 Vision - Momentary Button . . . 140
1.2.6 Vision - Toggle Button . . . 151
1.2.7 Vision - Check Box . . . 161
1.2.8 Vision - Radio Button . . . 171
1.2.9 Vision - Tab Strip . . . 181
1.3 Vision - Display Palette . . . 190
1.3.1 Vision - Label . . . 191
1.3.2 Vision - Numeric Label . . . 198
1.3.3 Vision - Multi-State Indicator . . . 205
1.3.4 Vision - LED Display . . . 213
1.3.5 Vision - Moving Analog Indicator . . . 222
1.3.6 Vision - Image . . . 230
1.3.7 Vision - Progress Bar . . . 237
1.3.8 Vision - Cylindrical Tank . . . 245
1.3.9 Vision - Level Indicator . . . 252
1.3.10 Vision - Linear Scale . . . 260
1.3.11 Vision - Barcode . . . 268
1.3.12 Vision - Meter . . . 275
1.3.13 Vision - Compass . . . 285
1.3.14 Vision - Thermometer . . . 292
1.3.15 Vision - IP Camera Viewer . . . 300
1.4 Vision - Tables Palette . . . 308
1.4.1 Vision - Table . . . 309
1.4.1.1 Vision - Table Customizer . . . 326
1.4.2 Vision - Power Table . . . 332
1.4.2.1 Vision - Power Table Customizer . . . 347
1.4.3 Vision - List . . . 351
1.4.4 Vision - Tree View . . . 363
1.4.4.1 Vision - Tree View Customizer . . . 372
1.4.5 Vision - Comments Panel . . . 375
1.4.6 Vision - Tag Browse Tree . . . 389
1.5 Vision - Charts Palette . . . 396
1.5.1 Vision - Easy Chart . . . 397
1.5.1.1 Vision - Easy Chart Customizer . . . 413
1.5.2 Vision - Chart . . . 421
1.5.2.1 Vision - Chart Customizer . . . 429
1.5.3 Vision - Sparkline Chart . . . 452
1.5.4 Vision - Bar Chart . . . 459
1.5.5 Vision - Radar Chart . . . 468
1.5.6 Vision - Status Chart . . . 476
1.5.7 Vision - Pie Chart . . . 489
1.5.8 Vision - Box and Whisker Chart . . . 498
1.5.9 Vision - Equipment Schedule . . . 509
1.5.10 Vision - Gantt Chart . . . 523
1.6 Vision - Calendar Palette . . . 531
1.6.1 Vision - Calendar . . . 532
1.6.2 Vision - Popup Calendar . . . 539
1.6.3 Vision - Date Range . . . 546
1.6.4 Vision - Day View . . . 556
1.6.5 Vision - Week View . . . 563
1.6.6 Vision - Month View . . . 571
1.7 Vision - Admin Palette . . . 579
1.7.1 Vision - User Management . . . 580
1.7.2 Vision - Schedule Management . . . 590
1.7.3 Vision - Roster Management . . . 600
1.7.4 Vision - SFC Monitor . . . 608
1.8 Vision - Alarming Palette . . . 615
1.8.1 Vision - Alarm Status Table . . . 616
1.8.1.1 Vision - Alarm Row Style Customizer . . . 624

1.8.2 Vision - Alarm Journal Table 629
1.9 Vision - Containers Palette 642
1.9.1 Vision - Container 643
1.9.2 Vision - Template Repeater 650
1.9.3 Vision - Template Canvas 653
1.10 Vision - Misc Palette 660
1.10.1 Vision - Paintable Canvas 661
1.10.2 Vision - Line 670
1.10.3 Vision - Pipe Segment 678
1.10.4 Vision - Pipe Joint 685
1.10.5 Vision - Sound Player 692
1.10.6 Vision - Timer 698
1.10.7 Vision - Signal Generator 700
1.11 Vision - Reporting Palette 702
1.11.1 Vision - Report Viewer 703
1.11.2 Vision - Row Selector 711
1.11.3 Vision - Column Selector 719
1.11.4 Vision - File Explorer 727
1.11.5 Vision - PDF Viewer 734
1.12 Vision - Web Browser Palette 744
1.12.1 Vision - Web Browser Component 745
1.13 Vision - The Window Object 751

Vision Components

This section covers all the built-in Vision components. While the component is selected, you can use the Property Editor panel to alter the component's properties, which changes the component's appearance and behavior. Shapes are Vision components too. Each shape may be individually selected, named, and has its own properties. Shapes have some additional capabilities that other Vision components don't have, such as the ability to be rotated. Shapes are created using the shape tools, not dragged from the component palette.

To make any of these components do something useful, like display dynamic information or control a device register, you configure property bindings for the component. To make the component react to user interaction, you configure event handlers for it.

Vision - Input Palette

Vision - Buttons Palette

Vision - Display Palette

Vision - Tables Palette

Vision - Charts Palette

Vision - Calendar Palette

Vision - Admin Palette

Alarming

Vision - Containers Palette

Vision - Misc Palette

Vision - Reporting Palette

Vision - Web Browser Palette

Vision - The Window Object

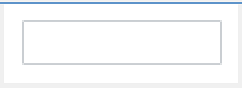
Vision - Input Palette

Input Components

The following components allow users to enter or select data.

In This Section ...

Vision - Text Field

General

Component Palette Icon:

Description
The Text Field component is used for input of any single-line text. This component will accept any alpha-numeric input. If you're looking for a numeric field, see the Vision - Numeric Text Field. This field features a protected mode. When you enable the protectedMode property, the field is not editable even when it receives input focus. The user must double click on the field or press enter in order to edit the field. When they are done (press enter again or leave the field), the field becomes non-editable again. The Text Field also supports the reject updates during edit feature. This feature ignores updates coming from property bindings while the component is being edited by a user.

Properties

Name	Description	Property Type	Scripting	Category
Background	The background color of the text box (when editable).	Color	.editableBackground	Appearance
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border ⚠ The border is unaffected by rotation.	Border	.border	Common
Commit On Focus Loss	If true, any pending edit will take effect when focus is lost. If false, the user must press ENTER for an edit to take effect.	boolean	.commitOnFocusLost	Behavior
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Defer Updates	When true, the 'text' property will not fire updates while typing, it will wait for Enter to be pressed.	boolean	.deferUpdates	Behavior
Editable?	If true, this is an input box, if false, this is display-only.	boolean	.editable	Behavior
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance

Horizontal Alignment	Determines the alignment of the label's contents along the X axis.	int	.horizontalAlignment	Layout
Maximum Characters	The text box will be limited to this number of characters. Use -1 for unlimited.	int	.maxChars	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up when the user mouses over of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Non-Editable Background	The background color to use when this text box is non-editable.	Color	.nonEditableBackground	Appearance
Protected Mode?	If true, users will need to double-click in the field in order to edit the text.	boolean	.protectedMode	Behavior
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Reject Updates During Edit	If true, this field will not accept updates from external sources (like DB bindings) while the user is editing the field.	boolean	.rejectUpdatesDuringEdit	Behavior
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Text	Text of this component.	String	.text	Data
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

- Description

  Returns the currently selected or highlighted text in the text field.

- Parameters

  Nothing

- Return

  String - Returns the currently selected or highlighted text in the text field.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event.
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when a component that had the input focus lost it to another component.

.source	The component that fired this event
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

An integer that indicates whether the state was changed to "Selected" (on) or "Deselected" (off). Compare this to the event object's constants to determine what the new state is.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is released and the source component has the input focus. Works for all characters, including non-printable ones, such as SHIFT and F3.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is pressed and then released when source component has the input focus. Only works for characters that can be printed on the screen.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires *aft er* the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed.
.propertyName	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

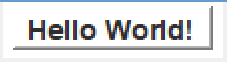
- Component Customizers
- Style Customizer

Examples

Code Snippet

```
#The following code will return the value of the text box's previous value into a variable.
#This code is fired on the propertyChange event for this component.

oldValue = event.source.oldValue
```

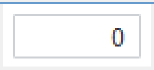
Titled Panel



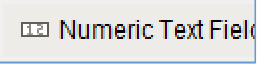
Property Name	Value
Border	Bevel (Raised)
Font	Dialog, Bold, 14
Horizontal Alignment	Center

Vision - Numeric Text Field

General



Component Palette Icon:



Description

The Numeric Text Field is similar to the standard Text Field, except that it is specialized for use with numbers. Instead of a Text property, it has four numeric "value" properties: integer, double, long, and float. Which one you use depends on the mode of the text box.

Like the standard Text Field, this text field can operate in protected mode. When you enable the protected property, the field is not editable even when it receives input focus. The user must double click on the field or press enter in order to edit the field. When they are done (press enter again or leave the field), the field becomes non-editable again.

The Numeric Text Field also supports the reject updates during edit feature. This feature ignores updates coming from property bindings while the component is being edited by a user.

Properties

Name	Description	Property Type	Scripting	Category
Background	The background color of the text box (when editable).	Color	.editableBackground	Appearance
Border	The border surrounding this component. No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Commit On Focus Loss	If true, any pending edit will take effect when focus is lost. If false, the user must press Enter for an edit to take effect.	boolean	.commitOnFocusLost	Behavior
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Decimal Format	The formatting string used for displaying numbers.	String	.decimalFormat	Appearance
Defer Updates	When true, the value properties will not fire updates while typing, it will wait for Enter to be pressed.	boolean	.deferUpdates	Behavior
Editable?	If true, this is an input box, if false, this is display-only.	boolean	.editable	Behavior
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common

Error on Out-of-Bounds	Show an error message if the user input is out-of-bounds?	boolean	.errorOnOutOfBounds	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance
Horizontal Alignment	Determines the alignment of the label's contents along the X axis.	int	.horizontalAlignment	Layout
Maximum	The maximum value (inclusive), if useBounds is true.	double	.maximum	Data
Minimum	The minimum value (inclusive), if useBounds is true.	double	.minimum	Data
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Non-Editable Background	The background color to use when this text box is non-editable	Color	.nonEditableBackground	Appearance
Number Type	What type of numbers should this field accept?	int	.mode	Data
Out Of Bounds Message	The error message to display if input is out-of-bounds.	String	.outOfBoundsMessage	Behavior
Protected Mode?	If true, users will need to double-click in the field in order to edit the value.	boolean	.protectedMode	Behavior
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Reject Updates During Edit	If true, this field will not accept updates from external sources (like DB bindings) while the user is editing the field.	boolean	.rejectUpdatesDuringEdit	Behavior
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Suffix	A string to display after the value.	String	.suffix	Appearance
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
Use Bounds?	Only allows user-entered values between a minimum and maximum. Unless you turn on "Error on out-of-bounds", user-entered values will be silently modified to be in-bounds.	boolean	.useBounds	Behavior
Value (Double)	The value as a double. Make sure you use the value property that corresponds to your Number Type setting.	double	.doubleValue	Data
Value (Float)	The value as a float. Make sure you use the value property that corresponds to your Number Type setting.	float	.floatValue	Data
Value (Integer)	The value as an integer. Make sure you use the value property that corresponds to your Number Type setting.	int	.intValue	Data
Value (Long)	The value as a long. Make sure you use the value property that corresponds to your Number Type setting.	long	.longValue	Data

Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

- Description

  Returns the currently selected or highlighted text in the text field.

- Parameters

  Nothing

- Return

  String - Returns the currently selected or highlighted text in the text field.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event.
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when a component that had the input focus lost it to another component.

.source	The component that fired this event
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

An integer that indicates whether the state was changed to "Selected" (on) or "Deselected" (off). Compare this to the event object's constants to determine what the new state is.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is released and the source component has the input focus. Works for all characters, including non-printable ones, such as SHIFT and F3.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is pressed and then released when source component has the input focus. Only works for characters that can be printed on the screen.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires *aft er* the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

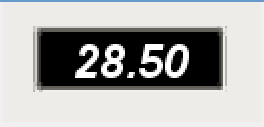
- Component Customizers
- Style Customizer

Examples

Code Snippet

```
#The following script can be executed on a mouse released event handler.
#This would write the selected text to a custom property called highlightedText.

event.source.highlightedText = event.source.getSelectedText()
```

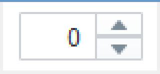
2-digit Numeric Format



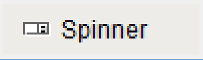
Property Name	Value
Border	Field Border
Number Type	Float
Font	Dialog, BoldItalic, 15
Decimal Format	#,##0.00

Vision - Spinner

General



Component Palette Icon:



Description

The spinner component represents a value that is part of a series of values, such as numbers and dates. It allows you to not only edit the value directly, but to 'spin' the value up or down, using the up and down buttons that are part of the component. When setting up property bindings, make sure you use the value property that corresponds to the spinner mode. For example, if you chose the Double spinner mode, you should bind the doubleValue property.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Date Format	A date format pattern to use when the spinner is in date mode.	String	.dateFormat	Appearance
Date in Milliseconds	The date in milliseconds from epoch time. (Read only. Usable in bindings and scripting.)	long	.dateInMillis	Uncategorized
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.foreground	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Number Format	A number format pattern to use when the spinner is in numeric mode.	String	.numberFormat	Appearance

Numeric Maximum	The maximum value this spinner will accept when in 'Integer' or 'Double' mode.	double	.maxValue	Data
Numeric Minimum	The minimum value this spinner will accept when in 'Integer' or 'Double' mode.	double	.minValue	Data
Numeric Step Size	The size to step up or down when in 'Integer' or 'Double' mode.	double	.stepSize	Behavior
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Spinner Mode	The mode controls which data type this spinner accepts.	int	.spinnerMode	Behavior
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
Value (Date)	The current value if mode is 'Date'.	Date	.dateValue	Data
Value (Double)	The current value if mode is 'Double'.	double	.doubleValue	Data
Value (Integer)	The current value if mode is 'Integer'.	int	.intValue	Data
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

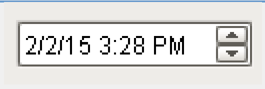
.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Component Customizers
- Style Customizer

Examples

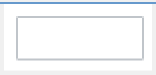
Date Spinner



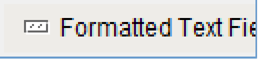
Property Name	Value
Spinner Mode	Date

Vision - Formatted Text Field

General



Component Palette Icon:



Description

This specialized text field is used for alphanumeric text input that must match some specific pattern or needs to be formatted in a specific way. It operates in two modes:

Formatted Mask
In this mode, input is automatically formatted and restricted based on a format mask. For example, a format mask like: (###) ###-#### will allow the entry of a 10-digit US phone number. The formatting characters are automatically inserted if the user does not type them in. Any other characters are restricted. The following characters may be used in a formatted mask pattern:

Symbol	Description
#	Any valid number, Such as 0-9.
'	Escape character, used to escape any of the special formatting characters.
U	Any letter. All lowercase letters will be mapped to upper case automatically.
L	Any letter. All upper case letters will be mapped to lower case automatically.
A	Any letter or number.
?	Any letter, case is preserved.
*	Anything.
H	Any hex character (0-9, a-f or A-F).

Regular Expression
In this mode, input is validated against a regular expression. A regular expression is a special string that defines a set of allowed strings. Any input that matches the given regular expression is allowed, and input that doesn't match, is restricted. And yes, while powerful, regular expressions are decidedly difficult to decipher.

Properties

Name	Description	Property Type	Scripting	Category
Allows Invalid Text	Allows Invalid text to Commit.	boolean	.allowsInvalid	Behavior
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance

Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Commit While Typing	Commits valid text while user is typing.	boolean	.commitsOnValidEdit	Behavior
Committed Value	Committed text value.	String	.committedValue	Data
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Focus Lost Behavior	Controls how a transaction can be committed.	int	.focusLostBehavior	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.foreground	Appearance
Formatted Mask Pattern	Formatted Mask Validation Pattern.	String	.formattedMaskPattern	Behavior
Horizontal Alignment	Determines the alignment of the label's contents along the X axis.	int	.horizontalAlignment	Layout
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Overwrites Text	Overwrites text while typing.	boolean	.overwriteMode	Behavior
Reg Ex Pattern	Regular Expression Validation Pattern.	String	.validationPattern	Behavior
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Text	Contents of this Text Field.	String	.text	Data
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
Validation Mode	Select regular expression or mask-driven field validation.	int	.validationMode	Behavior

Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event.
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when a component that had the input focus lost it to another component.

.source	The component that fired this event
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

An integer that indicates whether the state was changed to "Selected" (on) or "Deselected" (off). Compare this to the event object's constants to determine what the new state is.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is released and the source component has the input focus. Works for all characters, including non-printable ones, such as SHIFT and F3.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is pressed and then released when source component has the input focus. Only works for characters that can be printed on the screen.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires *aft er* the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed.
.propertyName	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers
- Style Customizer

Examples

Formatted Mask

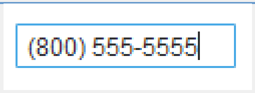
Example	Description
##U-####/UU	A product code with a specifc format, like 28E-8213/AR
0xHHHH	A hex digit, automatically prepends "0x" on the front. e.g. "0x82FF"
#UUU###	A California license plate, eg. 4ABC123

Regular Expression

Example	Description
\p{Upper}\p{Lower}*, \p{Upper}\p{Lower}*	A name, formatted such as Smith, John
\d{3}-\d{2}-\d{4}	A US social security number, like 123-45-6789
\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,3}	A network IPv4 address, like 67.82.120.116
^[a-f0-9A-F]{6}$	A six-digit hexadecimal number

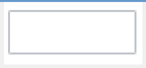
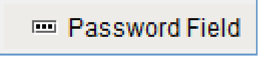
Gallery

Phone Number Format


(800) 555-5555

Property Name	Value
Validation Mode	Formatted Mask
Formatted Mask Pattern	(###) ###-####

Vision - Password Field

General

Component Palette Icon:
 Password Field

Description
A password field is like a text field that doesn't display the text that is being edited. You may alter the echo character (*) if you'd like.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Echo Character	The character that is displayed instead of the real ones.	String	.echoCharacter	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Text	Text of this component	String	.text	Data
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event.
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when a component that had the input focus lost it to another component.

.source	The component that fired this event
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

An integer that indicates whether the state was changed to "Selected" (on) or "Deselected" (off). Compare this to the event object's constants to determine what the new state is.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is released and the source component has the input focus. Works for all characters, including non-printable ones, such as SHIFT and F3.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is pressed and then released when source component has the input focus. Only works for characters that can be printed on the screen.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires *after* the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

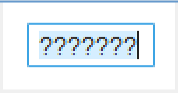
.source	The component that fired this event.
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed.
. property Name	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers
- Style Customizer

Examples

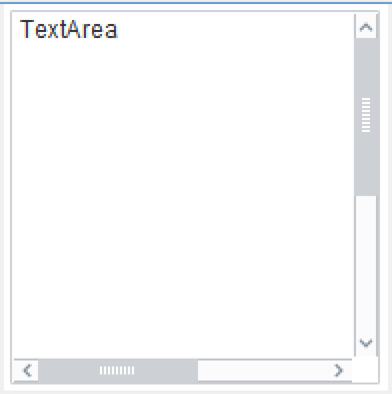
Password Field with Question Marks as the Echo Character


???????

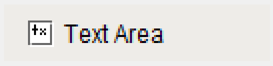
Property Name	Value
Echo Character	?

Vision - Text Area

General



Component Palette Icon:



Description

Suitable for multi-line text display and editing. Will scroll vertically on demand. Will scroll horizontally if line wrap is off. Only supports plain-text, no HTML formatting or styled text.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Columns	The number of columns you expect to display (used as a hint for scrollbars).	int	.columns	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Defer Updates	When true, the 'text' property will not fire updates while typing. It will wait for the component to lose focus.	boolean	.deferUpdates	Behavior
Editable	Controls whether or not the user can edit the text within this text area. When the option is not selected, the text is not editable in the client and the background of the component will be grey.	boolean	.editable	Behavior
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common

Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance
Line Wrap	Should this area wrap lines?	boolean	.lineWrap	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Reject Updates During Edit	If true, this field will not accept updates from external sources (like DB bindings) while the user is editing the field.	boolean	.rejectUpdatesDuringEdit	Behavior
Rows	The number of rows you expect to display (used as a hint for scrollbars).	int	.rows	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Tab Size	This adjusts the default size of tab characters.	int	.tabSize	Appearance
Text	Text of this component.	String	.text	Data
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event.
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when a component that had the input focus lost it to another component.

.source	The component that fired this event
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

An integer that indicates whether the state was changed to "Selected" (on) or "Deselected" (off). Compare this to the event object's constants to determine what the new state is.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is released and the source component has the input focus. Works for all characters, including non-printable ones, such as SHIFT and F3.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is pressed and then released when source component has the input focus. Only works for characters that can be printed on the screen.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires *aft er* the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event.
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed.
. property Name	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

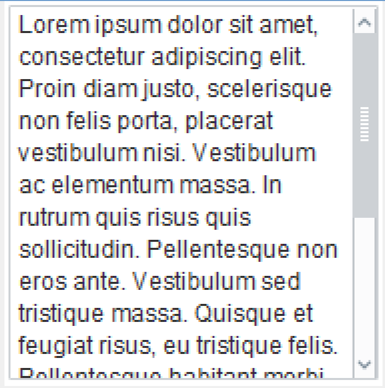
Customizers

- Vision Component Customizers
- Style Customizer

Examples

Gallery

Word Wrap and no Scroll Bars until they are needed

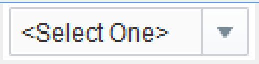


Lorem ipsum dolor sit amet, consectetur adipiscing elit. Proin diam justo, scelerisque non felis porta, placerat vestibulum nisi. Vestibulum ac elementum massa. In rutrum quis risus quis sollicitudin. Pellentesque non eros ante. Vestibulum sed tristique massa. Quisque et feugiat risus, eu tristique felis.

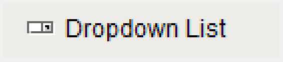
Property Name	Value
Line Wrap	True
Text	468 Characters
Rows	0
Columns	0

Vision - Dropdown List

General



Component Palette Icon:



INDUCTIVE UNIVERSITY

Dropdown

Watch the Video

Description

The Dropdown component is a great way to display a list of choices in a limited amount of space. The current selection is shown, and the choices are only presented when the user clicks on the dropdown button. The choices that are shown depend on the data property. This is a dataset, which can be typed in manually in the Designer, or (more commonly) it can be populated dynamically from a property binding, often a SQL Query binding.

It is often the case that you want to display choices to the user that are 'dressed up' versions of the actual choices. For instance, suppose that you are selecting choices for a downtime tracking entry. The choices might be: "Operator Error", "Machine Malfunction", and "Other". But, you really want to map these choices to some numeric code which is how the choice is stored. So, for instance, when the user chooses "Other" you really want to get the number 3. The dropdown component is perfect for such a use. The data property can be set up in one of three fashions, which control how the "selected values" properties change.

The three ways to set up the data dataset and the corresponding behavior is as follows:

Scenario 1: One column with a set of string values

Column1
Apples
Oranges
Bananas

- Drop down displays values from the first column
- **Selected value** is undefined
- **Selected String Value** represents value from first column
- **Selected Label** represents value from first column

Scenario 2: Two column with an integer and string column

Column1	Column2
201	Apples
202	Oranges
203	Bananas

- Dropdown displays values from the second column
- **Selected Value** represents a value from the first column
- **Selected String Value** represents value from second column
- **Selected Label** represents value from second column

Scenario 3: Two column with two string columns

Column1	Column2
APL	Apples
ORN	Oranges
BAN	Bananas

- Dropdown displays values from second column
- **Selected Value** is undefined
- **Selected String Value** represents value from first column
- **Selected Label** represents value from second column

The dropdown component can operate in one of three Selection Modes. These modes affect how the dropdown's current selection (defined by the values of its Selected Value, Selected String Value, and Selected Label properties) behave when the selection properties are set to values not present in the choice list, or conversely, when the choice list is set to a new dataset that doesn't contain the current selection:

• **Strict**. Selected values must always correlate to an option in the list defined by the Data property. If an invalid selection is set (via a binding or a script), the selection will be set to the values defined by the No Selection properties. If the Data property is set to a list that does not contain the current selection, the current selection will be reset to the No Selection values.

• **Lenient**. (default) Selected values are independent of the list defined by the Data property. This mode is useful to avoid race conditions that can cause problems in Strict mode when both the Data and the Selected Value properties are bound. If the current selection is not present in the Data list, the read-only property Selected Index will be -1.

• **Editable**. The same selection rules as defined by Lenient mode, except that the dropdown itself becomes editable, allowing a user to type in their own arbitrary value. This value will be set as the dropdown's Selected Label.

Properties

Name	Description	Property Type	S
Background Color	The background color of the component.	Color	. b r
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	. c c
Data	The data which fills up the combo box. Either a one-column or two-column DataSet, with the first column being the value, and the second being the display	Dataset	.
Dropdown Display Mode	Changes the dropdown's display.	int	.
Enabled	If disabled, a component cannot be used.	boolean	. c r
Font	Font of text on this component.	Font	.
Foreground Color	The foreground color of the component.	Color	. f
Hide Table Columns?	A comma separated list of columns to hide from the dropdown table, for example, "0,2" (only used in table mode).	String	. h e

Horizontal Alignment	Determines the alignment of the contents along the X axis.	int	. h A
Max Row Count	The number of rows to display in the dropdown list before displaying a scrollbar.	int	. r F r
Max Table Height	The maximum height allowed for the dropdown table (only used in table mode).	int	. r e
Max Table Width	The maximum width allowed for the dropdown table (only used in table mode).	int	. r e
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	. t x
Name	The name of this component.	String	.
No Selection Label	The label to display when nothing is selected.	String	. r c
No Selection String	The string value when nothing is selected.	String	. r c
No Selection Value	The value when nothing is selected.	int	. r c
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.
Row Height	The following feature is new in Ignition version **8.0.16** Click here to check out the other new features Determines the height of each item in the dropdown list. The default is 16 pixels.	int	. r
Selected Index	The index of the selected item. (Read only. Usable in bindings and scripting.)	int	. s r
Selected Label	The currently selected label.	String	. s a
Selected String Value	The currently selected value, if the value column is a string.	String	. s S u
Selected Value	The currently selected value.	Integer	. s V
Selection Background	The background color of a selected cell in the dropdown list.	Color	. s B r
Selection Mode	The selection mode determines the behavior of the dropdown: whether its selected value must strictly be in the underlying set of choices, whether it is flexible, or if users can type into the component.	int	. s M

Show Table Header?	Selects whether or not the dropdown table header is displayed (only used in table mode).	boolean	. s e
Styles	Contains the component's styles.	Dataset	.
Vertical Alignment	Determines the alignment of the contents along the Y axis.	int	. v g
Visible	If disabled, the component will be hidden.	boolean	.
Deprecated Properties			
Data Quality	The data quality code for any Tag bindings on this component.	int	. c t

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event.
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when a component that had the input focus lost it to another component.

.source	The component that fired this event
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

An integer that indicates whether the state was changed to "Selected" (on) or "Deselected" (off). Compare this to the event object's constants to determine what the new state is.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is released and the source component has the input focus. Works for all characters, including non-printable ones, such as SHIFT and F3.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is pressed and then released when source component has the input focus. Only works for characters that can be printed on the screen.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires *aft er* the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed.
.propertyName	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

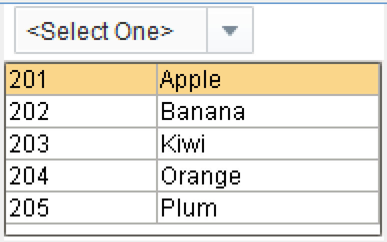
- Vision Component Customizers
- Style Customizer

Examples

Code Snippet

```
#The following code will return the first column value of the selection.
#This code would be on a button in the same container as the dropdown.

selRow = event.source.parent.getComponent('Dropdown').selectedIndex
pyData = system.dataset.toPyDataSet(event.source.parent.getComponent('Dropdown').data)
code = pyData[selRow][0]
print code
```

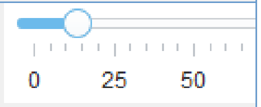
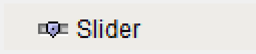
Display Multiple Columns in Dropdown



201	Apple
202	Banana
203	Kiwi
204	Orange
205	Plum

Property Name	Value
Dropdown Display Mode	Table
Show Table Header	False

Vision - Slider

General

Component Palette Icon:


Description
The slider component lets the user drag an indicator along a scale to choose a value in a range. The slider can be configured to orient horizontally or vertically.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Defer Updates	Only publish updates to value when not actively being changed.	boolean	.deferred	Behavior
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.foreground	Appearance
Horizontal Slider	If true, slider is horizontal, otherwise, it's vertical.	boolean	.horizontal	Appearance
Inverted?	Specify true to reverse the value range shown for the slider and false to put the value range in the normal order.	boolean	.inverted	Behavior
Major Tick Spacing	The distance, measured in values, between each major tick mark.	int	.majorTickSpacing	Appearance

Maximum Value	The value when the slider is all the way right or up.	int	.maximum	Data
Minimum Value	The value when the slider is all the way left or down.	int	.minimum	Data
Minor Tick Spacing	The distance, measured in values, between each minor tick mark.	int	.minorTickSpacing	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Paint Labels?	If true, value labels will be shown.	boolean	.paintLabels	Appearance
Paint Ticks?	If true, value tick marks will be shown.	boolean	.paintTicks	Appearance
Paint Track?	If true, the track of the slider will be shown.	boolean	.paintTrack	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Snap To Ticks?	Only allows selection of values at the tick marks.	boolean	.snapToTicks	Behavior
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Value	The current value of the slider.	int	.value	Data
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event.
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when a component that had the input focus lost it to another component.

.source	The component that fired this event
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

An integer that indicates whether the state was changed to "Selected" (on) or "Deselected" (off). Compare this to the event object's constants to determine what the new state is.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation. The keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is released and the source component has the input focus. Works for all characters, including non-printable ones, such as SHIFT and F3.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation. The keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is pressed and then released when source component has the input focus. Only works for characters that can be printed on the screen.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation. The keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires *aft er* the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

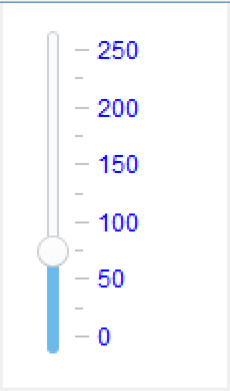
- Vision Component Customizers
- Style Customizer

Examples

Code Snippet

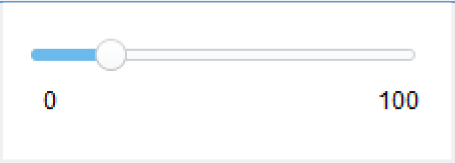
```
#The following code will return the value of the slider's previous value into a variable.
#This code is fired on the property change scripting for this component.

oldValue = event.source.oldValue
```

Vertical Slider with Border and Blue Text



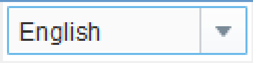
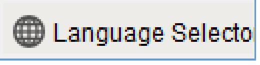
Property Name	Value
Maximum Value	250
Minor Tick Spacing	25
Foreground Color	0,0,255
Major Tick Spacing	50

Horizontal Slider without Tickmarks



Property Name	Value
Paint Ticks?	False
Minor Tick Spacing	0
Major Tick Spacing	100

Vision - Language Selector

General

Component Palette Icon:


Description
The Language Selector component gives an easy way to set the user's locale to control display of dates, times, numbers, and the language used for translations.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. The border is unaffected by rotation.	Border	.border	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Selected Locale	The display name of the currently selected locale. (Read only. Usable in bindings and scripting.)	String	.selectedLocale	Uncategorized
Selection Background	The background color of a selected cell in the dropdown list.	Color	.selectionBackground	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

Scripting Functions
This component does not have scripting functions associated with it.

Extension Functions
This component does not have extension functions associated with it.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

⚠ This event fires *after* the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

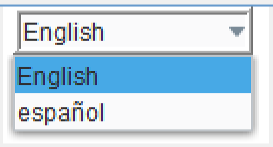
.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

This component does not have any customizers.

Examples

Select Between Languages



Property Name
No property changes made to this component for this example, but there must be at least one Spanish translation in the system.

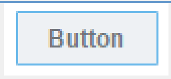
Vision - Buttons Palette

Button Components

The following components give you push-button options for displaying and writing values.

In This Section ...

Vision - Button

General

Component Palette Icon:

Description
The Button component is a versatile component, often used for things like opening/closing windows, writing to tags, and triggering any sort of scripting logic. It can be used for showing status, as well. For example, if you have three buttons, Hand, Off, and Auto, not only can they set those modes, but their background color can display the current mode, although you'd be better off using the Multi-State Button for this. To get buttons to do things, you add an event handler to the *actionPerformed* event. While you could configure your script on a mousePressed or mouseClicked event handlers, it is better to use the actionPerformed event. Why? Buttons can also be activated by tabbing over to them and hitting the space key, or they could be activated by pressing Alt and the button's mnemonic character. So, to make sure that your button works in all of these cases, configure your event handler on the actionPerformed event, not the mouseClicked event.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the button. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.buttonBG	Appearance
Border	The border surrounding this component. No Border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Border Painted?	Indicates if the border of this button should be displayed.	boolean	.borderPainted	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Default Button	If true, this button will be activated when the user presses Enter on the window.	boolean	.defaultBtn	Behavior
Disabled Image Path	The relative path of the image to be displayed when this component is not enabled.	String	.disabledPath	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Fill Area?	Controls whether or not this button's internal area is filled.	boolean	.contentAreaFilled	Appearance

Focusable	If a button is not focusable, you will not be able to interact with it with the keyboard. This means you can't "tab" over to it.	boolean	.focusable	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See Color Selector.	Color	.foreground	Appearance
Horizontal Alignment	The horizontal alignment of the button's contents (text and/or image).	int	.horizontalAlignment	Layout
Horizontal Text Position	The horizontal position of the button's text relative to its image.	int	.horizontalTextPosition	Layout
Icon-Text Spacing	The space (in pixels) between the icon (if any) and the text (if any).	int	.iconTextGap	Appearance
Image Path	The relative path of the image.	String	.path	Appearance
Margin	The space between a button's text and its borders.	Insets	.margin	Layout
Mnemonic	A single letter that will activate the button using 'ALT-*mnemonic*'.	String	.mnemonicChar	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Opaque	Is this button completely opaque? Most aren't, so this should usually be false.	boolean	.opaque	Common
Quality	The data quality code for any bindings on this component.	QualityCode	.quality	Data
Rollover	If true, the button may indicate that the mouse is hovering over it.	boolean	.rolloverEnabled	Behavior
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Text	Text of this component.	String	.text	Appearance
Vertical Alignment	The vertical alignment of the button's contents (text and/or image).	int	.verticalAlignment	Layout
Vertical Text Position	The vertical position of the button's text relative to its image.	int	.verticalTextPosition	Layout
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated				
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data

Scripting

Scripting Functions

- Description

  Virtually "clicks" the button, meaning that its actionPerformed event handler will run.

- Parameters

  Nothing

- Return

  Nothing

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event.
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when a component that had the input focus lost it to another component.

.source	The component that fired this event
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

An integer that indicates whether the state was changed to "Selected" (on) or "Deselected" (off). Compare this to the event object's constants to determine what the new state is.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is released and the source component has the input focus. Works for all characters, including non-printable ones, such as SHIFT and F3.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is pressed and then released when source component has the input focus. Only works for characters that can be printed on the screen.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires *aft er* the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

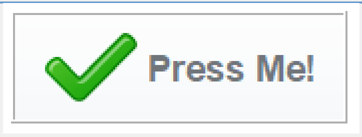
.source	The component that fired this event.
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed.
. property Name	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Component Customizers
- Style Customizer

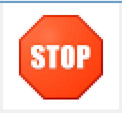
Examples

Styled Button



Property Name	Value
Border	Etched (Raised)
Font	Dialog, Bold, 18
Text	Press Me!
Image Path	Builtin/icons/48/check2.png

Styled Button



Property Name	Value
Border	No Border
Fill Area?	False
Border Painted?	False
Text	*None*
Image Path	Builtin/icons/48/stop.png

Vision - 2 State Toggle

General
Component Palette Icon:

Description
This button is similar to the basic Toggle Button, but more finely tuned to work in realistic controls environments. Use this button any time you want to toggle a value between two states, such as On/Off, Stop/Run, etc. If you have more than two states (for example, Hand/Off/Auto, use the Multi-State Button). If you have a tag whose value you want to toggle between 2 values (like zero and one), you can simply drag and drop the tag onto the button. This will bind both the Control Value and Indicator Value properties to that tag. Now set the State 1 Value and State 2 Value to your two states (they default to zero and one, respectively). Lastly, use the Styles Customizer to define the styles for your two states. This button has four integer values that you use to set it up: the Control Value, the Indicator Value, and values that define the 2 different states: State 1 Value and State 2 Value. Every time you press the button, one of the state values is written to the control value. The Indicator Value is used to determine which state you're in. For example, suppose that State 1 Value was zero and State 2 Value is one. If Indicator Value is zero and you press the button, it'll write a one to the Control Value. This means that if the Indicator value never changes, the button will continue to write the same value to the Control Value. The Style of the component is typically driven by the read-only property Current State. Current State equals zero when Indicator Value=State 1 Value and one otherwise.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the button. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.buttonBG	Appearance
Border	The border surrounding this component. No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Border Painted?	Indicates if the border of this button will be displayed.	boolean	.borderPainted	Appearance
Confirm Text	The message to ask the user if confirmation is turned on.	String	.confirmText	Behavior
Confirm?	If true, a confirmation box will be shown.	boolean	.confirm	Behavior
Control Value	Bind this to the tag that controls the state. (Typically, this is bound to the same location as *Indicator Value*).	int	.controlValue	Data
Current State	Read-only property that shows what state (0 or 1) this button is currently in.	int	.state	Data

Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Disabled Image Path	The relative path of the image to be displayed when this component is not enabled.	String	.disabledPath	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Fill Area?	Controls whether or not this button's internal area is filled.	boolean	.contentAreaFilled	Appearance
Focusable	If a button is not focusable, you will not be able to interact with it with the keyboard. This means you can't "tab" over to it.	boolean	.focusable	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See Color Selector.	Color	.foreground	Appearance
Horizontal Alignment	The horizontal alignment of the button's contents (text and/or image)	int	.horizontalAlignment	Layout
Horizontal Text Position	The horizontal position of the button's text relative to its image.	int	.horizontalTextPosition	Layout
Icon-Text Spacing	The space (in pixels) between the icon (if any) and the text (if any).	int	.iconTextGap	Appearance
Image Path	The relative path of the image.	String	.path	Appearance
Indicator Value	Bind this to the tag that indicates the current state. (If you don't have separate tags for status and control, this is bound to the same location as *Control Value*)	int	.indicatorValue	Data
Margin	The space between a button's text and its borders.	Insets	.margin	Layout
Mnemonic	A single letter that will activate the button using 'ALT-*mnemonic*'.	String	.mnemonicChar	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Rollover	If true, the button may indicate that the mouse is hovering over it.	boolean	.rolloverEnabled	Behavior
State 1 Value	The value that will be written to **controlValue** when the button is pushed in state 2.	int	.state1Value	Data
State 2 Value	The value that will be written to **controlValue** when the button is pushed in state 1.	int	.state2Value	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance

Text	Text of this component.	String	.text	Appearance
Vertical Alignment	The vertical alignment of the button's contents (text and/or image).	int	.verticalAlignment	Layout
Vertical Text Position	The vertical position of the button's text relative to its image.	int	.verticalTextPosition	Layout
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated
Opaque	Is this button completely opaque? Most aren't, so this should usually be false.	boolean	.opaque	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event.
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when a component that had the input focus lost it to another component.

.source	The component that fired this event
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

An integer that indicates whether the state was changed to "Selected" (on) or "Deselected" (off). Compare this to the event object's constants to determine what the new state is.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is released and the source component has the input focus. Works for all characters, including non-printable ones, such as SHIFT and F3.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is pressed and then released when source component has the input focus. Only works for characters that can be printed on the screen.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires *aft er* the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

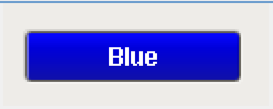
.source	The component that fired this event.
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed.
. property Name	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

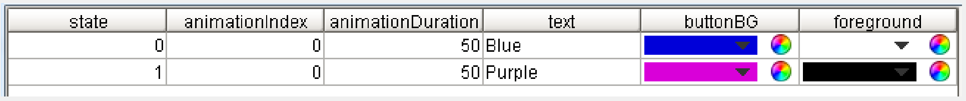
Customizers

- Component Customizers
- Style Customizer

Examples

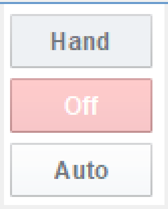
Vertical Slider with Border and Blue Text



Property Name	Dataset
Styles	

state	animationIndex	animationDuration	text	buttonBG	foreground
0	0	50	Blue	🟦	
1	0	50	Purple	🟪	⬛

Vision - Multi-State Button

General
 **Component Palette Icon:**

Description
This button is really a series of two or more buttons, arranged in a column, row, or grid. Each button represents an integer-valued state. Each state defines two styles for a button: the selected style, and the unselected style. Each button is automatically displayed with the correct style based on the current state (the value of Indicator Value). When a button is pressed then released, its state's value is written to the Control Value. To configure a Multi-State Button, simply drag a Tag that represents your state onto the Multi-State Button. This will bind both the Control Value and Indicator Value to that Tag. Now open up the Multi-State Button customizer, and define your states: their order, values and styles. Lastly choose if you want the buttons to be a column, row, or grid by setting the Display Style property.

Properties

Name	Description	Property Type	Scripting	Category
Confirm Text	The message to ask the user if Confirm is turned on. Default is "Are you sure?"	string	.confirmText	Behavior
Confirm?	If true, a confirmation box will be shown.	boolean	.confirm	Behavior
Control Value	Bind this to the tag that controls the state. (Typically, this is bound to the same location as Indicator Value.)	int	.controlValue	Data
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Display Style	The display style (rows or columns) for this N-state button.	int	.displayStyle	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Focusable	If a button is not focusable, you will not be able to interact with it with the keyboard. This means you can't "tab" over to it.	boolean	.focusableEnabled	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Grid Cols	The number of columns if the Display Style is set to "Grid" mode.	int	.gridCols	Appearance
Grid Rows	The number of rows if the Display Style is set to "Grid" mode.	int	.gridRows	Appearance
Horizontal Gap	The horizontal spacing between buttons.	int	.hGap	Appearance
Indicator Value	Bind this to the tag that indicates the current state. (Typically, this is bound to the same location as Control Value.)	int	.indicatorValue	Data
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any tag bindings on this component.	QualityCode	.quality	Data
Rollover	If true, the button may indicate that the mouse is hovering over it.	boolean	.rolloverEnabled	Behavior
States	A Dataset that stores the information for the different states.	Dataset	.states	Behavior
Vertical Gap	The vertical spacing between buttons.	int	.vGap	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

An integer that indicates whether the state was changed to "Selected" (on) or "Deselected" (off). Compare this to the event object's constants to determine what the new state is.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is released and the source component has the input focus. Works for all characters, including non-printable ones, such as SHIFT and F3.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is pressed and then released when source component has the input focus. Only works for characters that can be printed on the screen.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires *after* the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

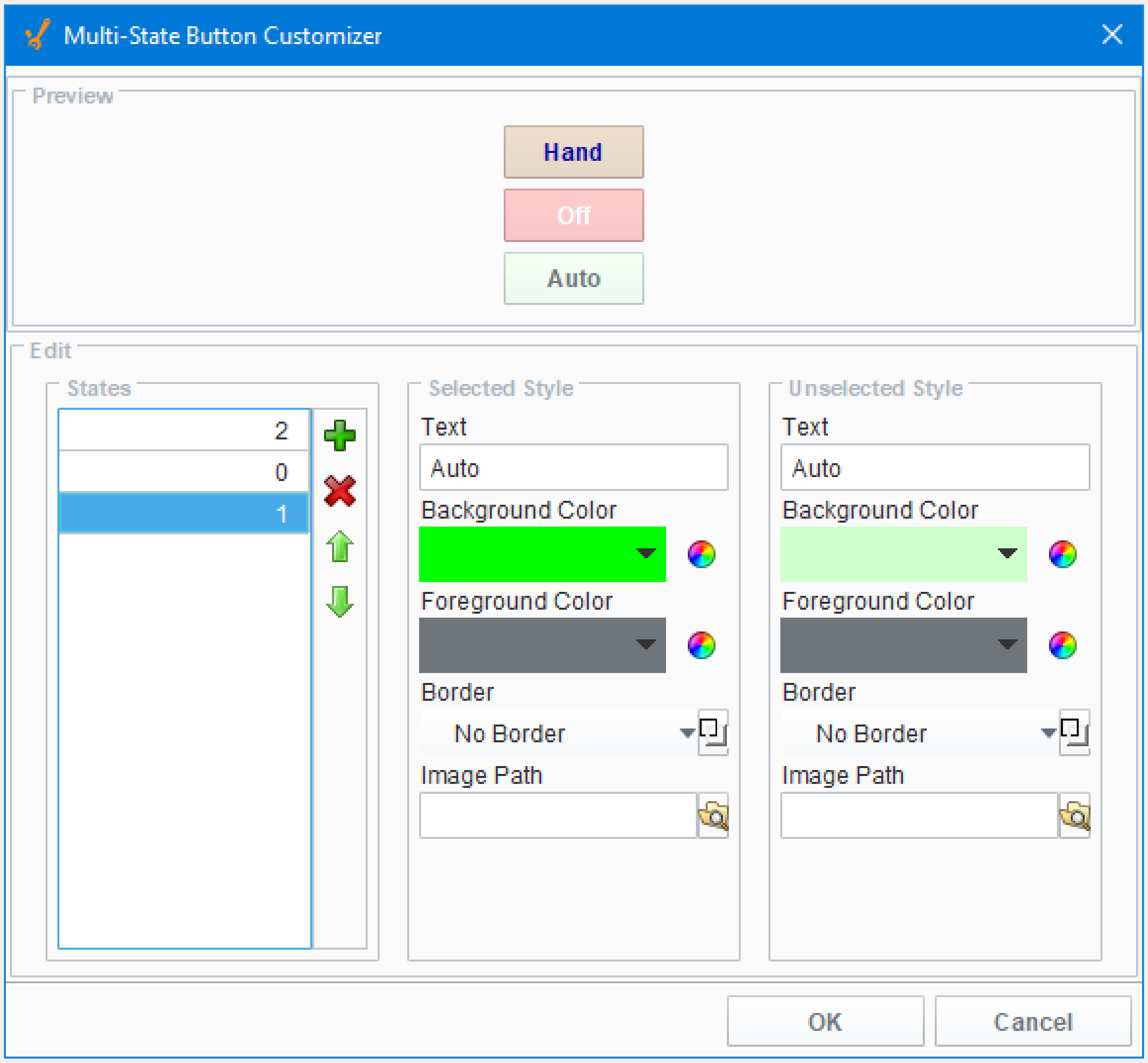
Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed.
.propertyName	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

The Multi-State Button has its own Customizer. Here, you can define your states, change the order, values, and styles. You can organize your buttons to be a column, row, or grid by setting the **Display Style** property in the Property Editor. You'll notice that the Multi-State Button Customizer already has some preset states and pre-defined styles to help you get started.

The Multi-State Button works by defining a set of visual styles that change based on a single State. When one of the buttons is pressed, its state value is written to the Control Value property and the Indicator value, and then the visual style will change.



Multi-State Button Customizer Property Descriptions

Property	Description
Preview	Lets you preview each button's display style, states, and the selected style and unselected style as you configure it.
States	Shows a list of all possible states. You can add, remove, and the change the order of each state listed. Each state also defines two visual styles for a button: Selected Style and Unselected Style.
Selected Style	Shows the visual style when the button is selected. You can configure the styles you want to change: **Text**, **Back ground Color**, **Foreground Color**, **Border** type, and even add an **Image**.
Unselect ed Style	Shows the visual style when the button is *not* selected. You can configure the styles you want to change: **Text**, **B ackground Color**, **Foreground Color**, **Border** type, and even add an **Image**.
Text	Text displayed on the button.
Backgrou nd Color	Color of the button
Foregrou nd Color	Color of the text
Border	Type of border around the button
Image Path	Relative path name for an image on the button

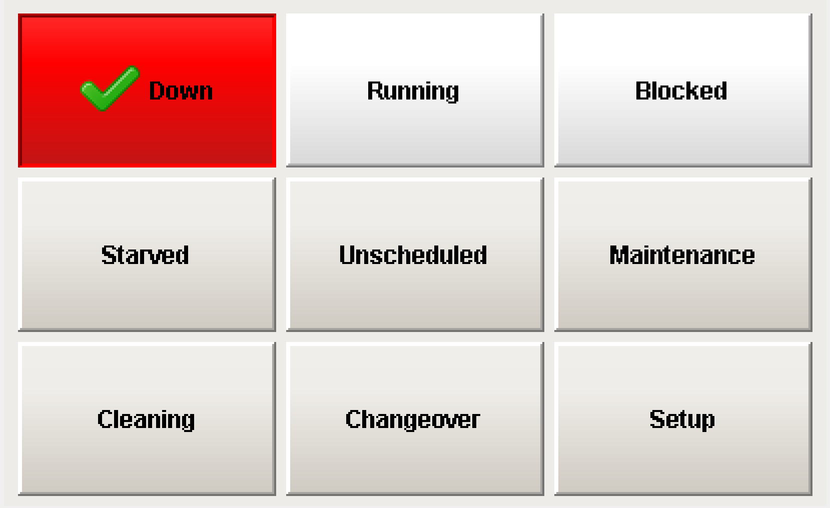
For additional customizers, see:

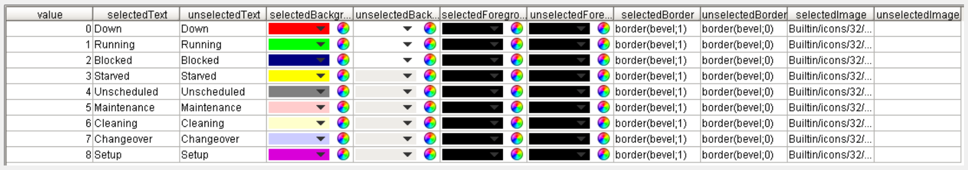
- Vision Component Customizers
- Style Customizer

Examples

The Multi-State Button component in this example has its **Display Style** property set to **Grid**. Each button represents an integer-valued state. Each state defines two styles for a button: the selected style, and the unselected style. When a button is pressed, its state's value is written to the submit your changes property. The displayed state is based on the **Indicator Value** property.

Stylized Multi-State Button

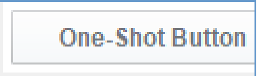


Property Name	Value
Display Style	Grid
Styles	

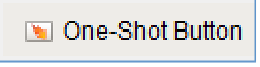
value	selectedText	unselectedText	selectedBackgr...	unselectedBack...	selectedForegro...	unselectedFore...	selectedBorder	unselectedBorder	selectedImage	unselectedImage
0	Down	Down	🟥		⬛	⬛	border(bevel;1)	border(bevel;0)	Builtin/icons/32/...	
1	Running	Running	🟩		⬛	⬛	border(bevel;1)	border(bevel;0)	Builtin/icons/32/...	
2	Blocked	Blocked	🟦		⬛	⬛	border(bevel;1)	border(bevel;0)	Builtin/icons/32/...	
3	Starved	Starved	🟨	⬜	⬛	⬛	border(bevel;1)	border(bevel;0)	Builtin/icons/32/...	
4	Unscheduled	Unscheduled	▩	⬜	⬛	⬛	border(bevel;1)	border(bevel;0)	Builtin/icons/32/...	
5	Maintenance	Maintenance	🟥	⬜	⬛	⬛	border(bevel;1)	border(bevel;0)	Builtin/icons/32/...	
6	Cleaning	Cleaning	🟨	⬜	⬛	⬛	border(bevel;1)	border(bevel;0)	Builtin/icons/32/...	
7	Changeover	Changeover	🟪	⬜	⬛	⬛	border(bevel;1)	border(bevel;0)	Builtin/icons/32/...	
8	Setup	Setup	🟪	⬜	⬛	⬛	border(bevel;1)	border(bevel;0)	Builtin/icons/32/...	

Vision - One-Shot Button

General



Component Palette Icon:



Description

The One-Shot button is great for telling a PLC to do something. It simply writes a value, and then waits for it to be reset by the PLC before it is available again. This is only applicable when the PLC is programmed to reset the value after reading it. If your PLC expects the HMI to reset the bit, use the Momentary Button. Also note that this component is considered safer than the momentary button, because it receives positive feedback from the PLC that the signal was received, avoiding the timing dangers associated with a Momentary Button.

To use the One-Shot button, bind an OPC tag bidirectionally to the button's Value property. When clicked, the button will write the value in its Set Value property to the Value property. Typically, Set Value is 1, and Value is 0 in a ready state, although the logic could be reversed or change simply by altering Set Value. The button can disable itself when it is writing, and will display different text. Note that the button considers itself to be writing whenever Value equals Set Value - you must make sure that the PLC resets this value, otherwise the button will remain in a writing state.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the button. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.buttonBG	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Border Painted?	Indicates whether the border of this button will be displayed.	boolean	.borderPainted	Appearance
Confirm Text	The message to ask the user if confirmation is turned on.	String	.confirmText	Behavior
Confirm?	If true, a confirmation box will be shown.	boolean	.confirm	Behavior
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Disable While Writing	If true, the button will be disabled while it is writing.	boolean	.disableWhileWriting	Behavior
Disabled Image Path	The relative path of the image to be displayed when this component is not enabled.	String	.disabledPath	Appearance

Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Fill Area?	Controls whether or not this button's internal area is filled	boolean	.contentAreaFilled	Appearance
Focusable	If a button is not focusable, you will not be able to interact with it with the keyboard. This means you can't "tab" over to it.	boolean	.focusable	Behavior
Font	Font of text on this component	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.foreground	Appearance
Horizontal Alignment	The horizontal alignment of the button's contents (text and/or image).	int	.horizontalAlignment	Layout
Horizontal Text Position	The horizontal position of the button's text relative to its image.	int	.horizontalTextPosition	Layout
Icon-Text Spacing	The space (in pixels) between the icon (if any) and the text (if any).	int	.iconTextGap	Appearance
Idle Text	The text of the button while its value is not being written.	String	.normalText	Behavior
Image Path	The relative path of the image.	String	.path	Appearance
Margin	The space between a button's text and its borders.	Insets	.margin	Layout
Mnemonic	A single letter that will activate the button using 'ALT-<i>mnemonic</i>'.	String	.mnemonicChar	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Opaque	Is this button completely opaque? Most aren't, so this should usually be false.	boolean	.opaque	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Rollover	If true, the button may indicate that the mouse is hovering over it.	boolean	.rolloverEnabled	Behavior
Set Value	The value to set the control value to when the button is pushed.	int	.setValue	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Value	The current value. Should be bound bi-directionally to a tag.	int	.value	Data
Vertical Alignment	The vertical alignment of the button's contents (text and/or image).	int	.verticalAlignment	Layout
Vertical Text Position	The vertical position of the button's text relative to its image.	int	.verticalTextPosition	Layout
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Writing Text	The text of the button while its value is being written.	String	.writePendingText	Behavior
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event is fired when the 'action' of the component occurs. This means when somebody selects the radio button.

.source	The component that fired this event

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event.
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when a component that had the input focus lost it to another component.

.source	The component that fired this event
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

An integer that indicates whether the state was changed to "Selected" (on) or "Deselected" (off). Compare this to the event object's constants to determine what the new state is.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is released and the source component has the input focus. Works for all characters, including non-printable ones, such as Shift and F3.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is pressed and then released when source component has the input focus. Only works for characters that can be printed on the screen.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

This event fires after the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

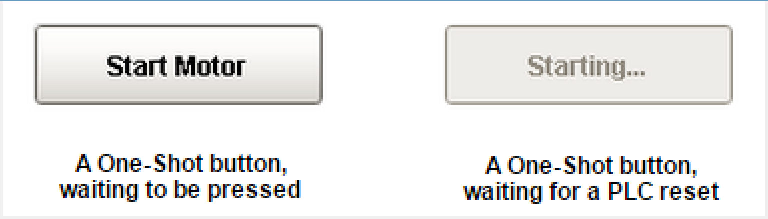
.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

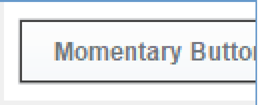
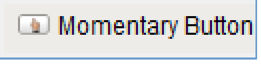
- Vision Component Customizers
- Style Customizer

Examples

One Shot Button


Start Motor
Starting...
A One-Shot button, waiting to be pressed
A One-Shot button, waiting for a PLC reset

Vision - Momentary Button

General

Component Palette Icon:


Description
Momentary buttons are used to set a value for either a fixed amount of time, or however long the button remains held down, whichever is longer. Once the button is released, or the minimum time expires, the value is reset. The momentary button uses its Control Value property to affect the underlying data. Typically, this property uses a bidirectional tag binding to an OPC tag. When pressed, it will write its On Value to the Control Value property. When released, it will either write Off Value to the Control Value immediately, or wait until On Time has elapsed (since the pressed event). The button's Indicator Value, which is typically bound to the same OPC tag as Control Value, is used to draw an "active" indication border around the button. This gives the operator positive feedback that the value has written successfully. It also lets an operator at one terminal know if an operator at a different terminal is using the button currently.

If the client is closed before the **Min Hold Time** period on the Momentary Button expires, then it is possible for the button to remain in the **ON** or latched state. Thus, if the **Control Value** property of the component is bound to a tag, the tag will remain in the **ON** state after the client is closed. Some logic or functionality will need to be applied to reset the tag in this scenario: typically the PLC is relied on in these scenarios to reset the value

Alternatively, you may wish to use a Vision - One-Shot Button instead, as that component was designed for use in situations where the PLC will reset the value.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the button. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.buttonBG	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border.	Border	.innerBorder	Common
Control Value	Bind this to the tag that you want to control. (Typically, this is bound to the same location as **Indicator Value**).	int	.controlValue	Data
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Disabled Image Path	The relative path of the image to be displayed when this component is not enabled.	String	.disabledPath	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common

Fill Area?	Controls whether or not this button's internal area is filled.	boolean	.contentAreaFilled	Appearance
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.foreground	Appearance
Horizontal Alignment	The horizontal alignment of the button's contents (text and/or image).	int	.horizontalAlignment	Layout
Horizontal Text Position	The horizontal position of the button's text relative to its image.	int	.horizontalTextPosition	Layout
Icon-Text Spacing	The space (in pixels) between the icon (if any) and the text (if any).	int	.iconTextGap	Appearance
Image Path	The relative path of the image.	String	.path	Appearance
Indicator Value	Bind this to the tag that indicates the current state of the control value. (Typically, this is bound to the same location as <i>Control Value</i>).	int	.indicatorValue	Data
Indicator Width	The width of the indication border that shows whether or not the indicator value is currently set.	int	.indicatorWidth	Appearance
Max Hold Time	The maximum amount of time to keep the control value at the "On Value". When set to 0, this property is ignored.	int	.maxOnTime	Behavior
Min Hold Time	The minimum amount of time to keep the control value at the "On Value".	int	.onTime	Behavior
Mnemonic	A single letter that will activate the button using 'ALT-<i>mnemonic</i>'.	String	.mnemonicChar	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Off Color	The color of the indicator border when the indicator value is off. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.offColor	Appearance
Off Value	The value that will be written to the Control Value on mouse-up.	int	.offValue	Behavior
On Color	The color of the indicator border when the indicator value is on. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.onColor	Appearance
On Value	The value that will be written to the Control Value on mouse-down.	int	.onValue	Behavior
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Rollover?	If true, the button may indicate that the mouse is hovering over it.	boolean	.rolloverEnabled	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance

Text	Text of this component.	String	.text	Appearance
Vertical Alignment	The vertical alignment of the button's contents (text and/or image).	int	.verticalAlignment	Layout
Vertical Text Position	The vertical position of the button's text relative to its image.	int	.verticalTextPosition	Layout
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event is fired when the 'action' of the component occurs. This means when somebody selects the radio button.

.source	The component that fired this event

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event.
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when a component that had the input focus lost it to another component.

.source	The component that fired this event
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

An integer that indicates whether the state was changed to "Selected" (on) or "Deselected" (off). Compare this to the event object's constants to determine what the new state is.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is released and the source component has the input focus. Works for all characters, including non-printable ones, such as SHIFT and F3.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is pressed and then released when source component has the input focus. Only works for characters that can be printed on the screen.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires *aft er* the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
. clickCo unt	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
. popupT rigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
. altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
. control Down	True (1) if the Control key was held down during this event, false (0) otherwise.
. shiftDo wn	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
. clickCo unt	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
. popupT rigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
. altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
. control Down	True (1) if the Control key was held down during this event, false (0) otherwise.
. shiftDo wn	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

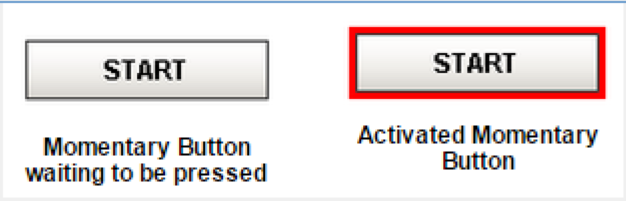
.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers
- Style Customizer

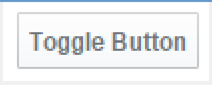
Examples

Vertical Slider with Border and Blue Text

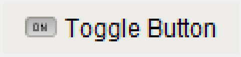

START
Momentary Button waiting to be pressed
START
Activated Momentary Button

Vision - Toggle Button

General



Component Palette Icon:



Description

The Toggle button represents a bit: on (selected) or off (not selected). Visually the button looks down or depressed when it is selected, and up when it is not selected. Logically, this component is very similar to the Check Box component. Note that for implementing a controls screen, the 2 State Toggle is usually more appropriate than this component.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the button. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.buttonBG	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Border Painted?	Indicates whether the border of this button is displayed.	boolean	.borderPainted	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Fill Area?	Controls whether or not this button's internal area is filled.	boolean	.contentAreaFilled	Appearance
Focusable	If a button is not focusable, you will not be able to interact with it with the keyboard. This means you can't "tab" over to it.	boolean	.focusable	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.foreground	Appearance
Image Path	The relative path of the image.	String	.path	Appearance
Label	Text displayed on this button.	String	.text	Appearance

Margin	The space between a button's text and its borders.	Insets	.margin	Layout
Mouse over Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Opaque	Set this to false if you want the button to be completely opaque.	boolean	.opaque	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Rollover?	If true, the button may indicate that the mouse is hovering over it.	boolean	.rolloverEnabled	Appearance
Selected	State of this toggle button.	boolean	.selected	Data
Selected Image Path	The relative path of the image to be displayed when this component is selected (toggled on).	String	.selectedPath	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event is fired when the 'action' of the component occurs. This means when somebody selects the radio button.

.source	The component that fired this event

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event.
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when a component that had the input focus lost it to another component.

.source	The component that fired this event
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when the state of the component changes, as when the radio button goes from selected to not selected.

.source	The component that fired this event
.stateChange	An integer that indicates what the state was changed to.
SELECTED	The constant that the stateChange property will be equal to if this event represents a selection.
DESELECTED	The constant that the stateChange property will be equal to if this event represents a de-selection.

An integer that indicates whether the state was changed to "Selected" (on) or "Deselected" (off). Compare this to the event object's constants to determine what the new state is.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is released and the source component has the input focus. Works for all characters, including non-printable ones, such as SHIFT and F3.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is pressed and then released when source component has the input focus. Only works for characters that can be printed on the screen.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires *after* the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

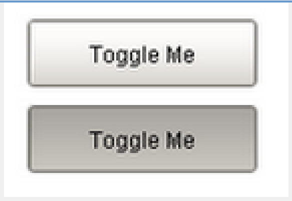
Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event.
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed.
. property Name	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

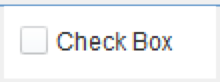
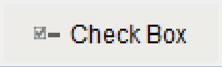
Customizers

- Vision Component Customizers
- Style Customizer

Examples



Vision - Check Box

General

Component Palette Icon:


Description
A CheckBox is a familiar component that represents a bit - it is either on (selected) or off (not selected). It is functionally equivalent to the Toggle Button component.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Fill Background	If true, the label's background color will be drawn. If false, it will have a transparent background.	boolean	.fillBackground	Appearance
Focusable	If a button is not focusable, you will not be able to interact with it with the keyboard. This means you can't "tab" over to it.	boolean	.focusable	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.foreground	Appearance
Horizontal Alignment	The horizontal alignment of the button's contents (text and/or image).	int	.horizontalAlignment	Layout
Margin	The internal margin that provides padding for the contents of this button.	Insets	.margin	Appearance

Mouse over Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Rollover	If true, the button may indicate that the mouse is hovering over it.	boolean	.rolloverEnabled	Behavior
Selected	The current state of the checkbox.	boolean	.selected	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Text	The text displayed on the checkbox.	String	.text	Appearance
Vertical Alignment	The vertical alignment of the button's contents (text and/or image).	int	.verticalAlignment	Layout
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event is fired when the 'action' of the component occurs. This means when somebody selects the radio button.

.source	The component that fired this event

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event.
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when a component that had the input focus lost it to another component.

.source	The component that fired this event
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when the state of the component changes, as when the radio button goes from selected to not selected.

.source	The component that fired this event
.stateChange	An integer that indicates what the state was changed to.
SELECTED	The constant that the stateChange property will be equal to if this event represents a selection.
DESELECTED	The constant that the stateChange property will be equal to if this event represents a de-selection.

An integer that indicates whether the state was changed to "Selected" (on) or "Deselected" (off). Compare this to the event object's constants to determine what the new state is.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is released and the source component has the input focus. Works for all characters, including non-printable ones, such as SHIFT and F3.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is pressed and then released when source component has the input focus. Only works for characters that can be printed on the screen.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires *aft er* the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
. clickCo unt	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
. popupT rigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
. altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
. control Down	True (1) if the Control key was held down during this event, false (0) otherwise.
. shiftDo wn	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
. clickCo unt	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
. popupT rigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
. altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
. control Down	True (1) if the Control key was held down during this event, false (0) otherwise.
. shiftDo wn	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

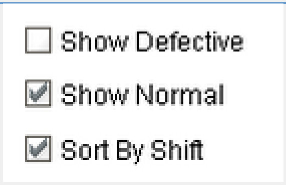
Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed.
.propertyName	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

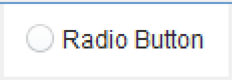
- Vision Component Customizers
- Style Customizer

Examples



Vision - Radio Button

General



Component Palette Icon:

Description

The radio button is similar to the CheckBox component, except for one special property. All radio buttons in the same Container (including the Root Container) will automatically be mutually exclusive. This means that only one radio button can be selected at a time. Radio buttons are a good way to let the user choose just one of a number of options. Dropdown Lists are another good way to do this.

Properties

Name	Description	Property Type	Scripting	category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector .	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Fill Background	If true, the label's background color will be drawn. If false, it will have a transparent background.	boolean	.fillBackground	Appearance
Focusable	If a button is not focusable, you will not be able to interact with it with the keyboard. This means you can't "tab" over to it.	boolean	.focusable	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector .	Color	.foreground	Appearance
Horizontal Alignment	The horizontal alignment of the button's contents (text and/or image).	int	.horizontalAlignment	Layout
Margin	The internal margin that provides padding for the contents of this button.	Insets	.margin	Appearance

Mouse over Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Rollover	If true, the button may indicate that the mouse is hovering over it.	boolean	.rolloverEnabled	Behavior
Selected	The current state of the RadioButton.	boolean	.selected	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Text	Text of this component.	String	.text	Appearance
Vertical Alignment	The vertical alignment of the button's contents (text and/or image).	int	.verticalAlignment	Layout
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event is fired when the 'action' of the component occurs. This means when somebody selects the radio button.

.source	The component that fired this event

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event.
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when a component that had the input focus lost it to another component.

.source	The component that fired this event
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when the state of the component changes, as when the radio button goes from selected to not selected.

.source	The component that fired this event
.stateChange	An integer that indicates what the state was changed to.
SELECTED	The constant that the stateChange property will be equal to if this event represents a selection.
DESELECTED	The constant that the stateChange property will be equal to if this event represents a de-selection.

An integer that indicates whether the state was changed to "Selected" (on) or "Deselected" (off). Compare this to the event object's constants to determine what the new state is.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is released and the source component has the input focus. Works for all characters, including non-printable ones, such as SHIFT and F3.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is pressed and then released when source component has the input focus. Only works for characters that can be printed on the screen.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires *aft er* the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
. clickCo unt	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
. popupT rigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
. altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
. control Down	True (1) if the Control key was held down during this event, false (0) otherwise.
. shiftDo wn	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
. clickCo unt	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
. popupT rigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
. altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
. control Down	True (1) if the Control key was held down during this event, false (0) otherwise.
. shiftDo wn	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

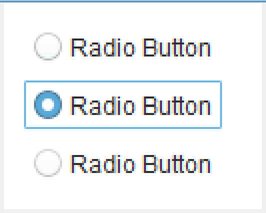
.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed.
.propertyName	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers
- Style Customizer

Examples

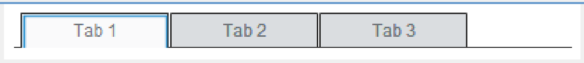
A Selection of Radio Buttons



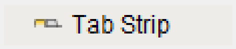
Radio buttons inside a container will be exclusive therefore selecting one radio button will de-select the other radio buttons.

Vision - Tab Strip

General



Component Palette Icon:



Description

In general, a Tab Strip is just a single-selection multiple choice component. In practice it is used anywhere that a user needs to be able to select between multiple windows or to select between containers to display. It is most commonly used in a docked window to provide automatic window navigation. To support this typical use-case, the tab strip has two navigation modes:

1. **Swap to Window** - (default) The Tab Strip will automatically call system.nav.swapTo() with the name of the selected tab. This facilitates very easy navigation for most common projects.
2. **Disabled** - The Tab Strip doesn't do anything when the tab selection changes. Users can implement their own via property bindings or by responding to the propertyChange scripting event.

The Tab Strips visual style is highly customizable. There are different rendering styles, and things such as fonts, colors, line thicknesses, hover colors, and gradients are customizable within each rendering style. Use the Tab Strip's customizer to come up with a style that suits your project, as well as to manage the tabs that are present. The tabs and their styles are all stored in a dataset property (called Tab Data), so they can be modified at runtime as well.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are:No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Intertab Space	The amount of space between each tab.	int	.interTabSpace	Appearance
Name	The name of this component.	String	.name	Common
Navigation Mode	Navigation mode. Disabled does nothing when a tab is pressed. Swap to window swaps to the window whose name corresponds to the name of the selected tab, provided that window exists.	int	.navigationMode	Behavior
Orientation	Orientation of the tab strip.	int	.orientation	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data

Renderer	The renderer to use when rendering tabs.	int	.renderer	Appearance
Rounding Radius	Rounding radius for the tab corners.	int	.roundingRadius	Appearance
Selected Tab	Name of the selected tab. This is also the name of the window that, if it exists, will be swapped to when this tab is pressed.	String	.selectedTab	Appearance
Separator Color	Color of the line drawn across the bottom and around each tab. See Color Selector.	Color	.separatorColor	Appearance
Separator Thickness	Thickness of the line drawn across the bottom and around each tab.	float	.separatorThickness	Appearance
Size Mode	The sizing mode tabs use when deciding their size. Automatic means every tab is the same fixed size. Individual lets each tab decide its own size based on the size of its text.	int	.sizeMode	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Tab Data	Tab data to be displayed.	Dataset	.tabData	Data
Text Alignment	The alignment of the tab text.	int	.textAlignment	Appearance
Text Offset	Padding on the left or right side of tab's text, depending on alignment.	int		Appearance
Text Padding	Padding on each side of the text inside a tab.	int	.textPadding	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

The following feature is new in Ignition version **8.0.16**
Click here to check out the other new features

As of 8.0.16, the Tab Strip's "mouse" events .x and .y coordinates are now based on cursor position over the entire component, as opposed to coordinates based on the individual tab that the event was triggered from.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

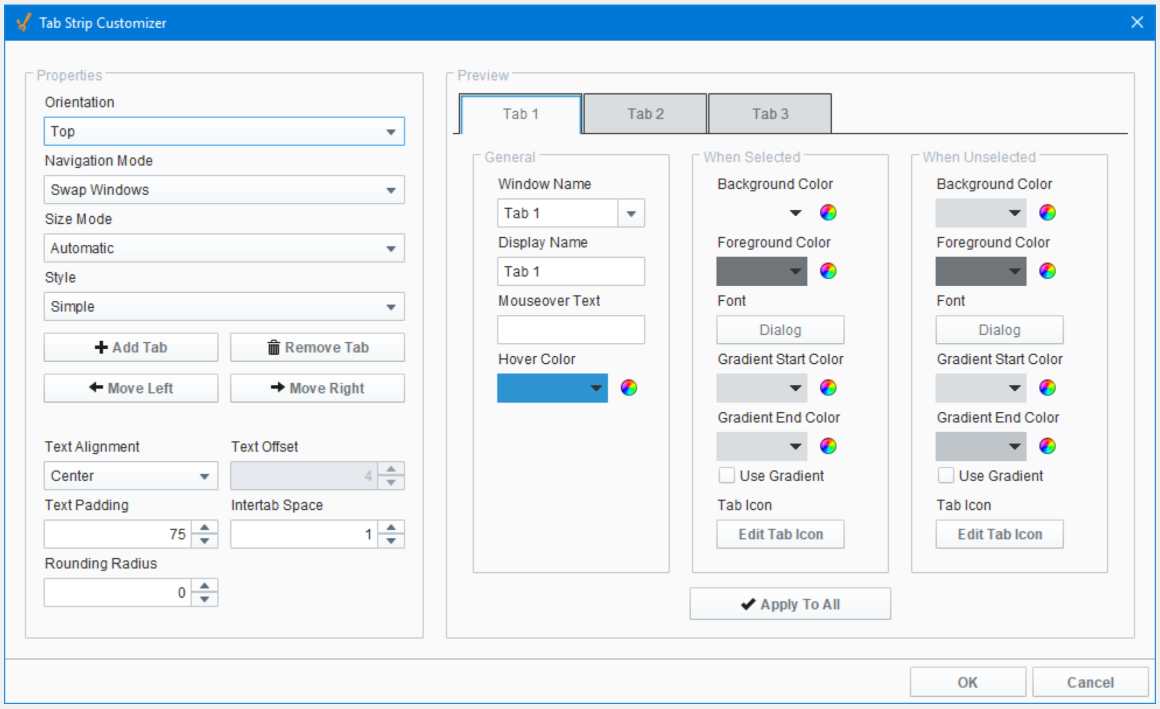
Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

The Tab Strip Customizer has its own set of properties that you can set and modify which dictate how the Tab Strip component looks and behaves whether or not it is used for window navigation. The tabs and the styles are stored in the **Tab Data** dataset property.

When customizing the Tab Strip, keep in mind how you are using the component when setting your properties. Some Tab Strip properties may behave a little differently based on style, tab orientation, or text alignment. It's a good idea to use the preview window to verify the style you configured is the style you want.



Orientation: Top
Navigation Mode: Swap Windows
Size Mode: Automatic
Style: Simple
Text Alignment: Center
Text Offset: 4
Text Padding: 75
Intertab Space: 1
Rounding Radius: 0

Window Name: Tab 1
Display Name: Tab 1
Mouseover Text:
Hover Color

When Selected — Background Color, Foreground Color, Font: Dialog, Gradient Start Color, Gradient End Color, Tab Icon

When Unselected — Background Color, Foreground Color, Font: Dialog, Gradient Start Color, Gradient End Color, Tab Icon

Tab Strip Customizer - Property Descriptions

Properties	Description
Orientation	Orientation of the Tab Strip on a window: **Top**, **Left**, **Bottom** and **Right**. For example, use the **Top** orientation to place the Tab Strip component at the top of your window.

Navigation Mode	Two Navigation modes: • **Swap Windows** - the Tab Strip automatically calls system.nav.swapTo() to perform a window swap from the current window to another window when a tab is pressed. Swap Windows is the default mode. • **Disabled** - the Tab Strip only sets the **Selected Tab** property when pressed. You can set the component's behavior using property bindings or by responding to the propertyChange scripting event.
Size Mode	Two Size modes: • **Individual** - all the tabs are the same size. • **Automatic** - all the tabs are sized to fit the text.
Style	Three style options to change the appearance of the individual tabs: **Simple**, **Fancy**, and **Folder**.
Add Tab	Adds a new tab next to the selected tab.
Remove Tab	Removes a selected tab.
Move Up / Move Down	Depends on the current Orientation selection. Moves the selected tab **Up** or **Down** in the tab strip when using the **Left** or **Right orientation**.
Move Left / Move Right	Depends on the current Orientation selection. Moves the selected tab either **Left** or **Right** in the tab strip when using the **Top / Bottom orientation.**
Text Alignment	Inserts text in the Center, Left, or Right inside a tab.
Text Offset	Specifies how many pixels to move text to the left or right within a tab.
Text Padding	Specifies the number of pixels around the text in the tab.
Intertab Space	Specifies the number of pixels between tabs.
Rounding Radius	Specifies the number of pixels to round the corners of the tab depending on the tab orientation.
General	
Window Name	Pathname of the window location
Display name	The name to display on the tab.
Mouseover Text	The text to display in the tooltip which pops up when mousing over a tab.
Hover Color	The color to display in the tootip which pops up when mousing over a tab.
When Selected / When Unselected	
Background Color	The background color of the tab.
Foreground Color	The foreground color is the color of the text.
Font	Select the font type, font size, and style.
Gradient Start Color	Select a start color to begin the gradient. Gradients are not valid for the Fancy style, and are shown as being grayed out. Select Simple or Folder style to use the gradient feature.
Gradient End Color	Select an end color to end the gradient. Gradients are not valid for the Fancy style, and are shown as being grayed out. Select Simple or Folder style to use the gradient feature.
Use Gradient	Select Use Gradient checkboxes to use gradient features. Uncheck the Use Gradient checkboxes to disable the gradient feature.

Tab Icon	Select an image from the Image Browsser to insert on a tab.
Apply to All	The button applies all of the currently shown settings (except Window Name and Display Name) to all of the tabs. Note: this does not save your changes.

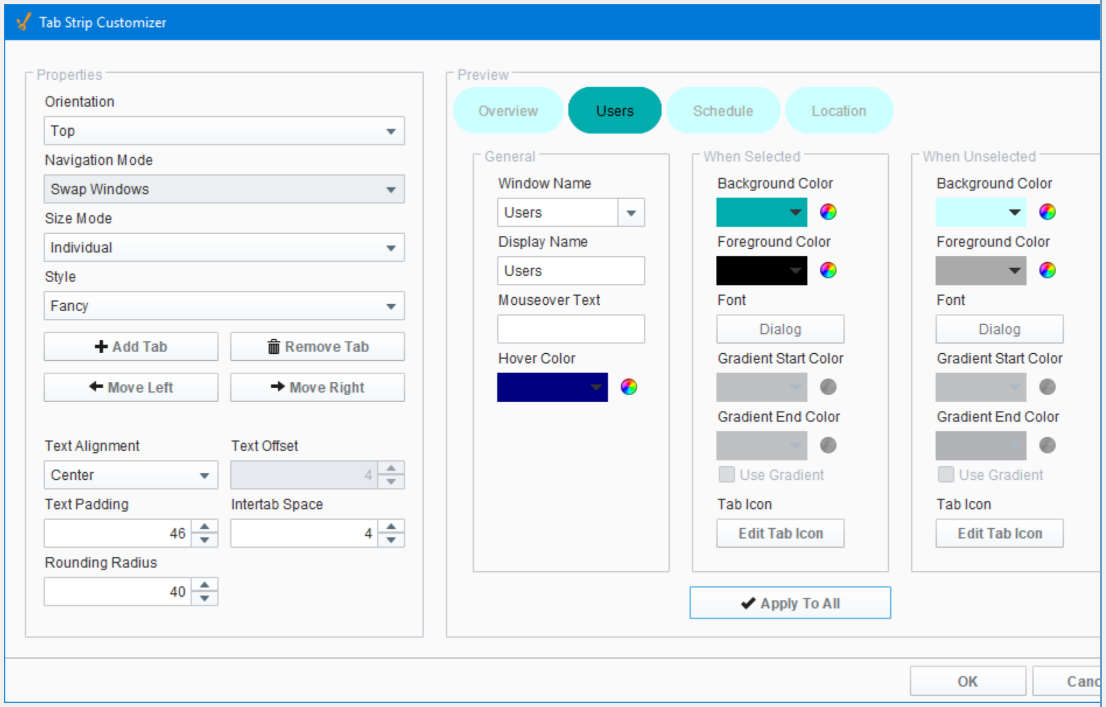
- Navigation - Tab Strip
- Vision Component Customizers
- Style Customizer

Examples

Horizontal Tabs

Horizontal Tab - Property Descriptions

Property Name	Value
Style	Fancy
Orientation	Top
Tab Data	Dataset customized with the Tab Strip Customizer. Notice how the Gradient features are grayed out with the Fancy style.


Tab Strip Customizer

Properties
Orientation: Top
Navigation Mode: Swap Windows
Size Mode: Individual
Style: Fancy
Text Alignment: Center
Text Offset: 4
Text Padding: 46
Intertab Space: 4
Rounding Radius: 40

Preview
General
Window Name: Users
Display Name: Users
Mouseover Text:
Hover Color

When Selected
Background Color
Foreground Color
Font: Dialog
Gradient Start Color
Gradient End Color
Use Gradient
Tab Icon

When Unselected
Background Color
Foreground Color
Font: Dialog
Gradient Start Color
Gradient End Color
Use Gradient
Tab Icon

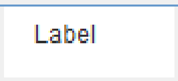
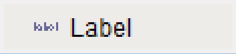
Vision - Display Palette

Display Components

The following components give you various options for displaying values and more.

In This Section ...

Vision - Label

General

Component Palette Icon:


Description
The Label is one of the most versatile components. It can display text, images, or both. Its text can be HTML formatted (like most components). It can even be made to respond to user interaction through its events. Labels are one of the most common components that you will want to add dynamic properties to. For instance, you can put an integer dynamic property "state" on a label, and then bind the text to be "On" when the state is 1 and "Off" otherwise, using an expression binding. Bind the background color to be red when the state is 0, and green when the state is 1 using a property binding. Now you have a re-usable binary state indicator. While you could have used the Multi-State Indicator to achieve the same effect, the exercise is good practice for creating custom components. You can see how the flexibility of bindings and dynamic properties make the Label extremely versatile.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the label, if opaque is set to "true". Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Disabled Image Path	The relative path of the image to be displayed when this component is not enabled.	String	.disabledPath	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Fill Background	If true, the label's background color will be drawn. If false, it will have a transparent background.	boolean	.fillBackground	Appearance
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The color of the Label's text.	Color	.foreground	Appearance

Horizontal Alignment	Determines the alignment of the label's contents along the X axis.	int	.horizontalAlignment	Layout
Horizontal Text Position	Determines the horizontal position of the label's text, relative to its image.	int	.horizontalTextPosition	Layout
Icon-Text Spacing	The space (in pixels) between the icon (if any) and the text (if any).	int	.iconTextGap	Appearance
Image Path	The relative path of the image.	String	.path	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Deprecated
Rotation	The angle of rotation in degrees.	int	.rotation	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Text	Text of this Label.	String	.text	Data
Vertical Alignment	Determines the alignment of the label's contents along the Y axis.	int	.verticalAlignment	Layout
Vertical Text Position	Determines the vertical position of the label's text, relative to its image.	int	.verticalTextPosition	Layout
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Data

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

This event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

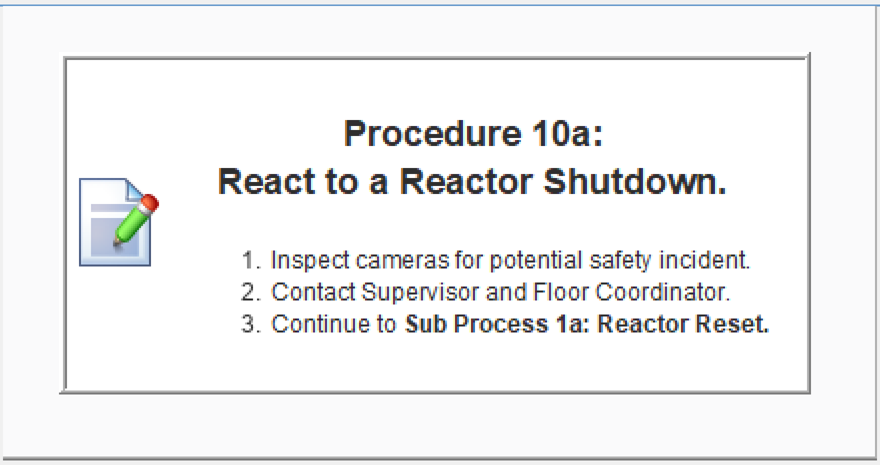
.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers
- Style Customizer

Examples

Stylized Label Inside a Popup Window



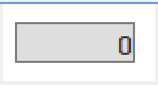
Procedure 10a:
React to a Reactor Shutdown.

1. Inspect cameras for potential safety incident.
2. Contact Supervisor and Floor Coordinator.
3. Continue to **Sub Process 1a: Reactor Reset.**

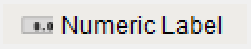
Property Name	Value
Image Path	Builtin/icons/48/document_edit.png
Text	<html><p><strong><center><h2>Procedure 10a: React to a Reactor Shutdown.</h2></center></strong></p> <ol> <li>Inspect cameras for potential safety incident.</li> <li>Contact Supervisor and Floor Coordinator.</li> <li>Continue to <strong>Sub Process 1a: Reactor Reset.</strong></li> </ol></html>

Vision - Numeric Label

General



Component Palette Icon:


Numeric Label

Description

This component is a specialized label designed to display a number. It can include units, and has an integrated number format string. By default the number is displayed bold and the units are not. This can be customized, see the Prefix and Suffix expert properties. This label's text is constructed as follows:

Prefix + numberFormat (Value, Pattern) + Suffix + Units

It is important to note that you could customize the standard Label component using custom properties and bindings to mimic this component exactly. If this component doesn't do something that you need, you can make your own numeric label and use it everywhere in your project.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Disabled Image Path	The relative path of the image to be displayed when this component is not enabled.	String	.disabledPath	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Fill Background	If true, the label's background color will be drawn. If false, it will have a transparent background.	boolean	.fillBackground	Appearance
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.foreground	Appearance

Horizontal Alignment	Determines the alignment of the label's contents along the X axis.	int	.horizontalAlignment	Layout
Horizontal Text Position	Determines the horizontal position of the label's text, relative to its image.	int	.horizontalTextPosition	Layout
Icon-Text Spacing	The space (in pixels) between the icon (if any) and the text (if any).	int	.iconTextGap	Appearance
Image Path	The relative path of the image.	String	.path	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Number Format Pattern	The number formatting string used to format the value.	String	.pattern	Appearance
Prefix	A string that will be placed before the number.	String	.prefix	Data
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Rotation	The angle of rotation in degrees.	int	.rotation	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Suffix	A string that will be placed after the number, and before the units.	String	.suffix	Data
Units	The engineering units to display after the number.	String	.units	Data
Value	The numeric value of this label.	double	.value	Data
Vertical Alignment	Determines the alignment of the label's contents along the Y axis.	int	.verticalAlignment	Layout
Vertical Text Position	Determines the vertical position of the label's text, relative to its image.	int	.verticalTextPosition	Layout
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

⚠ This event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

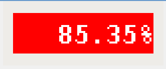
.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers
- Style Customizer

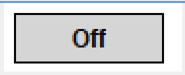
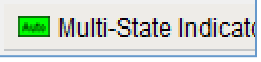
Examples

Numeric label with red background and percent sign



Property Name	Value
Units	%
Background Color	255,0,0

Vision - Multi-State Indicator

General

Component Palette Icon:


Description
This component is a specialized label used to display a discrete state. The state must be represented by an integer, but the values and number of different states is customizable. Use the component's styles customizer to configure the different states.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Disabled Image Path	The relative path of the image to be displayed when this component is not enabled.	String	.disabledPath	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.foreground	Appearance
Horizontal Alignment	Determines the alignment of the label's contents along the X axis.	int	.horizontalAlignment	Layout
Horizontal Text Position	Determines the horizontal position of the label's text, relative to its image.	int	.horizontalTextPosition	Layout
Icon-Text Spacing	The space (in pixels) between the icon (if any) and the text (if any).	int	.iconTextGap	Appearance

Image Path	The relative path of the image.	String	.path	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
State	The current state of the component.	int	.state	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Text	Text of this Label.	String	.text	Data
Vertical Alignment	Determines the alignment of the label's contents along the Y axis.	int	.verticalAlignment	Layout
Vertical Text Position	Determines the vertical position of the label's text, relative to its image.	int	.verticalTextPosition	Layout
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

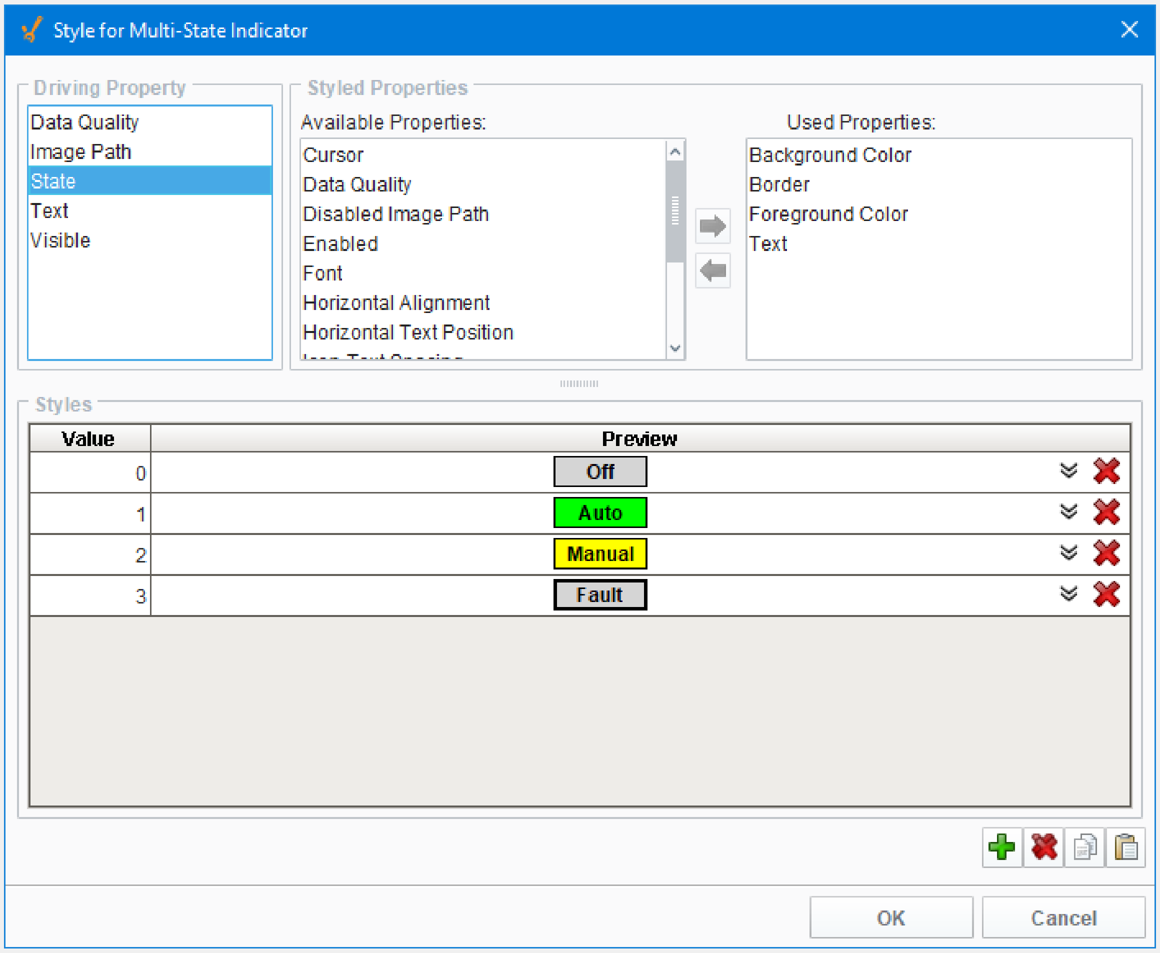
Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

The Multi-State Indicator component does not have a special customizer, however, it relies on the Style Customizer. When you open the Style Customizer, you'll notice that it has the **State** driving property selected, and several visual properties defined such as **Background Color**, **Border**, **Foreground Color**, and **Text**. If you don't like the predefined properties, you can change them, as well as add or remove any styled properties.

The Style Customizer for the Multi-State Indicator works by configuring a set of visual properties that change based on a different state. The State is represented by an integer, but the values and number of different states are customizable.



Value	Preview
0	Off
1	Auto
2	Manual
3	Fault

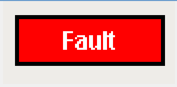
Style Customizer for the Multi-State Indicator - Property Description

Property	Description
Driving Property	Property that drives the style of the component.
Styled Properties	There are two categories of properties: **Available Properties** and **Used Properties**.
Available Properties	Styled properties that have *not* been used.
Used Properties	Styled properties that have been used.
Styles	Styles section for the defining states and styles.
Values	Driving property represented by an integer.
Preview	View the label after the visual styles are configured. Expand each value to configure, or change any of the styles. There is an Animate checkbox that you can check to enable the label to blink.

For additional Customizers, see:

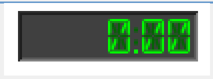
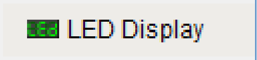
- Style Customizer
- Vision Component Customizers

Examples



Property Name	Value
Styles	As defined by the style customizer.

Vision - LED Display

General

Component Palette Icon:


Description
The LED display is a stylized numeric or alphanumeric label. It has three different visual styles which all correspond to a kind of physical display: 7-segment, 14-segment, and 5x7 matrix. By default this component is in numeric mode, which means you should use its Value property. If you need to display characters as well, switch the mode to alphanumeric, and use the Text pro perty.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The color of the background. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Horizontal Alignment	Determines the alignment of the display's contents along the X axis.	int	.horizontalAlignment	Layout
LED Lit	The color of lit LED segments. See Color Selector.	Color	.glyphForeground	Appearance
LED Unlit	The color of unlit LED segments. See Color Selector.	Color	.glyphBackground	Appearance
Letter Gap	The percentage of the height to be used as an inter-character spacing.	float	.gap	Layout
Margin	The margin for the interior of the display.	Insets	.margin	Layout
Mode	The mode of the display.	int	.mode	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Number Format Pattern	The number formatting string used to format the value.	String	.numberFormat	Behavior
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Style	The visual style of the display.	int	.style	Appearance
Styles	**Contains the component's styles.**	Dataset	.styles	Appearance
Text	The text value of the display, used when **Mode** is **Alphanumeric.**	String	.text	Data
Value	The numeric value of the display, used when **Mode** is **Numeric**.	double	.value	Data
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

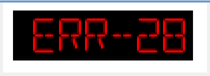
.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers
- Style Customizer

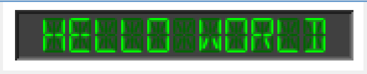
Examples

Custom LED Component



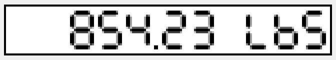
Property Name	Value
Mode	Alphanumeric
Text	ERR-28
Background Color	0,0,0
LED Lit	255,0,0
LED Unlit	0,0,0

Custom LED Component



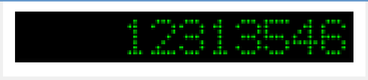
Property Name	Value
Mode	Alphanumeric
Text	Hello World
Horizontal Alignment	Center

Custom LED Component



Property Name	Value
Border	Line Border
Mode	Alphanumeric
Text	852.23 lbs
Style	7 Segment
Background Color	255,255,255
LED Lit	0,0,0
LED Unlit	255,255,255

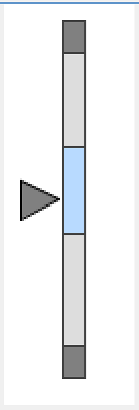
Custom LED Component



Property Name	Value
Style	5x7 Matrix
Background Color	255,255,255
Horizontal Alignment	Right

Vision - Moving Analog Indicator

General



Component Palette Icon:

Moving Analog Indicator

INDUCTIVE UNIVERSITY

Moving Analog Indicator

Watch the Video

Description

The Moving Analog Indicator is another component that fits well with the High Performance HMI techniques and practices. This component displays an analog value in context with other information about that value so that you can visually quickly see if the value is in the normal range or not. The current value is shown as an arrow pointing at a bar with segments showing the desired operating range, low and high alarm ranges, and interlock ranges.

The Moving Analog Indicator component allows for extremely fast information delivery. At a glance, it is obvious to an operator whether or not the value is where it should be, or if it needs attention. If the value is in one of its alarm ranges, then that range changes color to get attention.

To switch the Moving Analog Indicator between a horizontal vs vertical orientation, simply change the size so that it is either wide or tall, respectively. Typical setup of this component involves setting the ranges, and binding the Process Value property to a Tag's value. Some properties may be cleared out (null value) in order to disable them. For example, you may indicate where the current setpoint is by setting the Setpoint Value property. If you don't want to display the setpoint, simply clear this value out.

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Desired High	The upper value of the desired operating range.	Double	.desiredHi	Data
Desired Low	The lower value of the desired operating range.	Double	.desiredLo	Data

Desired Range Color	The color of the desired range. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.desiredRangeColor	Appearance
High Alarm	The value above which is a high alarm.	Double	.hiAlarm	Data
High High Alarm	The value above which is a high-high alarm.	Double	.hihiAlarm	Data
High Interlock	The value above which an interlock will be activated.	Double	.hiInterlock	Data
Inactive Alarm Color	The color of inactive alarm range. See Color Selector.	Color	.inactiveAlarmColor	Appearance
Interlock Color	The color of the interlock range. See Color Selector.	Color	.interlock Color	Appearance
Level 1 Alarm Color	The color of an active level 1 alarm (Hi-Hi or Lo-Lo). See Color Selector.	Color	.level1AlarmColor	Appearance
Level 2 Alarm Color	The color of an active level 2 alarm (Hi or Lo). See Color Selector.	Color	.level2AlarmColor	Appearance
Low Alarm	The value below which is a low alarm.	Double	.loAlarm	Data
Low Interlock	The value below which an interlock will be activated.	Double	.loInterlock	Data
Low Low Alarm	The value below which is a low-low alarm.	Double	.loloAlarm	Data
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Process Value	The current value of the process.	Double	.processValue	Data
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Range Fill	The background color of the range strip. See Color Selector.	Color	.rangeFill	Appearance
Range High	The overall high value for the display.	double	.rangeHi	Data
Range Low	The overall low value for the display.	double	.rangeLo	Data
Range Stroke	The stroke color for the range strip. See Color Selector.	Color	.rangeStroke	Appearance
Reverse Indicator	Put the indicator triangle on the other side of the track.	boolean	.reverseIndicatorLocation	Appearance
Setpoint Fill	The fill color of the setpoint indicator. See Color Selector.	Color	.setpointFill	Appearance

Setpoint Stroke	The stroke color of the setpoint indicator. See Color Selector.	Color	.setpointStroke	Appearance
Setpoint Value	The current value of the setpoint.	Double	.setpointValue	Data
Show Value	Show the current value above or beneath the value indicator.	boolean	.showValue	Appearance
Stroke Width	The stroke width for lines drawn.	float	.strokeWidth	Appearance
Styles	Contains the component's styles	Dataset	.styles	Appearance
Value Color	The color of the value label. See Color Selector.			
Value Font	The font for the value label.	Font	.font	Appearance
Value Format	The string format for the value, if it is shown.	String	.valueFormat	Appearance
Value Indicator Fill	The fill color of the value indicator. See Color Selector.	Color	.valueFill	Appearance
Value Indicator Stroke	The stroke color of the value indicator. See Color Selector.	Color	.valueStroke	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

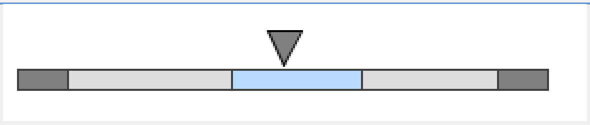
.source	The component that fired this event
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
. property Name	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers
- Style Customizer

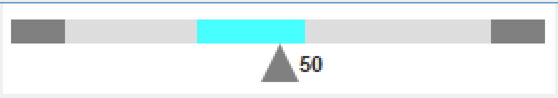
Examples

Moving Analog Indicator Expanded Horizontally



Property Name	Value
None	n/a

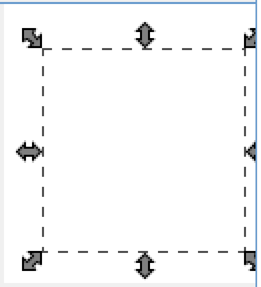
Stylized Moving Analog Indicator



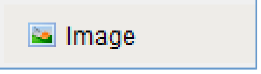
Property Name	Value
Show Value	True
Reverse Indicator	True
Stroke Width	0.0

Vision - Image

General



Component Palette Icon:



Description

The image component is a powerful component. While you can use other components, like the Label, to display images as well, this component gives you more flexibility. In particular, this component has four important features for displaying images:

1. Scaling
2. Rotation - Rotate to create spinning animations by binding to a timer component.
3. Color Tinting - Dynamically apply a color tint to an image to allow it to display real-time status
4. Color Swapping - Color swapping to change one specific color in an image to another in real time.

To choose an image, simply press the Browse icon next to this component's Image Path property. You can drag new images (*.png, *.gif, *.jpg, *.bmp) into the Image Management window to upload them.

Images are stored on the Gateway, not in your window or project. This means that you can alter an image globally, and it will affect all windows in all projects. It also means that you must be careful to migrate custom images if you do project backups (as opposed to Gateway backups, which will automatically include both projects and images)

External Images

The Image component can also be used to show external images stored relative to the local file system on the client. The file path is similar to having your browser view a local document:

```
file:///C:/folder/anotherFolder/image.PNG
```

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Color Swap Filter	Swap a specific color to another. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	boolean	.useColorSwap	Image Manipulation

Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Disabled Image Path	The relative path of the image to be displayed when this component is not enabled.	String	.disabledPath	Data
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Flip Horizontal	Flip (mirror) the image horizontally.	boolean	.flipHorizontal	Image Manipulation
Flip Vertical	Flip (mirror) the image vertically.	boolean	.flipVertical	Image Manipulation
Image Path	The relative path of the image.	String	.path	Data
Load In Background	Controls whether or not the image loading takes place on the UI thread or a background thread.	boolean	.loadInBackground	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Rotation	The angle of rotation in degrees.	int	.rotation	Image Manipulation
Stretch Height	If stretch mode is "Parameters", this will be the stretched height of the image<BR>If stretch mode is "% Bounds", this will be the percentage of the component's height.	int	.stretchHeight	Image Manipulation
Stretch Mode	Sets the stretch mode for this image.	int	.stretchMode	Image Manipulation
Stretch Width	If stretch mode is "Parameters", this will be the stretched width of the image<BR>If stretch mode is "% Bounds", this will be the percentage of the component's width.	int	.stretchWidth	Image Manipulation
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Swap From	If the Color Swap Filter is on, this color will be changed to the Swap To color.	Color	.swapFromColor	Image Manipulation
Swap Threshold	Threshold (0-255) for the swap from color matching. 0 is no tolerance, 255 is max tolerance.	int	.swapThreshold	Image Manipulation
Swap To	If the Color Swap Filter is on, the Swap From color will be changed to this color. See Color Selector.	Color	.swapToColor	Image Manipulation
Tint Color	If the Tint Filter is on, this is the color of the tint. See Color Selector.	Color	.tintColor	Image Manipulation
Tint Filter	Tint the entire image a color (works best with greyscale images).	boolean	.useTint	Image Manipulation

Use Cache	If false, this image will bypass the client image cache and load the image directly from the source.	boolean	.useCache	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

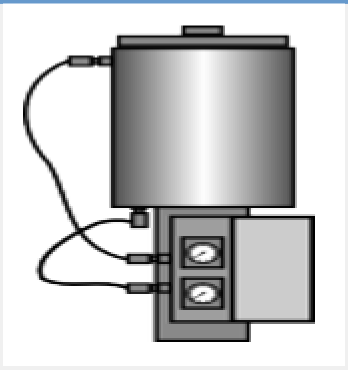
Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

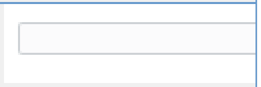
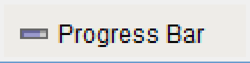
- Vision Component Customizers
- Style Customizer

Examples



Property Name	Value
Image Path	Builtin/Valve/Valve 29.png

Vision - Progress Bar

General

Component Palette Icon:
 Progress Bar

Description
Visually indicates the progress of some task. Can be used to display any value that has an upper and lower bound.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Direction	Determines the direction of progress for this progress bar.	int	.direction	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See Color Selector.	Color	.foreground	Appearance
Horizontal?	If true, the progress bar will display horizontally, else it will display vertically. Manually resize the progress bar to display vertically.	boolean	.horizontal	Appearance
Indeterminate?	When true, the progress bar displays animation indicating that something is happening, but it will take an indeterminate amount of time	boolean	.indeterminate	Behavior
Maximum	The maximum value that this progress bar will reach.	int	.maximum	Data
Minimum	The minimum value that this progress bar will reach.	int	.minimum	Data
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Show Percentage?	If true, the progress bar will display its percentage.	boolean	.stringPainted	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Value	The current state of the Progress Bar.	int	.value	Data
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

⚠ This event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

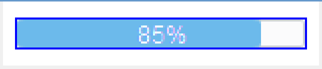
.source	The component that fired this event
. newValue	The new value for this property.
. oldValue	The value that this property was before it changed. Not all components include an accurate oldValue in their events.
. property Name	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers
- Style Customizer

Examples

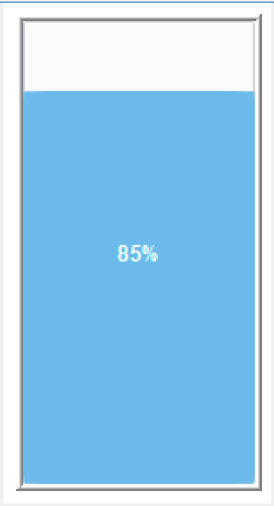
Horizontal Blue Progress Bar



Property Name	Value
Border	Line Border
Value	85
Foreground Color	0,0,255
Horizontal?	True
Show Percentage?	True

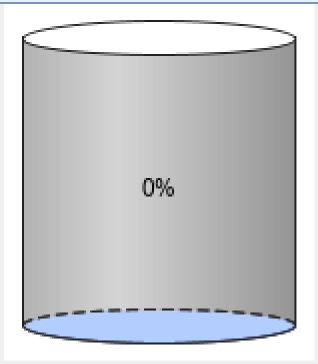
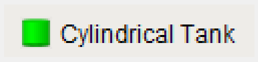
Gallery

Wide Vertical Blue Progress Bar



Property Name	Value
Border	Bevel (Double)
Value	85
Foreground Color	0,0,255
Horizontal?	False
Show Percentage?	True

Vision - Cylindrical Tank

General

Component Palette Icon:


Cylindrical Tank

Description
A component that looks like a 3D cylindrical tank, with some liquid inside. The liquid rises and falls as the Value property changes.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Capacity	Total capacity of tank.	double	.capacity	Data
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Font	Font of text on this component.	Font	.font	Appearance
Font Color	The color of the value and/or percentage labels. See Color Selector.	Color	.fontColor	Appearance
Foreground Color	The foreground color of the component. See Color Selector.	Color	.foreground	Appearance
Liquid Color	Color of the filled tank section. See Color Selector.	Color	.liquidColor	Appearance

Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Percent Format	Format string used for the percentage.	String	.percentFormat	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Rotation	The angle of rotation in degrees.	int	.rotation	Appearance
Show Percentage	Show percentage of tank filled?	boolean	.showPercent	Appearance
Show Value	Show numeric value, capacity, and units?	boolean	.showValue	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Tank Color	Color of the non-filled tank section. See Color Selector.	Color	.tankColor	Appearance
Units	Units of measure for tank contents.	String	.units	Appearance
Value	Numeric value of tank's level.	double	.value	Data
Value Format	Format string used for the value.	String	.valueFormat	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

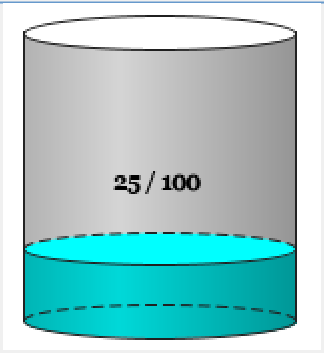
.source	The component that fired this event
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
. property Name	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers
- Style Customizer

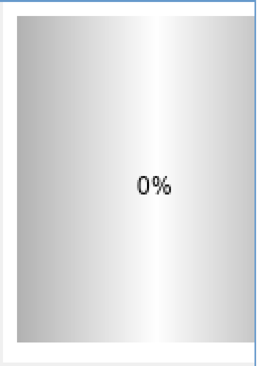
Examples

Cylindrical Tank



Property Name	Value
Value	25
Font	Georgia, Bold 12
Liquid Color	0,217,217
Show Value	True
Show Percentage	False

Vision - Level Indicator

General

Component Palette Icon:
Level Indicator

Description
A component that can be filled up with water. Usually used behind a symbol factor object that has a cutout in it.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The color of the background. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Capacity	Total capacity of tank.	double	.capacity	Data
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Filled Color	Set the color of filled portion. See Color Selector.	Color	.foreground	Appearance
Font	Font of text on this component.	Font	.font	Appearance
Font Color	The foreground color of the component. See Color Selector.	Color	.fontColor	Appearance
Gradient	Indicates whether the level will be drawn as a 3D gradient.	boolean	.gradient	Appearance
Liquid Waves	Indicate whether liquid waves are drawn.	boolean	.waves	Appearance

Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Orientation	Determines which direction the level "grows" for an increase in value.	int	.orientation	Appearance
Percent Format	Format string used for the percentage.	String	.percentFormat	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Show Percentage	Indicates whether the percentage of tank filled is displayed.	boolean	.showPercent	Appearance
Show Value	Indicates whether the numeric value, capacity, and units are displayed.	boolean	.showValue	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Units	Units of measure for tank contents.	String	.units	Appearance
Value	Numeric value of tank's level.	double	.value	Data
Value Format	Format string used for the value.	String	.valueFormat	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Wave Height	The height of each wave.	int	.waveHeight	Appearance
Wave Length	The length of each wave.	int	.waveLength	Appearance
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Data

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

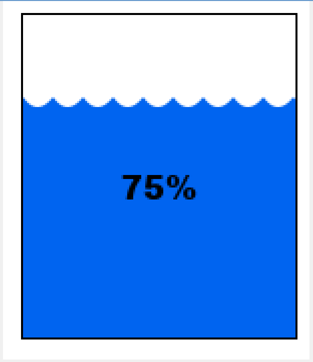
.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers
- Style Customizer

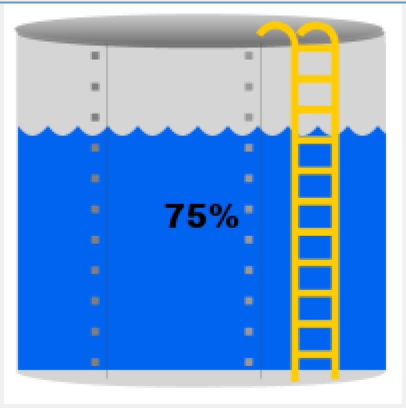
Examples

Level Indicator



Property Name	Value
Border	Line Border
Value	75
Units	Gallons
Show Value	True
Gradient	False
Filled Color	0,100,240
Font	Arial Black, Plain, 16
Wave Height	10
Wave Length	15

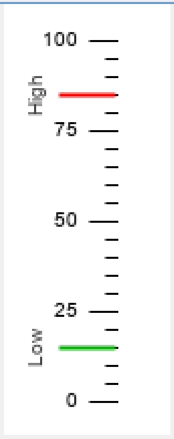
Level Indicator



Created using Symbol Factory Tanks > Tank with Rivets and Ladder. Then ungrouped twice. Fill paint set to 0,100,240.

Property Name	Value
Border	Line Border
Value	75
Units	Gallons
Show Value	True
Gradient	False
Filled Color	0,100,240
Background Color	250,250,251
Font	Arial Black, Plain, 16
Wave Height	10
Wave Length	15

Vision - Linear Scale

General

Component Palette Icon:
Linear Scale

Description
The Linear Scale component has two main purposes. The first is to display a series of tick marks and labels that visually represent a linear range between a minimum value and a maximum value. The second purpose is to display indicators that represent a value or range of values, correctly positioned on the linear scale. To configure the indicators, use the Linear Scale Customizer which is described below. To configure the tick marks, use the Linear Scale's various properties in the Property Editor that determine the minimum value, maximum value, and the various tick mark spans. There is no tall/wide option for this component. This is based on the width/height of the component. A tall Linear Scale has tick marks on the left or right, and a wide component has tick marks on the top or bottom.

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Fine Tick Color	The line color for fine ticks. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.fineTickColor	Appearance
Fine Tick Length	The line length for fine ticks, in pixels.	double	.fineTickLength	Appearance

Fine Tick Span	The span length for fine ticks. Should be a factor of the major and minor tick spans. Use zero to disable fine ticks.	double	.fineTicksSpan	Data
Fine Tick Thickness	The line thickness for fine ticks, in pixels.	float	.fineTicksStroke	Appearance
Indicators	This dataset stores the indicators (if any) for the scale.	Dataset	.indicators	Data
Label Angle	Changes the angle that the labels are drawn.	int	.labelAngle	Appearance
Label Color	The color used for drawing tick labels. See Color Selector.	Color	.majorTickLabelColor	Appearance
Label Font	The font used for drawing tick labels. See Color Selector.	Font	.majorTickFont	Appearance
Label Format	The label format string. Examples: "%.1f" will render numbers like "15.0", "%.0f" will render numbers like "15". Using the empty string "" will disable the labels.	String	.majorTickLabelFormat	Appearance
Major Tick Color	The line color for major ticks. See Color Selector.	Color	.majorTickColor	Appearance
Major Tick Length	The line length for major ticks, in pixels.	double	.majorTickLength	Appearance
Major Tick Span	The span length for major ticks. Should be a multiple of the minor and fine tick spans.	double	.majorTickSpan	Data
Major Tick Thickness	The line thickness for major ticks, in pixels.	float	.majorTickStroke	Appearance
Margin	The margin to leave blank as a percentage of the total height or width of the scale.	double	.margin	Appearance
Max Value	The upper bound of the scale.	double	.maxValue	Data
Min Value	The lower bound of the scale.	double	.minValue	Data
Minor Tick Color	The line color for minor ticks. See Color Selector.	Color	.minorTickColor	Appearance
Minor Tick Length	The line length for minor ticks, in pixels.	double	.minorTickLength	Appearance
Minor Tick Span	The span length for minor ticks. Should be a factor of the major tick span and a multiple of the fine tick spans. Use zero to disable minor ticks.	double	.minorTickSpan	Data
Minor Tick Thickness	The line thickness for minor ticks, in pixels.	float	.minorTickStroke	Appearance
Mirror	Mirror the scale so it paints against the opposite edge.	boolean	.mirror	Appearance

Mouse over Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Reverse Range	Reverse the scale so that values go from high to low instead of low to high.	boolean	.reverseRange	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

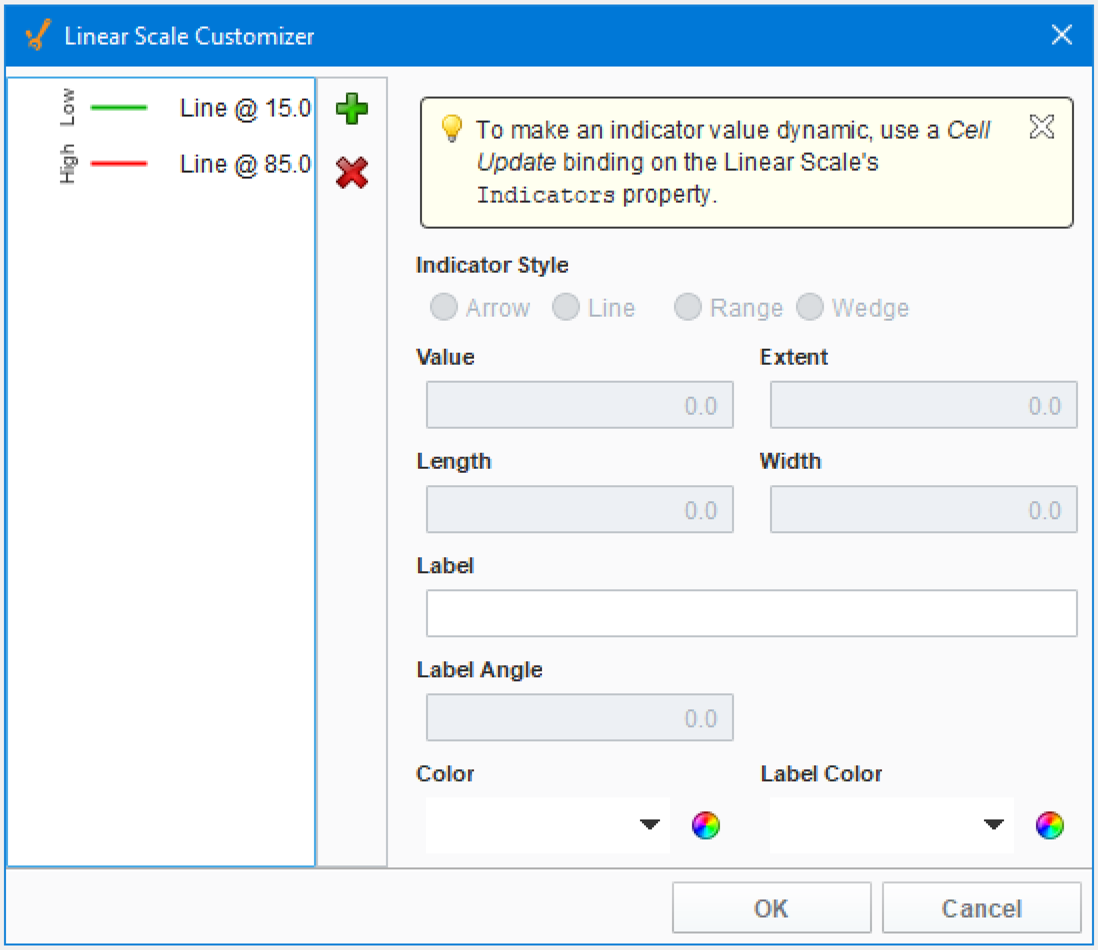
Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

The Linear Scale component has a special customizer called the Linear Scale Customizer. The customizer is where you configure the indicators that visually represent how your data is displayed on the scale. You can choose from several indicator styles: Arrow, Line, Range, and Wedge. There are a number of properties available to customize the appearance of your data on the Linear Scale. Not all Linear Scale Customizer properties are available with all indicator styles. The property will be grayed out if it is not available for that particular indicator. Use the preview window to validate the style you want to use for your data.

To make your indicator values dynamic, you can use a **Cell Update** binding on the **Indicators** property of this component.



To make an indicator value dynamic, use a *Cell Update* binding on the Linear Scale's `Indicators` property.

Linear Scale Customizer - Property Descriptions

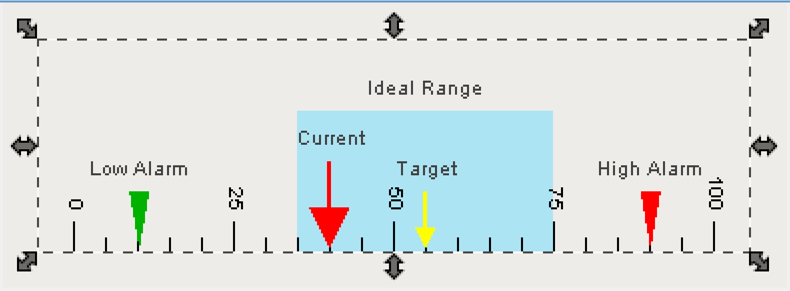
Property	Description
Indicator Style	There are four indicator styles to choose from: Arrow, Line, Range, and Wedge. • **Arrow**: A line with an arrow head at the given value • **Line**: A basic flat line at the given value • **Range**: a rectangle displayed with the given value at the bottom and a height equal to the Extent • **Wedge**: a wedge shape centered on the given value and a height equal to the Extent
Value	The position of the indicator.
Extent	Overall thickness of the indicator. Not valid for a Line style.
Length	The number of pixels to draw the indicator starting at the component edge.
Width	Thickness of the line in the indicator. Only valid for Arrow and Line styles.
Label	Name displayed next to the indicator.
Label Angle	The angle of the label specified in degrees.
Color	Color of the indicator.
Label Color	Color of the indicator Label.

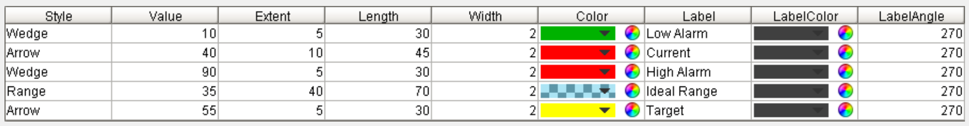
- Vision Component Customizers

Examples

In this example, the Linear Scale displays indicators for high and low levels. A **Cell Update** Binding was used on an Arrow indicator to make the "Current" level value dynamic.

Vertical Scale with Blue Indicators



Property Name	Value
Indicators	

Style	Value	Extent	Length	Width	Color	Label	LabelColor	LabelAngle
Wedge	10	5	30	2	🟩	Low Alarm	⬛	270
Arrow	40	10	45	2	🟥	Current	⬛	270
Wedge	90	5	30	2	🟥	High Alarm	⬛	270
Range	35	40	70	2	▨	Ideal Range	⬛	270
Arrow	55	5	30	2	🟨	Target	⬛	270

Vision - Barcode

General
Component Palette Icon:
Barcode

Description
The barcode component displays some text as a barcode. The supported formats are: • Code 128 • Code 39 • Extended Code 39 • Codabar • Interleaved Code 25 • MSI • EAN-13 • EAN-8 • Aztec* • Data Matrix* • PDF-417* • QR Code* • UPC-A* * Introduced in Ignition 7.8.0

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Barcode Background	The background color of the actual barcode. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.barcodeBackground	Appearance
Barcode Format	The barcode format to display.	int	.barcodeType	Data
Barcode Height	The height of the barcode.	int	.barcodeHeight	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Check Digit	Include Check Digit?	boolean	.checkDigit	Data
Code	The code string that is converted into a barcode to display.	String	.code	Data
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.foreground	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Narrowest Bar Width	The width (in pixels) of the narrowest bar.	int	.narrowestBarWidth	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Rotation	The angle of rotation in degrees.	int	.angleDegrees	Appearance
QRCode Error Correction Level	If you're creating a QR code, the QR code error correction level to use.	int	.qrEcLevel	Data
QRCode Version	If you're creating a QR code, the QR code version to use.	int	.qrCodeVersion	Data
Show Text?	If true, the code is displayed in human-readable text beneath the barcode.	boolean	.showText	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

This component does not have any custom properties.

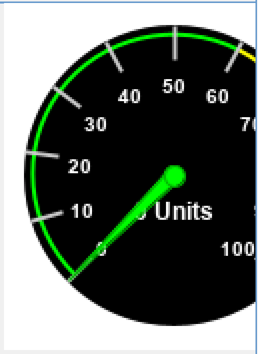
Examples

Barcode

123456789

Property Name	Value
Code	123456789
Barcode Format	Extended Code 39 (narrow)
Show Text?	True

Vision - Meter

General

Component Palette Icon:
Meter

Description
A meter display shows a value on a needle-gauge. The gauge's range can be broken up into five intervals. The intervals can have their own edge and background colors. How the meter looks is affected by its appearance properties. You can modify colors, thicknesses, start and extend angles, needle size, etc to get the meter that you want. For example, the meter on the far right of the example has a Meter Angle Extent of 90°, a Meter Angle of 45°, a reversed range, and two intervals.

Properties

Name	Description	Property Type	Scripting	Category
Arc Width	The width of the colored interval arcs.	float	.arcWidth	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Dial Background	The background color of the dial face. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.dialBackground	Appearance
Dial Shape	The shape of the dial. This property determines how the dial face looks in the area not covered by the meter angle extent.	int	.dialType	Appearance
Interval 1 Background	The color to fill the wedge of this interval. See Color Selector.	Color	.interval1Background	Intervals

Interval 1 High	The upper bound of this interval.	double	.interval1High	Intervals
Interval 1 Low	The lower bound of this interval.	double	.interval1Low	Intervals
Interval 1 Outline	The color to paint the arc of this interval. See Color Selector.	Color	.interval1Outline	Intervals
Interval 2 Background	The color to fill the wedge of this interval. See Color Selector.	Color	.interval2Background	Intervals
Interval 2 High	The upper bound of this interval.	double	.interval2High	Intervals
Interval 2 Low	The lower bound of this interval.	double	.interval2Low	Intervals
Interval 2 Outline	The color to paint the arc of this interval. See Color Selector.	Color	.interval2Outline	Intervals
Interval 3 Background	The color to fill the wedge of this interval. See Color Selector.	Color	.interval3Background	Intervals
Interval 3 High	The upper bound of this interval.	double	.interval3High	Intervals
Interval 3 Low	The lower bound of this interval.	double	.interval3Low	Intervals
Interval 3 Outline	The color to paint the arc of this interval. See Color Selector.	Color	.interval3Outline	Intervals
Interval 4 Background	The color to fill the wedge of this interval. See Color Selector.	Color	.interval4Background	Intervals
Interval 4 High	The upper bound of this interval.	double	.interval4High	Intervals
Interval 4 Low	The lower bound of this interval.	double	.interval4Low	Intervals
Interval 4 Outline	The color to paint the arc of this interval. See Color Selector.	Color	.interval4Outline	Intervals
Interval 5 Background	The color to fill the wedge of this interval. See Color Selector.	Color	.interval5Background	Intervals
Interval 5 High	The upper bound of this interval.	double	.interval5High	Intervals
Interval 5 Low	The lower bound of this interval.	double	.interval5Low	Intervals

Interval 5 Outline	The color to paint the arc of this interval. See Color Selector.	Color	.interval5Outline	Intervals
Meter Angle	The angle in degrees of the centerpoint of the meter (90 is straight up).	int	.meterAngle	Appearance
Meter Angle Extent	The extent, in degrees, of the entire meter.	int	.meterAngleExtent	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Needle Color	The color of the meter's needle. See Color Selector.	Color	.needleColor	Appearance
Needle Size	The size of the base of the needle.	float	.needleSize	Appearance
Needle Stroke Color	The color of the needle's stroke. See Color Selector.	Color	.needleStrokeColor	Appearance
Needle Stroke Size	The size of the needle's stroke.	float	.needleStrokeSize	Appearance
Overall High Bound	The high bound for the whole meter.	double	.overallHigh	Data
Overall Low Bound	The lower bound for the whole meter.	double	.overallLow	Data
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Reverse Range?	If true, the meter will consider right to left needle movement as positive.	boolean	.reverseRange	Data
Show Tick Labels?	If true, value will be shown on interval-boundary ticks.	boolean	.ticks	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Tick Color	The color of tick marks.	Color	.tickColor	Appearance
Tick Format	The number format to use for the tick labels.	String	.tickLabelFormat	Appearance
Tick Label Color	The color of the tick labels. See Color Selector.	Color	.tickLabelColor	Appearance
Tick Label Font	The font to use for the tick labels.	Font	.labelFont	Appearance
Tick Size	The distance between ticks.	double	.tickSize	Appearance
Units	A string to describe the units for the current value label.	String	.units	Appearance

Value	The value to display in this meter. The needle and current value label will change to reflect this.	double	.value	Data
Value Color	The color of the meter's current value label. See Color Selector.	Color	.valueColor	Appearance
Value Format	The number format to use for the value label.	String	.valueLabelFormat	Appearance
Value Label Font	The font to use for the current value label.	Font	.valueFont	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

The following feature is new in Ignition version **8.0.16**
Click here to check out the other new features

- Description

  Provides an opportunity to perform further configuration via scripting.

- Parameters

  Component self- A reference to the component that is invoking this function.

  JFreeChart chart- A JFreeChart object. Refer to the JFreeChart documentation for API details.

- Return

  Nothing

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

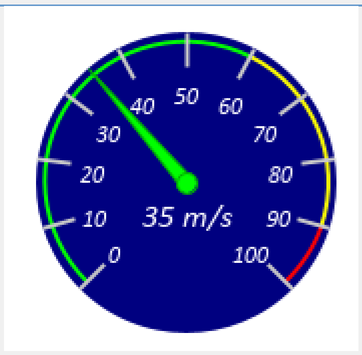
.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers
- Style Customizer

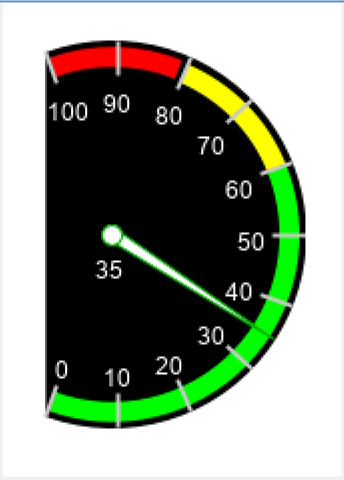
Examples

Updated fonts



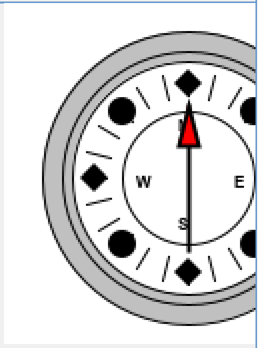
Property Name	Value
Dial Background	0,0,128
Value	35
Unit	m/s
Value Label Font	Caibri, Italic, 16
Tick Label Font	Caibri, Italic, 12

Chord Meter with modified value intervals



Property Name	Value
Value	35
Reverse Range?	True
Units	'None'
Arc Width	10
Meter Angle Extent	220
Meter Angle	0
Dial Shape	Chord
Interval 1 Low	40
Interval 2 High	60
Interval 2 Low	0
Interval 3 High	80
Interval 3 Low	60
Interval 4 High	100
Interval 3 Low	81

Vision - Compass

General

Component Palette Icon:
Compass

Description
The compass is a component that displays up to three needles at once on a cardinal direction compass. This can be useful for plotting anything that has a cardinal direction, such as the wind direction. Each needle can be one of nine different styles. Use the "Disabled" style to turn off any needle.

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. The border is unaffected by rotation.	Border	.border	Common
Center Color	The center color of the compass. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.centerColor	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Label Font	The font to use for the compass's labels.	Font	.labelFont	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data

Rose Color	The background color of the rose. See Color Selector.	Color	.roseColor	Appearance
Rose Highlight	The highlight color of the rose. See Color Selector.	Color	.roseHighlightColor	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Value 1	Value 1 for the compass.	double	.value1	Data
Value 1 Color	The main color for Value 1's needle. See Color Selector.	Color	.value1Color	Appearance
Value 1 Needle	The needle type for this value.	int	.value1Needle	Data
Value 1 Outline	The outline color for value 1's needle. See Color Selector.	Color	.value1OutlineColor	Appearance
Value 2	Value 2 for the compass.	double	.value2	Data
Value 2 Color	The main color for Value 2's needle. See Color Selector.	Color	.value2Color	Appearance
Value 2 Needle	The needle type for this value.	int	.value2Needle	Data
Value 2 Outline	The outline color for Value 2's needle. See Color Selector.	Color	.value2OutlineColor	Appearance
Value 3	Value 3 for the compass.	double	.value3	Data
Value 3 Color	The main color for Value 3's needle. See Color Selector.	Color	.value3Color	Appearance
Value 3 Needle	The needle type for this value.	int	.value3Needle	Data
Value 3 Outline	The outline color for Value 3's needle. See Color Selector.	Color	.value3OutlineColor	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

The following feature is new in Ignition version **8.0.16**
Click here to check out the other new features

The following feature is new in Ignition version **8.0.16**
Click here to check out the other new features

- Description

  Provides an opportunity to perform further configuration via scripting.

- Parameters

  Component self- A reference to the component that is invoking this function.

  JFreeChart chart- A JFreeChart object. Refer to the JFreeChart documentation for API details.

- Return

  Nothing

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

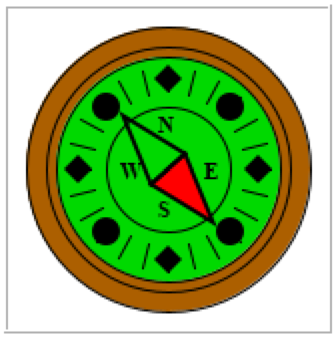
Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
. property Name	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

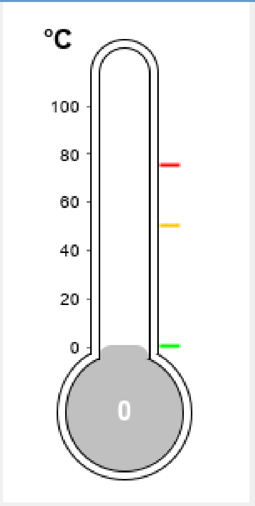
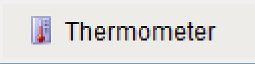
- Vision Component Customizers
- Style Customizer

Examples



Property Name	Value
Center Color	0,217,0
Rose Color	172,95,0
Label Font	Times New Roman, Bold, 14
Value 1	140
Value 1 Color	255,0,0
Value 1 Needle	Pointer

Vision - Thermometer

General

Component Palette Icon:


Description
This component displays a temperature value depicted as a level in a mercury thermometer. Three temperature intervals can optionally be defined with their own colors. The mercury will change color based on the range that it is in.

Properties

Name	Description	Property Type	Scripting	Category
Axis Label Color	The color of the meter's y-axis label. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.axisColor	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The borders is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Follow data in ranges	If true, the thermometer's Y axis will scale itself to zoom in on the current range.	boolean	.followDataInSubranges	Behavior
Interval 1 Color	The color of this interval. See Color Selector.	Color	.interval1Color	Intervals

Interval 1 High	The upper bound of this interval.	double	.interval1High	Intervals
Interval 1 Low	The lower bound of this interval.	double	.interval1Low	Intervals
Interval 2 Color	The color of this interval. See Color Selector.	Color	.interval2Color	Intervals
Interval 2 High	The upper bound of this interval.	double	.interval2High	Intervals
Interval 2 Low	The lower bound of this interval.	double	.interval2Low	Intervals
Interval 3 Color	The color of this interval. See Color Selector.	Color	.interval3Color	Intervals
Interval 3 High	The upper bound of this interval.	double	.interval3High	Intervals
Interval 3 Low	The lower bound of this interval.	double	.interval3Low	Intervals
Mercury Color	The default color of the mercury. See Color Selector.	Color	.mercuryColor	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Overall High Bound	The high bound for the whole thermometer	double	.overallHigh	Data
Overall Low Bound	The lower bound for the whole thermometer	double	.overallLow	Data
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Styles	Contains the component's styles	Dataset	.styles	Appearance
Thermometer Color	The color of the outline of the thermometer. See Color Selector.	Color	.thermometerColor	Appearance
Thermometer Width	The width of the lines used to draw the thermometer.	int	.strokeWidth	Appearance
Units	A string to describe the units for the current value label.	int	.units	Appearance
Use Range Color	Controls whether or not the mercury color changes based on the range it is in.	boolean	.useSubrangePaint	Appearance
Value	The value to display in this thermometer. The mercury level and value label will change to reflect this.	double	.value	Data
Value Color	The color of the meter's current value label. See Color Selector.	Color	.valueColor	Appearance

Value Label Font	The font to use for the current value label.	Font	.valueFont	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

The following feature is new in Ignition version **8.0.16**
Click here to check out the other new features

- Description

  Provides an opportunity to perform further configuration via scripting.

- Parameters

  Component self- A reference to the component that is invoking this function.

  JFreeChart chart- A JFreeChart object. Refer to the JFreeChart documentation for API details.

- Return

  Nothing

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

This event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

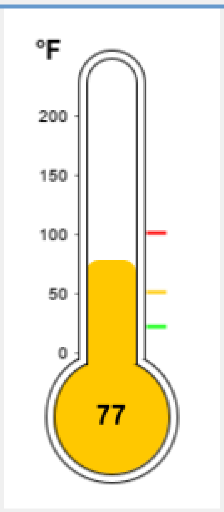
Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

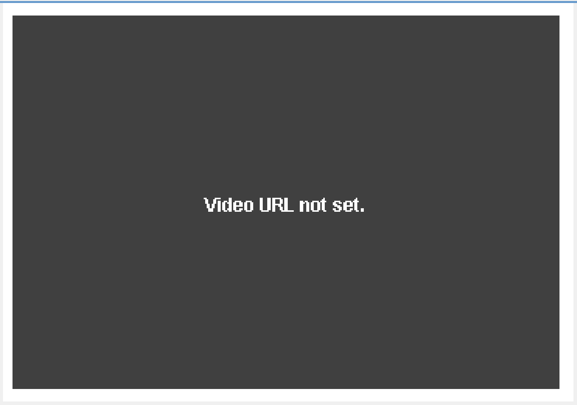
- Vision Component Customizers
- Style Customizer

Examples



Property Name	Value
Units	Fahrenheit
Value	192
Interval 1 High	59
Interval 1 Low	20
Interval 2 High	100
Interval 2 Low	50
Interval 3 High	187
Interval 3 Low	100
Mercury Color	255, 200,0
Use Range Color	True

Vision - IP Camera Viewer

General

Component Palette Icon:
IP Camera Viewer

Description
The IP camera viewing component displays a video stream from a network camera directly in one of your windows. This can be a very powerful tool for allowing operators to view remote or inaccessible locations. Cameras can provide positive feedback about the state and position of machinery, weather, and other factors.

This component is capable of displaying two types of video:

- MJPEG (a.k.a. Motion JPEG) is a streaming video protocol that compresses video frames using standard JPEG compression. Compression rates are quite good, requiring low network bandwidth utilization. Framerates depend greatly on the dimensions of the video, but typically range from 1-20 frames per second.
- JPEG stills is not a true video protocol, but is rather the practice of continually refreshing an image that a camera is constantly overwriting. Its simplicity means that many cameras support it (usually along with another protocol). Frame rates are typically lower than MJPEG because a new connection must be opened for each frame.

Most network cameras on the market support one, if not both of these protocols. Even better, if you have an existing CCTV camera system, video server devices are available that CCTV camera inputs and provide MJPEG streams the network.

Finding the URL for your network camera's video stream is usually the only challenge in connecting this component. Most, if not all, network cameras have an internal web server, allowing viewers to use web browsers to view their video stream. If you go to that webpage, and look at the HTML source of the page, you should be able to find the URL of the MJPEG or JPEG still stream.

High Resolution Streams

When viewing a feed from a High Resolution camera, the Camera Buffer Size property may need to be increased to contain all of the data from the stream.

Some examples:

Axis 2100 (MJPEG)

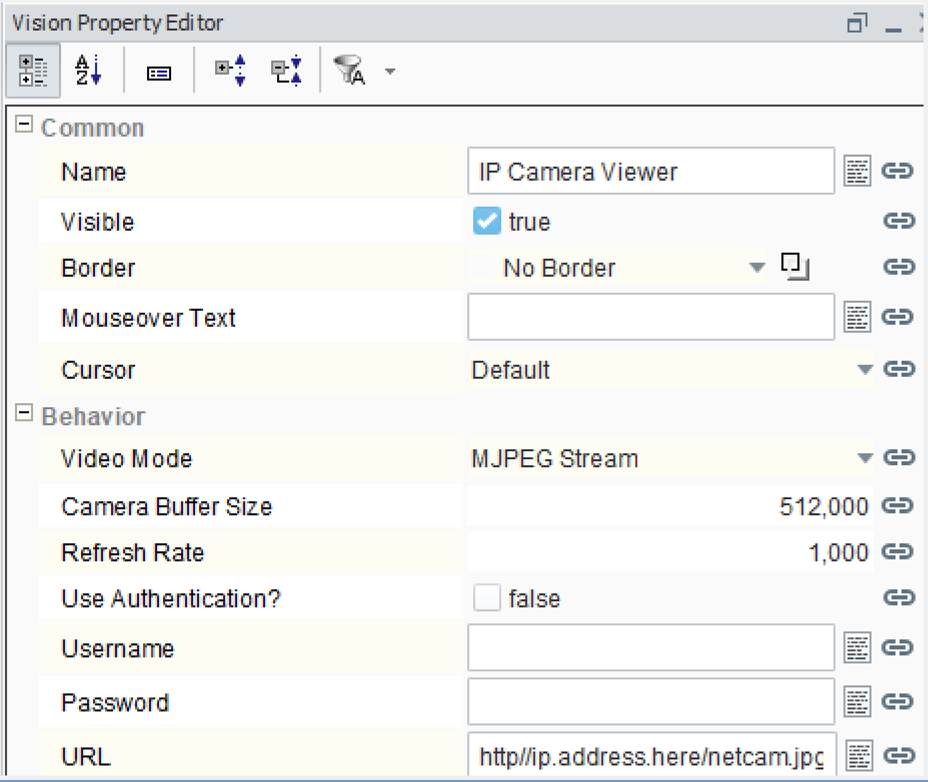
```
http://ip.address.here/axis-cgi/mjpg/video.cgi?resolution=640x480
```

Panasonic BL-C10A (MJPEG)

```
http://ip.address.here/nphMotionJpeg?Resolution=640x480&Quality=Standard
```

StarDot Netcam (JPEG stills)

```
http://ip.address.here/netcam.jpg
```


Vision Property Editor

Common

Name	IP Camera Viewer
Visible	true
Border	No Border
Mouseover Text	
Cursor	Default

Behavior

Video Mode	MJPEG Stream
Camera Buffer Size	512,000
Refresh Rate	1,000
Use Authentication?	false
Username	
Password	
URL	http//ip.address.here/netcam.jpg

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffecte by rotation.	Border	.border	Common
Camera Buffer Size	Set the size of the video buffer in bytes.	int	.cameraBufferSize	Behavior
Connection Retries	The number of times to attempt to connect to the stream.	int	.connectRetries	Behavior
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See Color Selector.	Color	.foreground	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Password	The password to authenticate with.	String	.password	Behavior
Refresh Rate	The rate (in ms) to poll the image if mode is 'JPEG Stills'.	int	.refreshRate	Behavior
Retry Delay	The delay (in ms) to wait between connection attempts.	int	.retryDelay	Behavior
Scale Mode	The scaling performance hint to use.	int	.scaleMode	Behavior
Scale Video	Scale the video to the size of the viewer component. Warning: CPU-intensive.	boolean	.scaleVideo	Behavior
Show Stats	If true, fps and Kbps statistical information will be overlaid on the video.	boolean	.showStats	Appearance
URL	The HTTP URL of the video stream to display.	String	.url	Behavior
Use Authentication?	If true, the URL connection will try to authenticate using the given username and password.	boolean	.useAuthentication	Behavior
User-Agent	If non-empty, the HTTP User-Agent to spoof.	String	.userAgent	Behavior
Username	The username to authenticate with.	String	.username	Behavior
Video Mode	Choose what type of video stream the URL points to.	int	.mode	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

This event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
. property Name	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

This component does not have any custom properties.

Examples

IP Camera Viewer

Property Name	Value
URL	http://trackfield.webcam.oregonstate.edu/mjpg/video.mjpg

Vision - Tables Palette

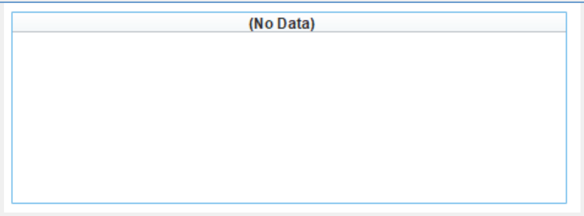
Table Components

The following components give you various types of tables for displaying values.

In This Section ...

Vision - Table

General



Component Palette Icon:

Table

INDUCTIVE UNIVERSITY

Table

Watch the Video

Description

The Table component is very powerful and easy to configure. It is very flexible, allowing you to easily display your tabular data in a variety of ways. Important features include:

- Column Sorting. Your users can easily sort the data by clicking on the column headers. The sorting is a 3-mode sort: Ascending, Descending, and "Natural", which uses the default order of the data.
- Mapped Row Coloring. Map the background color of each row to a particular column. This allows you to give powerful visual indication of different types of rows in you tables, such as differentiating between alarm states.
- Column Translation. Allow the table component to handle all code mapping, such as mapping 0 to "Off" and 1 to "On". No fancy SQL knowledge required.
- Images. Map values to images, allowing intuitive visual cues.
- Progress Bar Indication. Display numeric data as progress bars inside cells, providing fast visual reference for bounded amounts.
- Number and Date formatting. Format numbers and dates to your exact specification.
- Column Hiding. Hide columns from view that contain identifying data used by the row coloring or by other components.
- Printing. Print tables directly to multi-paged printouts.
- Editing. Columns can be made editable. Changes will be reflected in the underlying dataset, at which point they can be mapped back to a database.

Basic Usage

The basic usage of the Table is to use a SQL Query binding on its Data property to let the table display data from a database. Often this query will by dynamic or indirect. See the Property Binding section for more information.

Binding to Selected Data

It is common to want to bind other components to values in the selected row of the table. In order to do this safely, you need to write an expression binding that protects against the case when nothing is selected or there are no rows. An expression like this would bind a label to the selected row's value for a column named "ProductCode":

Expression Binding

```
if({Root Container.MyTable.selectedRow} = -1,
        "n/a", // this is the fail case
    {Root Container.MyTable.data}[{Root Container.MyTable.selectedRow},"ProductCode"] // this
selects from the dataset
)
```

If you're binding to an integer, date, or other non-String type value thats inside a dateset, you'll need to cast the value to the correct type to make the expression parser happy. This binding would cast the selected "Quantity" column to an integer:

Expression Binding

```
if({Root Container.MyTable.selectedRow} = -1,
    -1, // this is the fail case
    toInt({Root Container.MyTable.data}[{Root Container.MyTable.selectedRow},"Quantity"]) //
this selects from the dataset
)
```

Changing the Column Widths

To change a table's column's widths, simply switch into preview mode and use your mouse to resize the columns, and then switch back to design mode. To ensure that the changes to the column widths appear in the client, right-click on the table to open the table customizer and click OK without clicking anywhere else in the customizer. Clicking anywhere else in the customizer before clicking OK will reset the table column widths.

Editable Table

By setting any column to editable in the Table's customizer, the user will be able to double-click in the cell and edit the data. You can the respond to the resulting cellEdited event with an event handler and persist the data. See the Script Builders in Vision section for more information.

Exporting to HTML

You can export the table to an HTML file that retain's the table's formatting. To do this, use a script like this: (more about the table's export HTML function is here.)

Python Scripting

```
# Get a reference to the table
table = event.source.parent.getComponent("Table")

# Prompt user to save the exported file
table.exportHTML("MyTable.html", "My Table Header", 500)
```

Exporting to CSV

You can export the table's raw data to a CSV file. To do this, use a script like this: (more about the fpmi.db.exportCSV function is here.)

Python Scripting

```
# Get a reference to the table
table = event.source.parent.getComponent("Table")
system.dataset.exportCSV("mydata.csv", 1, table.data)
```

Printing

Printing a table is a snap! Simply use the table's built in print function like this: table = event.source.parent.getComponent ("Table") # Get a reference to the table table.print()

Python Scripting

```
table = event.source.parent.getComponent("Table") # Get a reference to the table table.print()
```

Properties

Name	Description	Property Type	Scripting	Category
Auto-Resize Mode	Determines how the table resizes the columns.	int	.autoResizeMode	Behavior
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Background Mode	This mode determines the color that this table's cell's backgrounds will be.	int	.backgroundColorMode	Appearance

Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Column Attributes Data	The dataset describing the column attributes.	Dataset	.columnAttributesData	Data
Column Selection Allowed	This flag is used in conjunction with the Row Selection Allowed flag to determine whether not whole-rows, whole-columns, or both (single-cells) are selectable.	boolean	.columnSelectionAllowed	Behavior
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Data	The data for this table.	Dataset	.data	Data
Edit Click Count	The number of clicks required to start editing a cell.	int	.clickCountToStart	Behavior
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See Color Selector.	Color	.foreground	Appearance
Grid Line Color	The color used to draw grid lines. See Color Selector.	Color	.gridColor	Appearance
Header Font	Font of the table's header text.	Font	.headerFont	Appearance
Header Foreground Color	The foreground color of the table's header. See Color Selector.	Color	.headerForeground	Appearance
Header Visible	Whether or not the table header is visible.	boolean	.headerVisible	Appearance
Initially Selected Row	The index of the row that should be selected by default when this table's<BR>data is filled in.	int	.initialRowSelection	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Odd Row Background	The color which odd rows will be colored if background mode is 'Alternating'. See Color Selector.	Color	.oddBackground	Appearance
Opaque	If false, backgrounds are not drawn. If true, backgrounds are drawn.	boolean	.opaque	Common
Properties Loading	The number of properties currently being loaded. (Read only. Usable in bindings and scripting.)	int	.propertiesLoading	Uncategorized

Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Deprecated
Resizing Allowed	Whether or not the user is allowed to resize table headers or not.	boolean	.resizingAllowed	Behavior
Row Height	The height of each row, in pixels.	int	.rowHeight	Appearance
Row Selection Allowed	This flag is used in conjunction with the Column Selection Allowed flag to determine whether not whole-rows, whole-columns, or both (single-cells) are selectable.	boolean	.rowSelectionAllowed	Behavior
Selected Column	The index of the first selected column, or -1 if none.	int	.selectedColumn	Data
Selected Row	The index of the first selected row, or -1 if none.	int	.selectedRow	Data
Selection Background	The background color of a selected cell. See Color Selector.	Color	.selectionBackground	Appearance
Selection Foreground	The foreground color of a selected cell. See Color Selector.	Color	.selectionForeground	Appearance
Selection Mode	This mode determines if only one row/cell/column can be selected at once, or single or multiple intervals.	int	.selectionMode	Behavior
Show Horizontal Grid Lines?	Shows horizontal grid lines.	boolean	.showHorizontalLines	Appearance
Show Vertical Grid Lines?	Shows vertical grid lines.	boolean	.showVerticalLines	Appearance
TestData	Toggle this property to fill in the table's data with random data.	boolean	.test	Misc
Touchscreen Mode	Controls when this table component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

- Description

  Adds a new row to the end of the table's dataset

- Parameters

  PySequence newRow - A sequence containing the values for the new row. The length of the sequence must match the number of columns in the table, and each value must be coercible into the correct datatype of the corresponding column.

- Return

  Nothing

- Description

  Deletes a row from the table's dataset.

- Parameters

  int rowIndex - The index of the row to delete.

- Return

  Nothing

- Description

  Prompts the user to save the table's data as a CSV file.

- Parameters

  String filename - A suggested filename for the user. For example: "table_data.csv"

  boolean showHeaders - If true, include headers in CSV file.

- Return

  String - The path to the saved file, or null if the operation was cancelled.

- Description

  Creates an HTML page as a string in memory. This can then be written to a file, a database, emailed, etc.

- Parameters

  String title - The title for the HTML page.

  int width - The width (in pixels) for the "table" element in the resulting html page.

- Return

  String - A string containing an HTML-formatted version of the table's data.

- Description

  Returns a list of ints that represent the underlying dataset's rows as they appear in the current sort order that the user is viewing.

- Parameters

  none

- Return

  List of Integers

- Description

  Returns the index of the currently selected column, or -1 if none is selected.
- Parameters

  none
- Return

  int
- Description

  Returns the number of columns that are currently selected.
- Parameters

  none
- Return

  int
- Description

  Returns the index of the currently selected row, or -1 if none is selected.
- Parameters

  none
- Return

  int
- Description

  Returns a list of the indexes of the selected row, or none if none is selected.
- Parameters

  none
- Return

  List, None
- Description

  Returns the number of rows that are currently selected.
- Parameters

  none
- Return

  int
- Description

  Tests whether the cell at the given row and column is currently selected or not.
- Parameters

  int row - The row to test.

  int column - The column to test.
- Return

  boolean

- Description

  Tests whether the given column is currently selected or not.

- Parameters

  int column- The column to test.

- Return

  boolean

- Description

  Tests whether the given row is currently selected or not.

- Parameters

  int row - The row to test.

- Return

  boolean

- Description

  This specialized print function will paginate the table onto multiple pages.This function accepts keyword-style invocation.

- Keyword Args

  boolean fitWidth- If true, the table's width will be stretched to fit across one page's width. Rows will still paginate normally. If false, the table will paginate columns onto extra pages. (default = true) [optio nal]

  string headerFormat- A string to use as the table's page header. The substring "{0}" will be replaced with the current page number. (default = None) [optional]

  string footerFormat- A string to use as the table's page footer. The substring "{0}" will be replaced with the current page number. (default = "Page {0}") [optional]

  boolean showDialog- Whether or not the print dialog should be shown to the user. Default is true. [opt ional]

  boolean landscape- Used to specify portrait (0) or landscape (1) mode. Default is portrait (0). [optiona l]

- Return

  boolean- True if the print job was successful.

- Description

  Used to set a column's header label to a new string at runtime.

- Parameters

  int column - The column index that will get a new headel label.

  String label - The new header label.

- Return

  nothing

- Description

    Sets the given range of columns to be selected. If index0==index1, it will select a single column.

- Parameters

    int index0 - the first index.

    int index1 - the second index.

- Return

    boolean - True if selection range is valid.

- Description

    Used to set a column's width at runtime.

- Parameters

    int column - The index of the column.

    int width - The width to set it at in pixels.

- Return

    nothing

- Description

    Sets the given range of rows to be selected. If index0==index1, it will select a single row.

- Parameters

    int index0 - The first index.

    int index1 - The second index.

- Return

    boolean - True if selection range is valid.

- Description

    Sets the given column to be the selected column.

- Parameters

    int column - Column to select.

- Return

    nothing

- Description

    Sets the given row to be the selected row.

- Parameters

    int row - Row to select.

- Return

    nothing

- Description

  Sets the value in the specified cell, altering the table's Data property. Will fire a propertyChange event for the "data" property, as well as a cellEdited event.

- Parameters

  int row - The index of the row to set the value at.

  int column - The index or name of the column to set a value at.

  PyObject value - The new value to use at the given row/column location.

- Return

  nothing

- Description

  Instructs the table to sort the data by the named column.

- Parameters

  String columnName - The name of the column.

  boolean asc - 1 means ascending, 0 means descending. (default = 1) [optional]

- Return

  nothing

- Description

  Instructs the table to clear any custom sort columns and display the data as it is sorted in the underlying dataset.

- Parameters

  nothing

- Return

  nothing

- Description

  Updates an entire row of the table's dataset.

- Parameters

  int rowIndex - The index of the row to update.

  PyDictionary changes - A sequence containing the updated values for the row. The length of the sequence must match the number of columns in the table, and each value must be coercible into the correct datatype of the corresponding column.

- Return

  nothing

Extension Functions

- Description

  Called for each cell, returns the appropriate background color. Do not block, sleep, or execute any I/O; called on painting thread.

- Parameters

  Component self - A reference to the component that is invoking this function.

  int row -The row index of the cell.

  int col -The column index of the cell.

  boolean isSelected - A boolean representing if the cell is currently selected.

  Object value -The value in the table's dataset at index [row, col].

  Color defaultColor -The color the table would have chosen if this function was not implemented.

- Return

  Color

- Description

  Called for each cell, returns the appropriate foreground (text) color. Do not block, sleep, or execute any I/O; called on painting thread.

- Parameters

  Component self - A reference to the component that is invoking this function.

  int row -The row index of the cell.

  int col -The column index of the cell.

  boolean isSelected - A boolean representing if the cell is currently selected.

  Object value -The value in the table's dataset at index [row, col].

  Color defaultColor - The color the table would have chosen if this function was not implemented.

- Return

  Color

- Description

  Called for each cell, returns a String which will be used as the text of the cell. Do not block, sleep or execute any I/O; called on the painting thread.

- Parameters

  Component self - A reference to the component that is invoking this function.

  int row-The row index of the cell.

  int col-The column index of the cell.

  boolean isSelected: A boolean representing if the cell is currently selected.

  Object value-The value in the table's dataset at index [row, col].

  String defaultText -The string the table would have chosen if this function was not implemented.

- Return

  String

Event Handlers

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event
. oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
. newValue	The new value that this property changed to.
.row	The row of the dataset this cell represents.
.column	The column of the dataset this cell represents.

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event.
. oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when a component that had the input focus lost it to another component.

.source	The component that fired this event
. oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

An integer that indicates whether the state was changed to "Selected" (on) or "Deselected" (off). Compare this to the event object's constants to determine what the new state is.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is released and the source component has the input focus. Works for all characters, including non-printable ones, such as SHIFT and F3.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is pressed and then released when source component has the input focus. Only works for characters that can be printed on the screen.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires *aft er* the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

The Table component has a Table customizer to manage column configurations and configure background color mapping. It allows you to customize how you want the table to look to users.

- Vision - Table Customizer
- Vision Component Customizers
- Understanding Component Customizers

Examples

Code Snippet

```
#The following would add a row to the table.
#Note that this function takes a list
#And that the property types of the list are the same as the table.

name = "Motor 1"
state = 2
amps = 35
list = [name, state, amps]
table = event.source.parent.getComponent('Table')
table.addRow(list)
```

Vision - Table Customizer

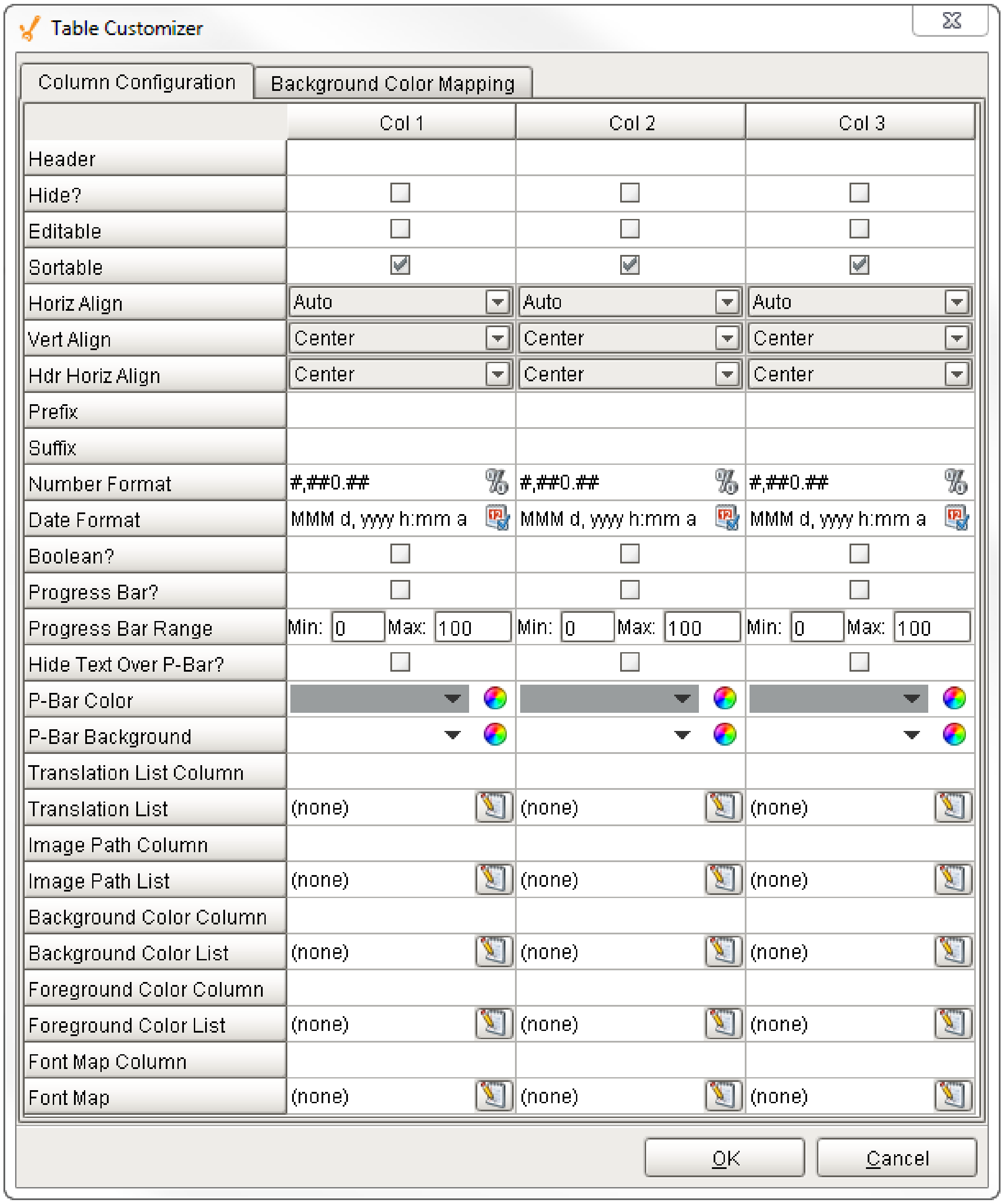


Table Customizer

Column Configuration | Background Color Mapping

	Col 1	Col 2	Col 3
Header			
Hide?	☐	☐	☐
Editable	☐	☐	☐
Sortable	☑	☑	☑
Horiz Align	Auto	Auto	Auto
Vert Align	Center	Center	Center
Hdr Horiz Align	Center	Center	Center
Prefix			
Suffix			
Number Format	#,##0.##	#,##0.##	#,##0.##
Date Format	MMM d, yyyy h:mm a	MMM d, yyyy h:mm a	MMM d, yyyy h:mm a
Boolean?	☐	☐	☐
Progress Bar?	☐	☐	☐
Progress Bar Range	Min: 0 Max: 100	Min: 0 Max: 100	Min: 0 Max: 100
Hide Text Over P-Bar?	☐	☐	☐
P-Bar Color	▨	▨	▨
P-Bar Background			
Translation List Column			
Translation List	(none)	(none)	(none)
Image Path Column			
Image Path List	(none)	(none)	(none)
Background Color Column			
Background Color List	(none)	(none)	(none)
Foreground Color Column			
Foreground Color List	(none)	(none)	(none)
Font Map Column			
Font Map	(none)	(none)	(none)

OK | Cancel

Description
The Table component is one of the most flexible and easy to configure components in Ignition. It has its own Table Customizer that allows you to make changes to tabular data and display it in a variety ways. The customizer not only lets you customize each column in the table, but together with its data properties and use of scripting and extension functions, it lets you configure how each cell in the table looks and behaves.

Customizers

The Table Customizer allows you to configure how you want the table to look to users. When you open the Table Customizer, you'll notice two tabs: Column Configuration and Background Color Mapping. The Column Configuration tab contains a number of column configuration properties that can be used to customize each column in the dataset to look a certain way. You can assign a header name, hide a column, make the column editable and sortable, align the text within the column, add a prefix by putting a "$" in front of a value, or suffix by adding a "%" at the end of a value, select a number and date format, turn the column into a progress bar, translate a number into a string or image or even into a background or foreground color. It's even possible to change the background, foreground, and font for the text in each particular column or cell.

In the Background Color Mapping tab, you can set the table's Background property to 'Mapped', and choose a column to govern the background color of each row. The column is specified in the Mapping Column dropdown selector. The column must be a numeric type. The number to color translation works with the contents of the mapping column rows to format the cells in accordance with the selected color.

TestData Property

If you want to test how the Table Customizer works in the Table, drag a Table on to your workspace, go to the Test Data property in the Property Editor, and check the 'false' checkbox. It will automatically fill the table with some test data so you get test out the Table Customizer

- Component Customizers
- Understanding Component Customizers

Table Customizer Properties

Column Configuration Tab

Property	Description
Header	Provide a custom name to the column header.
Hide	Hides the column.
Editable	Allows the editing of the cell pertaining to the column.
Sortable	Allows the user to sort the table according to the selected column.
Horiz Align	Aligns the contents of the column.
Vert Align	Aligns the contents of the column.
Hdr Horiz Align	Aligns the contents of the column.
Prefix	A custom text that proceeds the contents of each cell.
Suffix	A custom text that follows the contents of each cell.
Number Format	A format of the cell if the contents of the cell are number types.
Boolean	Changes the contents of the cell to reflect a 'check box' look and feel.
Progress Bar	A graphical bar is represented in the cell instead of a number.
Progress Bar Range	Sets the min and max range of the progress bar.
Hide Text Over P-Bar	Makes the value and text that controls the progress bar visible or invisible.
P-Bar Color	The color of the progress bar.
P-Bar Background	The color of the cell that has a progress bar.
Translation List Column	This works in conjunction with the Translation List. The key is provided by a named column resulting in the cells being translated according to the list that contains the key pairs.

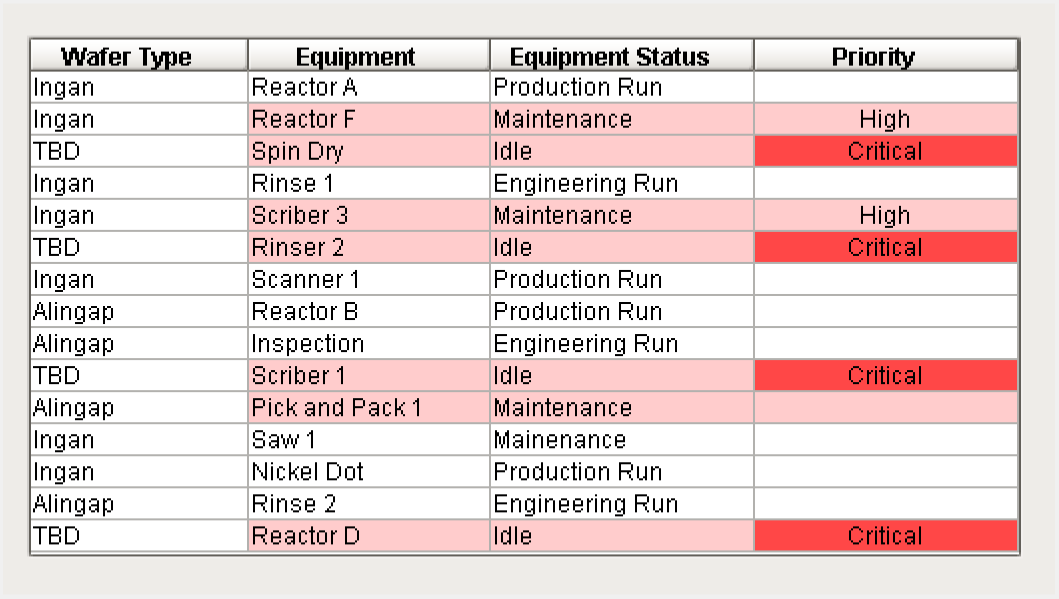
Translation List	Defines the key/Translation pairs and translates the contents of the cell accordingly.
Image Path Column	This works in conjunction with the Image Path List. The key is provided by a named column resulting in the cells being translated according to the list that contains the key pairs.
Image Path List	Defines the key/Translation pairs and translates the contents of the cell accordingly.
Background Color Column	This works in conjunction with the Background Color List. The key is provided by a named column resulting in the cells being translated according to the list that contains the key pairs.
Background Color List	Defines the key/Translation pairs and translates the contents of the cell accordingly.
Foreground Color Column	This works in conjunction with the Foreground Color List. The key is provided by a named column resulting in the cells being translated according to the list that contains the key pairs.
Foreground Color List	Defines the key/Translation pairs and translates the contents of the cell accordingly.
Font Map Column	This works in conjunction with the Foreground Color List. The key is provided by a named column resulting in the cells being translated according to the list that contains the key pairs.
Font Map	Defines the key/Translation pairs and translates the contents of the cell accordingly. An example of a font translation could look like this "Dialog, Bold, 12"
Color Mapping Tab	
Mapping Column	Select a column to govern the background color of each row.
Number to Color Translation	A numeric value (typically an integer) that drives the background and foreground color of a row. For every number or value, you can choose a different color.
Fallback Color	Default color that can be set when a value does is not defined.

Example

The table in this example uses several mappings:

- Col 4 changed a number into a string: translated a priority "1" to Critical, and priority "2" to High. It also change the background colors of the cells for both priorities.
- Col 3 changed the background colors for the equipment status's "Maintenance" and "Idle" to pale red.
- Col 2 change the background color of the equipment name to pale red for the equipment status's that were "Idle" and "Maintenance."

Table



Wafer Type	Equipment	Equipment Status	Priority
Ingan	Reactor A	Production Run	
Ingan	Reactor F	Maintenance	High
TBD	Spin Dry	Idle	Critical
Ingan	Rinse 1	Engineering Run	
Ingan	Scriber 3	Maintenance	High
TBD	Rinser 2	Idle	Critical
Ingan	Scanner 1	Production Run	
Alingap	Reactor B	Production Run	
Alingap	Inspection	Engineering Run	
TBD	Scriber 1	Idle	Critical
Alingap	Pick and Pack 1	Maintenance	
Ingan	Saw 1	Mainenance	
Ingan	Nickel Dot	Production Run	
Alingap	Rinse 2	Engineering Run	
TBD	Reactor D	Idle	Critical

Table Customizer

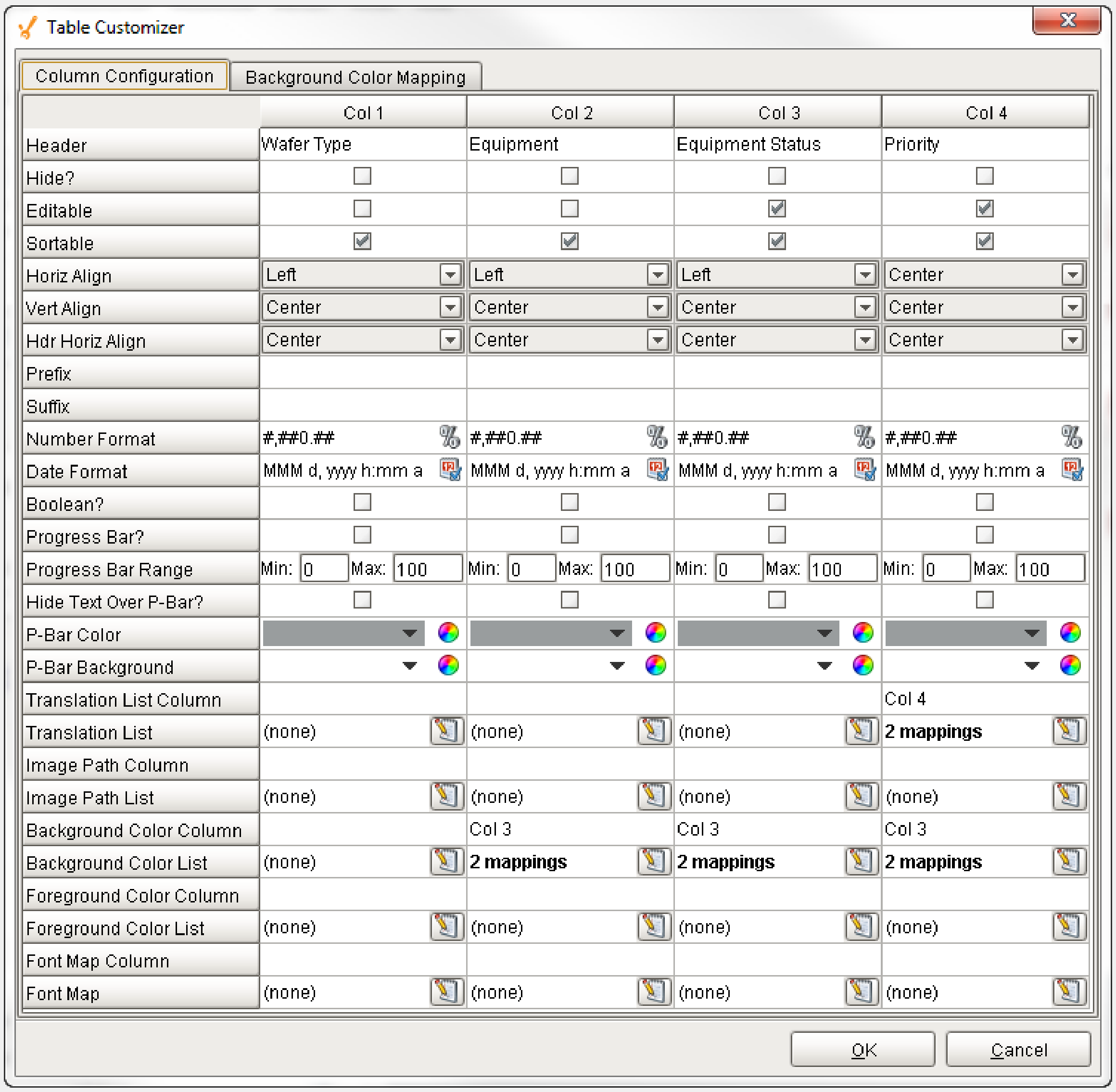


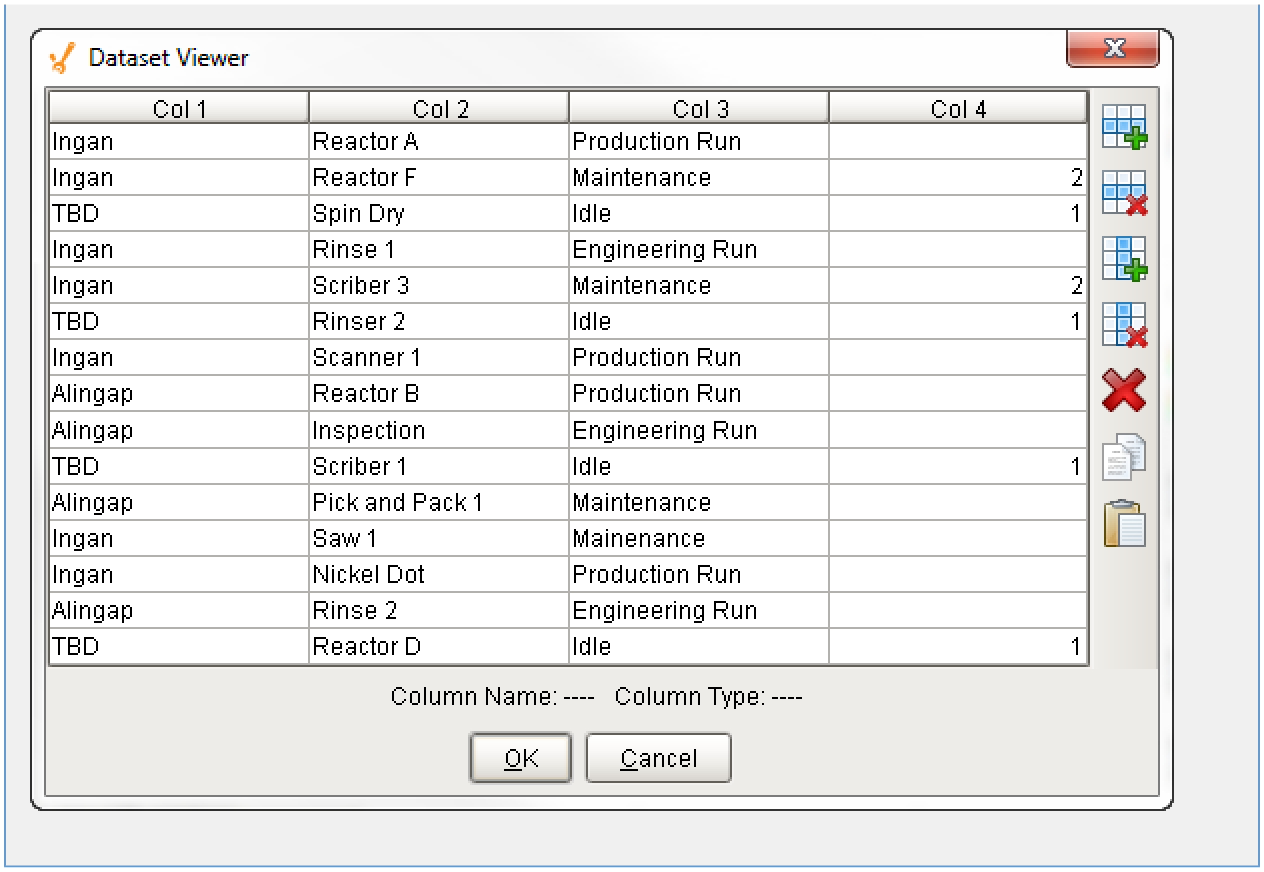
Table Customizer

Column Configuration | Background Color Mapping

	Col 1	Col 2	Col 3	Col 4
Header	Wafer Type	Equipment	Equipment Status	Priority
Hide?	☐	☐	☐	☐
Editable	☐	☐	☑	☑
Sortable	☑	☑	☑	☑
Horiz Align	Left	Left	Left	Center
Vert Align	Center	Center	Center	Center
Hdr Horiz Align	Center	Center	Center	Center
Prefix				
Suffix				
Number Format	#,##0.##	#,##0.##	#,##0.##	#,##0.##
Date Format	MMM d, yyyy h:mm a	MMM d, yyyy h:mm a	MMM d, yyyy h:mm a	MMM d, yyyy h:mm a
Boolean?	☐	☐	☐	☐
Progress Bar?	☐	☐	☐	☐
Progress Bar Range	Min: 0 Max: 100	Min: 0 Max: 100	Min: 0 Max: 100	Min: 0 Max: 100
Hide Text Over P-Bar?	☐	☐	☐	☐
P-Bar Color				
P-Bar Background				
Translation List Column				Col 4
Translation List	(none)	(none)	(none)	**2 mappings**
Image Path Column				
Image Path List	(none)	(none)	(none)	(none)
Background Color Column		Col 3	Col 3	Col 3
Background Color List	(none)	**2 mappings**	**2 mappings**	**2 mappings**
Foreground Color Column				
Foreground Color List	(none)	(none)	(none)	(none)
Font Map Column				
Font Map	(none)	(none)	(none)	(none)

OK | Cancel

Data Property Dataset

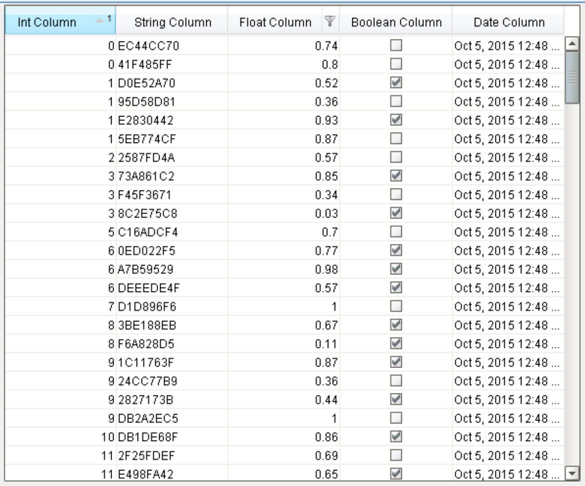


Col 1	Col 2	Col 3	Col 4
Ingan	Reactor A	Production Run	
Ingan	Reactor F	Maintenance	2
TBD	Spin Dry	Idle	1
Ingan	Rinse 1	Engineering Run	
Ingan	Scriber 3	Maintenance	2
TBD	Rinser 2	Idle	1
Ingan	Scanner 1	Production Run	
Alingap	Reactor B	Production Run	
Alingap	Inspection	Engineering Run	
TBD	Scriber 1	Idle	1
Alingap	Pick and Pack 1	Maintenance	
Ingan	Saw 1	Mainenance	
Ingan	Nickel Dot	Production Run	
Alingap	Rinse 2	Engineering Run	
TBD	Reactor D	Idle	1

Column Name: ---- Column Type: ----

Vision - Power Table

General



Int Column	String Column	Float Column	Boolean Column	Date Column
0	EC44CC70	0.74	☐	Oct 5, 2015 12:48 ...
0	41F485FF	0.8	☐	Oct 5, 2015 12:48 ...
1	D0E52A70	0.52	☑	Oct 5, 2015 12:48 ...
1	95D58D81	0.36	☐	Oct 5, 2015 12:48 ...
1	E2830442	0.93	☑	Oct 5, 2015 12:48 ...
1	5EB774CF	0.87	☐	Oct 5, 2015 12:48 ...
2	2587FD4A	0.57	☐	Oct 5, 2015 12:48 ...
3	73A861C2	0.85	☑	Oct 5, 2015 12:48 ...
3	F45F3671	0.34	☐	Oct 5, 2015 12:48 ...
3	8C2E75C8	0.03	☑	Oct 5, 2015 12:48 ...
5	C16ADCF4	0.7	☐	Oct 5, 2015 12:48 ...
6	0ED022F5	0.77	☑	Oct 5, 2015 12:48 ...
6	A7B59529	0.98	☑	Oct 5, 2015 12:48 ...
6	DEEEDE4F	0.57	☑	Oct 5, 2015 12:48 ...
7	D1D896F6	1	☐	Oct 5, 2015 12:48 ...
8	3BE188EB	0.67	☑	Oct 5, 2015 12:48 ...
8	F6A828D5	0.11	☑	Oct 5, 2015 12:48 ...
9	1C11763F	0.87	☑	Oct 5, 2015 12:48 ...
9	24CC77B9	0.36	☐	Oct 5, 2015 12:48 ...
9	2827173B	0.44	☑	Oct 5, 2015 12:48 ...
9	DB2A2EC5	1	☐	Oct 5, 2015 12:48 ...
10	DB1DE68F	0.86	☑	Oct 5, 2015 12:48 ...
11	2F25FDEF	0.69	☐	Oct 5, 2015 12:48 ...
11	E498FA42	0.65	☑	Oct 5, 2015 12:48 ...

Component Palette Icon:

Power Table

INDUCTIVE UNIVERSITY

Power Table

Watch the Video

Description

The power table is a more customizable version of the table component, and it comes with advanced features such as drag-and-drop rows, multi-column sorting, column filtering, and cell-spanning. Customization comes through extensive use of extension functions, which are available to configure how each cell of the table looks, how the headers look, etc.

Basic Usage

The basics are just like the classic table - you simply bind the table's "data" property to your data, most often by using a SQL query binding. Note that many of the options built into the classic table have been moved to extension functions in the power table.

Power Table Features

- **Multi-column sorting.** To sort multiple columns, select the header of the first column, hold down the Control key, then select the header of the next column. Click on the header again to reverse the sort order, and click a third time to remove sorting on the column.
- **Column filtering.** Columns can be temporarily hidden from view using column filtering. Right-click on the header of the table, and uncheck columns that you would like to hide. You can disable this feature by disabling the Column Chooser Menu property on the table.
- **Column reordering.** You can switch the locations of columns on the table using column reordering. Drag the header of the column that you would like to move to a new location on the table. You can disable this feature by disabling the Columns Re-Orderable property on the table.
- **Cell spanning.** A cell can be spanned across multiple columns and rows. Keep in mind that you must explicitly define the locations of cells that must be spanned. This means that if you would like to use cell spanning, any other table features that change how the table is displayed will be disabled automatically (such as sorting, column filtering and column reordering). Click on the Cell Span Data dataset to configure spanning. Within the dataset, add a row for each new span. The "row" column controls the row in the table where the span will start. The "column" column controls the column where the span will start. The "width" column controls how many columns the span will cover. The "height" column controls how many rows the span will cover. Adding a row where "row=4, column=1, width=2, height=3" results in a span starting on the fifth row of the table and the second column (using 0-based indexing). The span will cover the second and third columns in the row and will also cover two rows below the fifth row, as shown below.
- **Drag and Drop.** This feature allows you to drag rows from one power table to another power table. In order to perform drag and drop, you must implement the onRowsDropped() extension function on the destination table. This is so that you can adapt the data from one table to the other within the function. You must also enable the Row Dragging Enabled property on both tables.
- **Row Copying.** This feature allows you to select rows and copy them to the clipboard using the standard keyboard shortcut Ctrl + C. These can then be pasted anywhere, even outside of Ignition.

⚠ Even if a column is set to be editable, the edit must be handled by the onCellEdited extension function. If that extension function is not enabled and properly set up, the cell will revert back to its previous value.

Properties

Name	Description	Property Type	Scripting	Category
Auto Row Height	Enables automatic resizing of row height.	boolean	.rowResizeEnabled	Behavior
Auto-Resize Mode	Determines how the table resizes the columns.	int	.autoResizeMode	Behavior
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Cell Span Data	This dataset holds information about how cells in the table span multiple rows and/or columns. Incompatible with column sorting and re-ordering.	Dataset	.cellSpanData	Data
Column Attributes Data	The dataset describing the column attributes. Note: the data in this property doesn't get initialized until the customizer is opened and the OK button is pressed.	Dataset	.columnAttributesData	Appearance

Column Chooser Menu	Enables a right-click popup menu on the column headers with options to show and hide columns.	boolean	.headerColumnChooserMenus	Behavior
Column Resize Menu	Enables a right-click popup menu on the column headers with resizing options.	boolean	.headerResizeMenus	Behavior
Column Selection Allowed	This flag is used in conjunction with the Row Selection Allowed flag to determine whether not whole-rows, whole-columns, or both (single-cells) are selectable.	boolean	.columnSelectionAllowed	Behavior
Column Sizing	Represents column sizing and position to preserve user-selected ordering.	String	.defaultColumnView	Appearance
Columns Re-Orderable	Enables the re-ordering of columns by dragging the column headers.	boolean	.columnReorderingAllowed	Behavior
Columns Resizable	Enables the resizing of columns by dragging the margins of the column headers.	boolean	.columnResizingAllowed	Behavior
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Data	The data for this table.	Dataset	.data	Data
Edit Click Count	The number of clicks required to start editing a cell.	int	.clickCountToStart	Behavior
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See Color Selector.	Color	.foreground	Appearance
Grid Line Color	The color used to draw grid lines. See Color Selector.	Color	.gridColor	Appearance
Header Font	Font of the table's header text.	Font	.headerFont	Appearance
Header Visible	Allows for hiding of the table's header.	boolean	.headerVisible	Appearance
Inter Cell Spacing	The space (in pixels) between the cells.	Dimension	.interCellSpacing	Appearance
Name	The name of this component.	String	.name	Common
Non-Contiguous Selection	Enables totally non-contiguous selection in the table.	boolean	.nonContiguousCellSelection	Behavior

Properties Loading	The number of properties currently being loaded. (Read only. Usable in bindings and scripting.)	int	.propertiesLoading	Uncategorized
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Row Dragging Enabled	Enables drag-and-drop re-ordering for table rows. Implementing the 'onRowsDropped' extension function is also required to have functional drag-and-drop.	boolean	.rowDragEnabled	Behavior
Row Height	If row resizing is disabled, this will set the height of all rows.	int	.rowHeight	Behavior
Row Selection Allowed	This flag is used in conjunction with the Column Selection Allowed flag to determine whether not whole-rows, whole-columns, or both (single-cells) are selectable.	boolean	.rowSelectionAllowed	Behavior
Selected Column	The index of the first selected column, or -1 if none.	int	.selectedColumn	Data
Selected Row	The index of the first selected row, or -1 if none.	int	.selectedRow	Data
Selection Background	The default background color of selected cells. See Color Selector.	Color	.selectionBackground	Appearance
Selection Foreground	The default foreground color of selected cells. See Color Selector.	Color	.selectionForeground	Appearance
Selection Mode	This mode determines if only one row/cell/column can be selected at once, or single or multiple intervals.	int	.selectionMode	Behavior
Show Horizontal Grid Lines?	Shows horizontal grid lines.	boolean	.showHorizontalLines	Appearance
Show Vertical Grid Lines?	Shows vertical grid lines.	boolean	.showVerticalLines	Appearance
Sorting Enabled	Enables automatic multi-column sorting by clicking and CTRL-clicking on the table header.	boolean	.sortingEnabled	Behavior
TestData	Toggle this property to fill in the table's data with random data.	boolean	.test	Misc
View Dataset	A read-only copy of the data as it appears on screen in the table. The purpose of this property is to preserve the column ordering, column visibility, and applied sorting order. Other attributes, such as formatting, will not be preserved in this dataset.	Dataset	.viewDataset	Data
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

- Description

  Returns a list of ints representing the currently selected columns.

- Parameters

  none

- Return

  Object of Integers - An object containing integers that represent the indices of the selected columns. Can be iterated over in a similar manner to a Python List.

- Description

  Returns a list of ints representing the currently selected rows.

- Parameters

  none

- Return

  Object of Integers - An object containing integers that represent the indices of the selected rows. Can be iterated over in a similar manner to a Python List.

- Description

  This specialized print function will paginate the table onto multiple pages.This function accepts keyword-style invocation.

- Keyword Args

  boolean fitWidth - If true, the table's width will be stretched to fit across one page's width. Rows will still paginate normally. If false, the table will paginate columns onto extra pages. (default = true) [optional]

  String headerFormat - A string to use as the table's page header. The substring "{0}" will be replaced with the current page number. (default = None) [optional]

  String footerFormat - A string to use as the table's page footer. The substring "{0}" will be replaced with the current page number. (default = "Page {0}") [optional]

  boolean showDialog - Used to determine if the print dialog should be shown to the user. Default is true. [optional]

  boolean landscape - Used to specify portrait (0) or landscape (1) mode. Default is portrait (0). [optional]

- Return

  boolean - True if the print job was successful.

- Description

  Used to set a column's width at runtime.

- Parameters

  int column - Column to adjust.

  int width - Width in pixels.

- Return

  Nothing

Extension Functions

- Description

  Provides a chance to configure the contents of each cell. Returns a dictionary of name-value pairs with the desired attributes. Available attributes (and their Java types) include: 'background' (color), 'border' (border), 'font' (font), 'foreground' (color), 'horizontalAlignment' (int), 'iconPath' (string), 'text' (string), 'toolTipText' (string), 'verticalAlignment' (int).

  You can also specify the attribute 'renderer', which is expected to be a javax.swing.JComponent which will be used to render the cell.

- Parameters

  Component self - A reference to the component that is invoking this function.

  Object value - The value in the dataset at this cell.

  string textValue - The text the table expects to display at this cell (may be overriden by including 'text' attribute returned in dictionary).

  boolean selected - A boolean indicating whether this cell is currently selected.

  int rowIndex - The index of the row in the underlying dataset

  int colIndex - The index of the column in the underlying dataset

  string colName - The name of the column in the underlying dataset

  int rowView - The index of the row, as it appears in the table view (affected by sorting)

  int colView - The index of the column, as it appears in the table view (affected by column re-arranging and hiding)

- Return

  Dictionary of Attributes

- Description

  Provides a change to configure how each column is edited. Returns a dictionary of name-value pairs with desired editor attributes. Visual attributes to modify existing editors include: 'background', 'border', 'font', 'foreground', 'horizontalAlignment', 'toolTipText', and 'verticalAlignment'

  If the attribute 'options' is specified, it is expected to be a list of tuples representing (value, label). The editor in this case will become a dropdown list.

  If the attribute 'editor' is specified, it is expected to be an instance of javax.swing.table. TableCellEditor, and other attributes will be ignored.

The following feature is new in Ignition version **8.0.16**
Click here to check out the other new features

As of 8.0.16, the 'options' editor on the Power Table’s configureEditor Extension Function now accepts a rowHeight key allowing you to change the height of items in the dropdown. For example:

```
return {'options': [(0, 'Option A'), (1, 'Option B')], 'rowHeight':100}
```

- Parameters

  Component self - A reference to the component that is invoking this function

  int colIndex - The index of the column in the underlying dataset

  string colName - The name of the column in the underlying dataset

- Return

  Dictionary of name value pairs

- Description

  Provides a chance to configure the style of each column header. Return a dictionary of name-value pairs with the designed attributes. Availible attributes include: 'background', 'border', 'font', 'foreground', 'horizontalAlignment', 'toolTipText', 'verticalAlignment'

- Parameters

  Component self - A reference to the component that is invoking this function

  int colIndex - The index of the column in the underlying dataset

  string colName - The name of the column in the underlying dataset

- Return

  Dictionary of name value pairs

- Description

  Called when the window containing this table is opened, or the template containing it is loaded. Provides a chance to initialize the table further, for example, selecting a specific row.

- Parameters

  Component self - A reference to the component that is invoking this function

- Return

  Nothing

- Description

  Returns a boolean that determines whether or not the current cell is editable.

- Parameters

  Component self - A reference to the component that is invoking this function.

  int rowIndex - Index of the row that was edited, relative to the underlying dataset.

  int colIndex - Index of the column that was edited, relative to the underlying dataset.

  string colName - Name of the column in the underlying dataset.

  Object value - The value at the cell location.

- Return

  boolean

- Description

  Called when the user has edited a cell in the table. It is up to the implementation of this function to alter the underlying data that drives the table. This might mean altering the dataset directly, or running a SQL UPDATE query to update data in the database.

If the script on this extension function causes the Power Table to lose focus, the cell commit will occur twice. For example, if system.gui.confirm() is called, then two confirmation boxes will appear. In cases where the script will cause the focus to switch between multiple objects, the script should be placed in a function, and wrapped in a call to system.util.invokeLater()

```
        def myFunction():
                """
                Do your work here
                """
                system.gui.messageBox("Assuming you don't change focus
outside of this script\nYou will only see this message once per cell
edit")
        system.util.invokeLater(myFunction)
```

- Parameters

  Component self - A reference to the component that is invoking this function.

  int rowIndex - Index of the row that was edited, relative to the underlying dataset.

  int colIndex - Index of the column that was edited, relative to the underlying dataset.

  string colName - Name of the column in the underlying dataset.

  Object oldValue - The old value at the location, before it was edited.

  Object newValue - The new value input by the user.

- Return

  Nothing

The following feature is new in Ignition version **8.0.4**
Click here to check out the other new features

- Description

  Called when the user initially presses the mouse button on a table cell.

- Parameters

  Component self - A reference to the component that is invoking this function.

  int rowIndex - Index of the row, starting at 0, relative to the underlying dataset.

  int colIndex - Index of the column starting at 0, relative to the underlying dataset.

  Object value - The value at the location clicked on.

  MouseEvent event - The MouseEvent object that caused this pressed event.

- Return

  Nothing

The following feature is new in Ignition version **8.0.4**
Click here to check out the other new features

- Description

  Called when the user releases the mouse button on a table cell.

- Parameters

  Component self - A reference to the component that is invoking this function.

  int rowIndex - Index of the row, starting at 0, relative to the underlying dataset.

  int colIndex - Index of the column starting at 0, relative to the underlying dataset.

  Object value - The value at the location that the mouse is released on.

  MouseEvent event - The MouseEvent object that caused this released event.

- Return

  Nothing

The following feature is new in Ignition version **8.0.4**
Click here to check out the other new features

- Description

  Called when the user clicks on a table cell.

- Parameters

  Component self - A reference to the component that is invoking this function.

  int rowIndex - Index of the row, starting at 0, relative to the underlying dataset.

  int colIndex - Index of the column starting at 0, relative to the underlying dataset.

  Object value - The value at the location clicked on.

  MouseEvent event - The MouseEvent object that caused this click event.

- Return

  Nothing

- Description

  Called when the user double-clicks on a table cell.

- Parameters

  Component self - A reference to the component that is invoking this function.

  int rowIndex - Index of the row, starting at 0, relative to the underlying dataset.

  int colIndex - Index of the column starting at 0, relative to the underlying dataset.

  Object value - The value at the location clicked on.

  MouseEvent event - The MouseEvent object that caused this double-click event.

- Return

  Nothing

- Description

  Called when the user right-clicks on a table cell. This would be the appropriate time to create and display a popup menu.

- Parameters

  Component self - A reference to the component that is invoking this function.

  int rowIndex - Index of the row, starting at 0, relative to the underlying dataset.

  int colIndex - Index of the column starting at 0, relative to the underlying dataset.

  string colName - Name of the column in the underlying dataset.

  Object value - The value at the location clicked on.

  MouseEvent event - The MouseEvent object that caused this double-click event.

- Return

  Nothing

- Description

  Called when the user has dropped rows on this table. Note that the rows may have come from this table or another table. The source table must have dragging enabled.

- Parameters

  Component self - A reference to the component that is invoking this function

  Component sourceTable - A reference to the table that the rows were dragged and dropped in the same table.

  list rows - An array of the rows indices that were dragged, in the order they were selected

  Dataset rowData - A dataset containing the rows that were dragged

  int dropIndexLocation - Row index where the rows were dropped

- Return

  Nothing

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

This component has a table customizer that allows customization of the individual columns including hiding columns, enabling editing, changing format, etc. It is important to note that when editing cells directly in the Power Table, it doesn't modify the underlying dataset. You can use the onCellEdited extension function and uncomment the sample code to make table edits change the underlying Dataset, or even the original source of data (ie: if using a SQL Query).

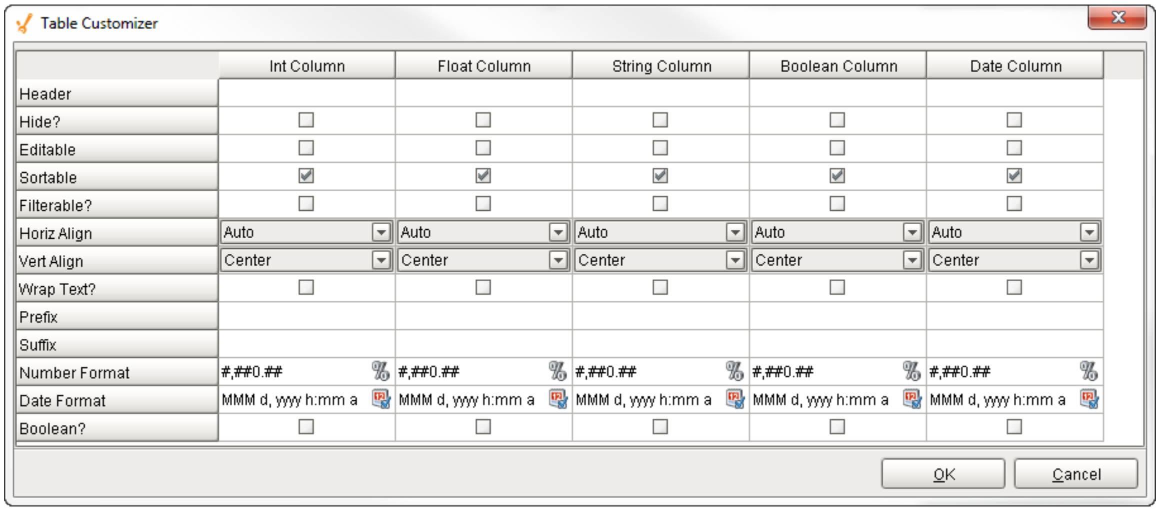
- Vision - Power Table Customizer
- Vision Component Customizers
- Understanding Component Customizers

Examples

Code Snippet

```
#Example of an onRowsDropped() extension script for two power tables with identical columns:

def onRowsDropped(self, sourceTable, rows, rowData, dropIndexLocation):
    if self != sourceTable:
        destDataset = self.getData()
        pyRowData = system.dataset.toPyDataSet(rowData)
        # Loop thru all the rows that have been selected and dragged to the
        # destination table.
        for row in pyRowData:
            newRow = []
            for column in row:
                newRow.append(column)
            destDataset = system.dataset.addRow(destDataset, dropIndexLocation, newRow)
        # Adds the rows to the destination table.
        self.setData(destDataset)
        # Optional. Deletes the dragged rows from the source table.
        sourceDataset = system.dataset.deleteRows(sourceTable.getData(), rows)
        sourceTable.setData(sourceDataset)
    else:
        system.gui.messageBox("Dropping on to same table not supported")
        # To drop onto the same table, the new row indices must be calculated
        # for both the dropped and deleted rows, taking changes into account.
```

Vision - Power Table Customizer



	Int Column	Float Column	String Column	Boolean Column	Date Column
Header					
Hide?	☐	☐	☐	☐	☐
Editable	☐	☐	☐	☐	☐
Sortable	☑	☑	☑	☑	☑
Filterable?	☐	☐	☐	☐	☐
Horiz Align	Auto	Auto	Auto	Auto	Auto
Vert Align	Center	Center	Center	Center	Center
Wrap Text?	☐	☐	☐	☐	☐
Prefix					
Suffix					
Number Format	#,##0.##	#,##0.##	#,##0.##	#,##0.##	#,##0.##
Date Format	MMM d, yyy h:mm a	MMM d, yyy h:mm a	MMM d, yyy h:mm a	MMM d, yyy h:mm a	MMM d, yyy h:mm a
Boolean?	☐	☐	☐	☐	☐

Description

The Vision - Power Table offers the same functionality as the classic Vision - Table component, but has more features. Just like the classic Table, it not only provides a Table Customizer that allows you to make changes to the table columns, but coupled with its data properties and use of extension functions, it lets you configure how each cell in the table looks and behaves.

Customizers

The Table Customizer allows you to configure how you want the table to look to users. When you open the customizer, you'll notice that the data is formatted into different columns. The left column contains all the Table Customizer properties. For each column in the customizer, you can assign a header name, hide the column, make it editable and sortable, change the horizontal and vertical alignment of text, and select a number format and date format style.

TestData Property

If you want to test how the Table Customizer works in the Power Table, drag a Power Table on to your workspace, go to the Test Data property in the Property Editor, and check the 'false' checkbox. It will automatically fill the table with some test data so you get test out the Table Customizer.

- Vision - Power Table
- Component Customizers
- Understanding Component Customizers

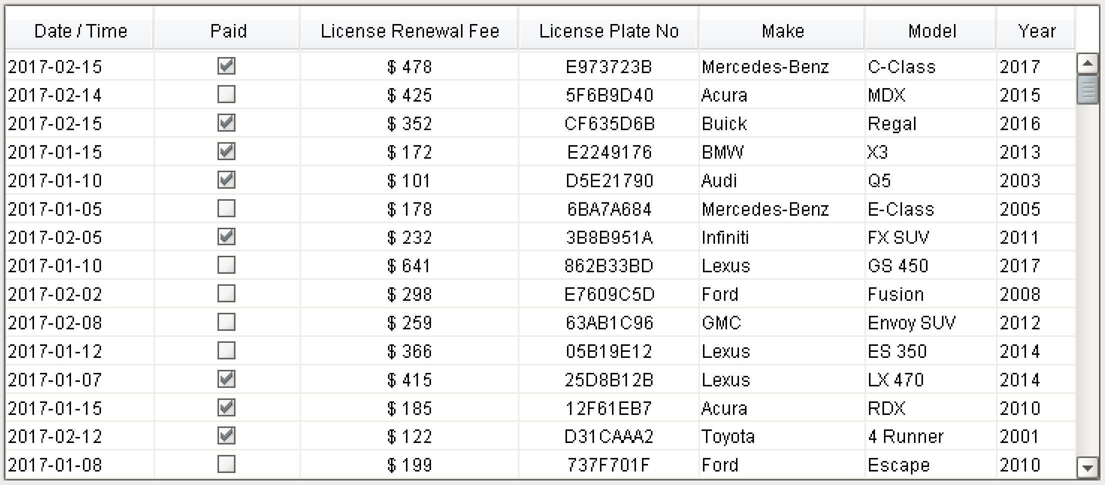
Table Customizer Properties

Property	Description
Header	Provide a custom name to the column header.
Hide	Hides the column.
Editable	Allows the editing of the cell pertaining to the column. While the cell will be editable, the edit won't do anything and the cell will revert back to its previous value unless the edit is handled by the onCellEdited extension function.
Sortable	Allows the user to sort the table according to the selected column.
Filterable	Allows the user to filter the table according to the selected column.
Horiz Align	Aligns the contents of the column: Auto, Left, Center, Right.
Vert Align	Aligns the contents of the column: Top, Center, Bottom.
Wrap Text	The text will wrap if its contents are longer than the width of the cell.
Prefix	A custom text that proceeds the contents of each cell.
Suffix	A custom text that follows the contents of each cell.
Number Format	A format of the cell if the contents of the cell are number types.
Date Format	A format of the cell if the contents of the cell are date types.
Boolean	Changes the contents of the cell to reflect a 'check box' look and feel.

Power Table Customizer

In this example, compare the columns in the dataset and the table customizer to see how the individual columns were customized to create the chart below.

Power Table



Date / Time	Paid	License Renewal Fee	License Plate No	Make	Model	Year
2017-02-15	✓	$ 478	E973723B	Mercedes-Benz	C-Class	2017
2017-02-14	☐	$ 425	5F6B9D40	Acura	MDX	2015
2017-02-15	✓	$ 352	CF635D6B	Buick	Regal	2016
2017-01-15	✓	$ 172	E2249176	BMW	X3	2013
2017-01-10	✓	$ 101	D5E21790	Audi	Q5	2003
2017-01-05	☐	$ 178	6BA7A684	Mercedes-Benz	E-Class	2005
2017-02-05	✓	$ 232	3B8B951A	Infiniti	FX SUV	2011
2017-01-10	☐	$ 641	862B33BD	Lexus	GS 450	2017
2017-02-02	☐	$ 298	E7609C5D	Ford	Fusion	2008
2017-02-08	☐	$ 259	63AB1C96	GMC	Envoy SUV	2012
2017-01-12	☐	$ 366	05B19E12	Lexus	ES 350	2014
2017-01-07	✓	$ 415	25D8B12B	Lexus	LX 470	2014
2017-01-15	✓	$ 185	12F61EB7	Acura	RDX	2010
2017-02-12	✓	$ 122	D31CAAA2	Toyota	4 Runner	2001
2017-01-08	☐	$ 199	737F701F	Ford	Escape	2010

Table Customizer

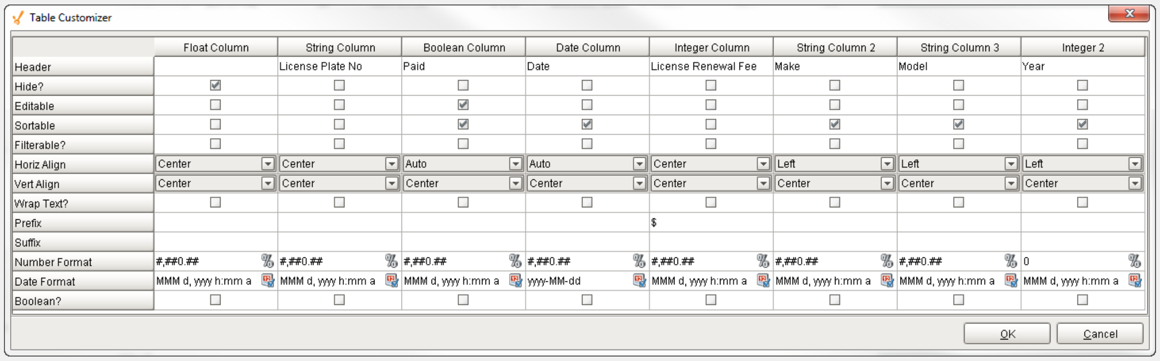
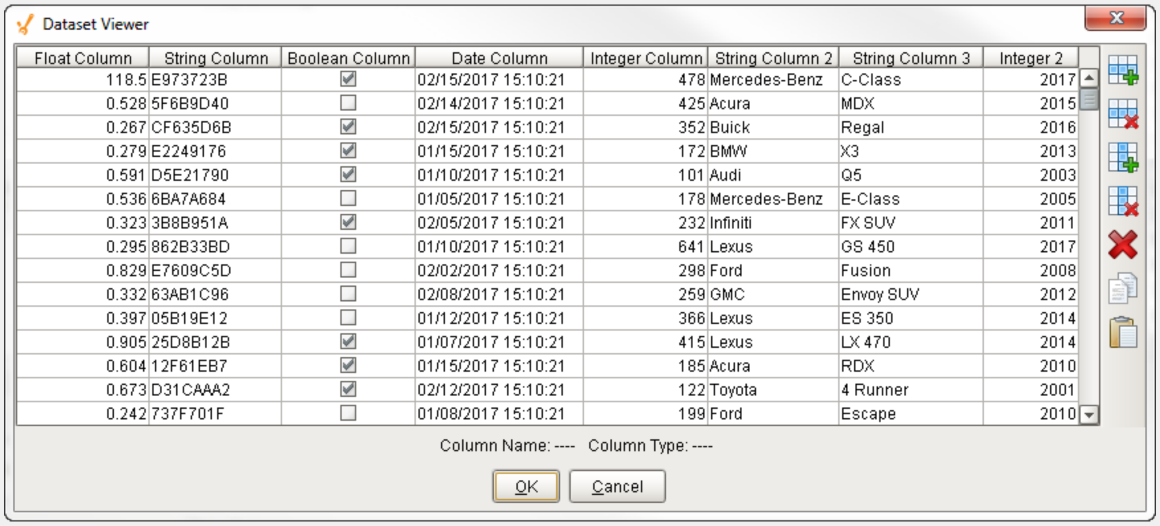


Table Customizer

	Float Column	String Column	Boolean Column	Date Column	Integer Column	String Column 2	String Column 3	Integer 2
Header		License Plate No	Paid	Date	License Renewal Fee	Make	Model	Year
Hide?	✓	☐	☐	☐	☐	☐	☐	☐
Editable	☐	☐	✓	☐	☐	☐	☐	☐
Sortable	☐	☐	✓	✓	☐	✓	✓	✓
Filterable?	☐	☐	☐	☐	☐	☐	☐	☐
Horiz Align	Center	Center	Auto	Auto	Center	Left	Left	Left
Vert Align	Center	Center	Center	Center	Center	Center	Center	Center
Wrap Text?	☐	☐	☐	☐	☐	☐	☐	☐
Prefix					$			
Suffix								
Number Format	#,##0.##	#,##0.##	#,##0.##	#,##0.##	#,##0.##	#,##0.##	#,##0.##	0
Date Format	MMM d, yyyy h:mm a	MMM d, yyyy h:mm a	MMM d, yyyy h:mm a	yyyy-MM-dd	MMM d, yyyy h:mm a	MMM d, yyyy h:mm a	MMM d, yyyy h:mm a	MMM d, yyyy h:mm a
Boolean?	☐	☐	☐	☐	☐	☐	☐	☐

OK Cancel

Data Property Dataset



Dataset Viewer

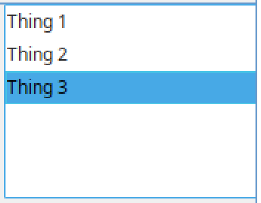
Float Column	String Column	Boolean Column	Date Column	Integer Column	String Column 2	String Column 3	Integer 2
118.5	E973723B	✓	02/15/2017 15:10:21	478	Mercedes-Benz	C-Class	2017
0.528	5F6B9D40	☐	02/14/2017 15:10:21	425	Acura	MDX	2015
0.267	CF635D6B	✓	02/15/2017 15:10:21	352	Buick	Regal	2016
0.279	E2249176	✓	01/15/2017 15:10:21	172	BMW	X3	2013
0.591	D5E21790	✓	01/10/2017 15:10:21	101	Audi	Q5	2003
0.536	6BA7A684	☐	01/05/2017 15:10:21	178	Mercedes-Benz	E-Class	2005
0.323	3B8B951A	✓	02/05/2017 15:10:21	232	Infiniti	FX SUV	2011
0.295	862B33BD	☐	01/10/2017 15:10:21	641	Lexus	GS 450	2017
0.829	E7609C5D	☐	02/02/2017 15:10:21	298	Ford	Fusion	2008
0.332	63AB1C96	☐	02/08/2017 15:10:21	259	GMC	Envoy SUV	2012
0.397	05B19E12	☐	01/12/2017 15:10:21	366	Lexus	ES 350	2014
0.905	25D8B12B	✓	01/07/2017 15:10:21	415	Lexus	LX 470	2014
0.604	12F61EB7	✓	01/15/2017 15:10:21	185	Acura	RDX	2010
0.673	D31CAAA2	✓	02/12/2017 15:10:21	122	Toyota	4 Runner	2001
0.242	737F701F	☐	01/08/2017 15:10:21	199	Ford	Escape	2010

Column Name: ---- Column Type: ----

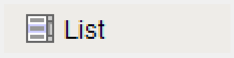
OK Cancel

Vision - List

General



Component Palette Icon:



Description

The List component displays a list of options, allowing freeform selection of the items. It is powered by a Dataset, from which it displays the first column.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Data	The data for the list. If multiple columns exist, the first will be used.	Dataset	.data	Data
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See Color Selector.	Color	.foreground	Appearance
Layout Orientation	This property defines the orientation of the list elements.	int	.layoutOrientation	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common

Opaque	If false, backgrounds are not drawn. If true, backgrounds are drawn.	boolean	.opaque	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Row Height	An integer specifying the row height, or -1 for automatic row height.	int	.rowHeight	Appearance
Selected Background	The color of the background for the selected cell(s).	Color	.selectedBackground	Appearance
Selected Focus Border	The border for the selected, focused cell.	Border	.selectedFocusBorder	Appearance
Selected Foreground	The color of the foreground for the selected cell(s). See Color Selector.	Color	.selectedForeground	Appearance
Selected Index	The index of the selected cell, or -1 if none.	int	.selectedIndex	Data
Selection Mode	This mode determines if only one cell can be selected at once, or single or multiple intervals.	int	.selectionMode	Behavior
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Visible Row Count	An integer specifying the preferred number of rows to display without requiring scrolling.	int	.visibleRowCount	Appearance
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

- Description

  Adds the options at indexes start through end (inclusive) to the selected options.

- Parameters

  int start - The first index (stating at 0) to add to the selection.

  int end - The last index (stating at 0) to add to the selection.

- Return

  Nothing

- Description

    Clears the current selection, making nothing selected.
- Parameters

    Nothing
- Return

    Nothing
- Description

    Returns a list of the selected indices in increasing order. Returns an empty list if nothing is selected.
- Parameters

    Nothing
- Return

    List of Integers
- Description

    Returns the currently selected value, or None if the selection is empty.
- Parameters

    Nothing
- Return

    Object
- Description

    Returns a list of the currently selected values. Returns an empty list if the selection is empty.
- Parameters

    Nothing
- Return

    Object[]
- Description

    Checks whether or not the given index is currently selected.
- Parameters

    int index
- Return

    boolean
- Description

    Checks to see if anything is selected in the list or not.
- Parameters

    Nothing
- Return

    boolean

- Description

  Sets the currently selected value to the argument, if found in the list.

- Parameters

  Object value

- Return

  Nothing

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event.
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when a component that had the input focus lost it to another component.

.source	The component that fired this event
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

An integer that indicates whether the state was changed to "Selected" (on) or "Deselected" (off). Compare this to the event object's constants to determine what the new state is.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is released and the source component has the input focus. Works for all characters, including non-printable ones, such as SHIFT and F3.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is pressed and then released when source component has the input focus. Only works for characters that can be printed on the screen.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires *aft er* the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event.
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed.
. property Name	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers
- Style Customizer

Examples

Code Snippet

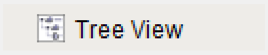
```
#The following code will print the selected value to the console when called on the
'mouseClicked' event handler.
value = event.source.getSelectedValue()
print value
```

Vision - Tree View

General

East Area
West Area

Component Palette Icon:



INDUCTIVE UNIVERSITY

Tree View

Watch the Video

Description

The Tree View component can display any tree hierarchy. It is configured by filling in a dataset. Each column title in the dataset is a property of the Tree View Customizer.

Each row in the dataset will become a node in the tree. Each node has a path that determines its location in the tree, for example, "West Area/Process/Valve1". The Separation Character property (by default is a forward-slash), dictates how the paths are broken up. Any missing folder nodes needed by a leaf node are created implicitly. The other columns in the dataset besides "Path" are used to configure the look for the node, both when it is selected and when it is not. All column properties in the dataset are described in the Tree View Customizer.

Tree View Component Properties

Name	Description	Property Type	Scripting	Category
Auto Expand	If true, the tree will automatically expand the tree structure up to the level specified by Auto Expansion Level.	boolean	.autoExpand	Behavior
Auto Expansion Level	If Auto Expand is true, this is the depth level that will be expanded. Zero means expand-all.	int	.autoExpansionLevel	Behavior
Auto Sort	Whether or not to automatically sort the tree.	boolean	.autoSort	Behavior
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance

Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Default Closed Icon	The default closed icon if no icon is set.	String	.defaultClosedIconPath	Appearance
Default Leaf Icon	The default leaf icon if no icon is set.	String	.defaultLeafIconPath	Appearance
Default Node Background	The default background of a node if no background is set. See Color Selector.	Color	.defaultBackground	Appearance
Default Node Border	The default border of a node if no border is set.	Border	.defaultBorder	Appearance
Default Node Foreground	The default foreground of a node if no foreground is set. See Color Selector.	Color	.defaultForeground	Appearance
Default Node Selected Background	The default selected background of a node if no background is set. See Color Selector.	Color	.defaultSelectedBackground	Appearance
Default Node Selected Border	The default selected border of a node if no border is set.	Border	.defaultSelectedBorder	Appearance
Default Node Selected Foreground	The default selected foreground of a node if no foreground is set. See Color Selector.	Color	.defaultSelectedForeground	Appearance
Default Open Icon	The default open icon if no icon is set.	String	.defaultOpenIconPath	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Items	Contains the items of the tree view.	Dataset	.data	Data
Line Style	The tree's line style.	int	.lineStyle	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Row Height	The height of each row in the tree.	int	.rowHeight	Appearance

Selected Item	The index of the currently selected item, or -1 if no selection.	int	.selectedItem	Data
Selected Path	The path of the currently selected item, or "" if no selection.	String	.selectedPath	Data
Selection Mode	What kind of selection regions does the tree allow.	int	.selectionMode	Behavior
Separation Character	The separation character for the path.	String	.separationCharacter	Behavior
Show Root Handles	Whether or not to show handles next to parent nodes.	boolean	.showRootHandles	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

- Description

  Clears the current selection.
- Parameters

  Nothing
- Return

  Nothing
- Description

  Collapses all nodes in the tree.
- Parameters

  Nothing
- Return

  Nothing
- Description

  Expands all nodes in the tree.
- Parameters

  Nothing
- Return

  Nothing
- Description

  Returns a list of the selected item's indexes. These are the row indexes that the selected tree nodes were found in the underlying dataset. Implicitly created folder nodes that have no index will not be included.
- Parameters

  Nothing
- Return

  List of Integers
- Description

  Returns a list of the selected item's paths. A path to an item is the path to its parent plus its normal (non-selected) text.
- Parameters

  Nothing
- Return

  List of Strings

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

The Tree View customizer allows for easy custom manipulation of the tree view components underlying formatting.

- Tree View Customizer
- Vision Component Customizers

Examples

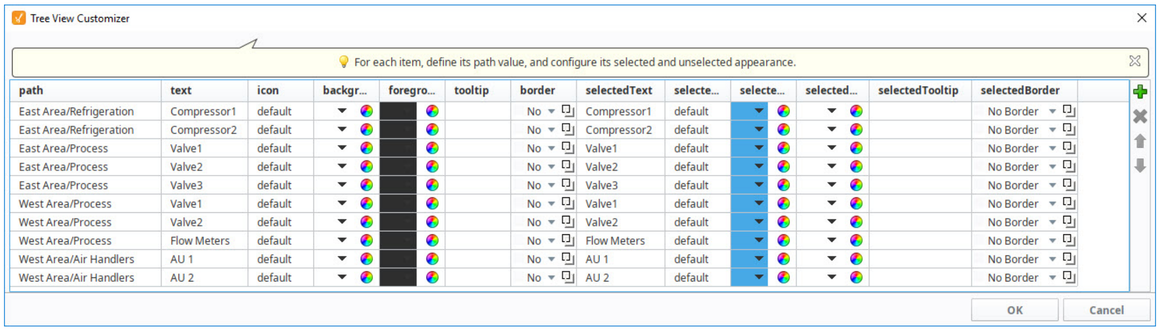
Expression Snippet

```
//The Selected Item property will be updated as the user selects different nodes in the tree.
//It represents the index in the Items dataset at which the node is defined. If the selected
//node was implicitly created, the Selected Item will be -1.
//You can use this index to get the path and name of the selected node with an expression
binding like this:
if ({Root Container.Tree View.selectedItem}<0,"n/a",{Root Container.Tree View.data}[{Root
Container.Tree View.selectedItem},"text"])
```

Script Snippet

```
#This script will swap to the script that was double clicked on, if this code is placed in
the mouseClicked event handler for the treeview
#This script utilizes an extra column called windowPath that contains the full path to the
window. You can add an extra column to the Items dataset property
#as long as the column name doesn't match one of the reserved column titles listed above.
if event.clickCount == 2:
        row = event.source.selectedItem
        data = event.source.data
        if row != -1:
                # Grab the window path value out of the tree view's items dataset
                windowPath = data.getValueAt(row, "windowPath")
                system.nav.swapTo(windowPath)
```

Vision - Tree View Customizer



Tree View Customizer

For each item, define its path value, and configure its selected and unselected appearance.

path	text	icon	backgr...	foregro...	tooltip	border	selectedText	selecte...	selecte...	selected...	selectedTooltip	selectedBorder
East Area/Refrigeration	Compressor1	default		⬛		No	Compressor1	default	🟦			No Border
East Area/Refrigeration	Compressor2	default		⬛		No	Compressor2	default	🟦			No Border
East Area/Process	Valve1	default		⬛		No	Valve1	default	🟦			No Border
East Area/Process	Valve2	default		⬛		No	Valve2	default	🟦			No Border
East Area/Process	Valve3	default		⬛		No	Valve3	default	🟦			No Border
West Area/Process	Valve1	default		⬛		No	Valve1	default	🟦			No Border
West Area/Process	Valve2	default		⬛		No	Valve2	default	🟦			No Border
West Area/Process	Flow Meters	default		⬛		No	Flow Meters	default	🟦			No Border
West Area/Air Handlers	AU 1	default		⬛		No	AU 1	default	🟦			No Border
West Area/Air Handlers	AU 2	default		⬛		No	AU 2	default	🟦			No Border

OK Cancel

Description
The Tree View has its own customizer which allows you to easily configure the items dataset property. The customizer provides some useful dropdowns and color selectors for certain properties that require more than just a name or a path. You can add and remove nodes, and change the node hierarchy and appearance through the properties in the dataset.

While the Customizer allows you to configure the columns of the Items dataset, the customizer will not display any columns that the user adds to the dataset. However, user added columns are still configurable in the dataset itself, and can be used to store additional information about each item such as a window path.

Customizers

The Tree View Customizer allows you to easily configure how you want the tree view to look to users. When you open the customizer for the first time, you'll notice the dataset contains some predefined nodes and settings. Each row in the dataset represents a node in the tree. Each column in the dataset represents properties that configure the appearance of the tree to look a certain way.

Configuring the Tree View Customizer is very straightforward. To add a node to the tree, click the green icon on the right side of the window, and a new row will be added at the the bottom of the dataset. All the columns will default to the predefined properties with the exception of the "path" to the node's location. This field will be blank so you need to enter a path to the node. You can edit any of the of the preset properties. At a minimum, you should always edit the **Text** and **SelectedText** properties replacing the default names with a more appropriate name so the item is easily identifiable when it is selected and unselected in the tree. You can also move a node up or down the tree hierarchy using the **Move Up** or **Move Down** arrows on the right side of the window. To delete a node from the tree, simply select the node and hit **Delete**.

The additional properties are optional, but can enhance your tree view for your users. For example:

- To change an icon for any node in the tree, choose an icon from the Image Management Tool. All you need to do is right click on the icon in the Image Management tool and select Copy Path, and paste it in the Icon field for that node.
- Add a tooltip for any item in the tree by simply typing in your tooltip in the Tooltip field for that node. When you hover over the item in the tree view, you'll see your tooltip.
- Add a foreground and background color for any item in the tree when it is selected or unselected.
- Add a border for any item in the tree when it selected or unselected.

The references to optional properties in the table below means that a dataset does not need to have them present in the dataset for the tree to render and function.

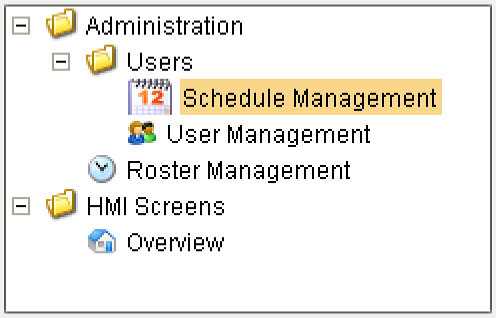
Tree View Customizer Properties

Property	Description
Path	Path that determines the node's locaton. Broken up into a list by splitting on the separation character.
Text	Text of the node while not selected.
Icon	Path to an icon for the node. Use the value: "default" to use the tree automatic folder/leaf icons. (optional)
Backgro und	Controls the background appearance of the unselected item. A string column that will be coerced into a color for the unselected background. (e.g., "white" or "(255,255,255)". Use an empty string to use the default color. (optional)
Foregrou nd	Control the foreground appearance of the unselected item. A string representation of the unselected foreground color. (optional)
Tooltip	If not empty, will be use as the tooltip for the node. (optional)
Border	A string that will be coerced into a border for the node while unselected. May be empty. (optional)
Selected Text	Text of the node while selected. (optional)
Selected Icon	A path to an icon for the node while selected. Use the value: "default" to use the tree automatic folder/leaf icons. (optional)
Selected Backgro und	Controls the background appearance of the selected item. A string representation of the the selected background color. (optional)
Selected Foregrou nd	Controls the background appearance of the selected item. A string representation of the selected foreground color. (optional)
Selected Tooltip	If not empty, will be used as the tooltip for the node while selected. (optional)
Selected Border	A string that will be coerced into a border for the node while selected. May be emplty. (optional)

Example

Tree View with Larger Version of SelectedIcons

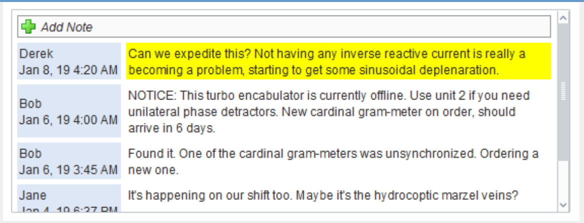
Below is an example configuration of the tree view's items property. Notice how not all of the fields listed in the property table above are used, because there are certain properties that are not necessary to build our tree view. A larger version of the images was chosen for the SelectedIcon, so that when an item gets selected, not only does the background color change, but the size of the image changes as well.



Path	Text	Icon	Background	Foreground	SelectedText	SelectedIcon	SelectedBackground	Selec
HMI Screens	Overview	Builtin/icons/16/home.png	color(255, 255, 255, 255)	color(0, 0, 0, 255)	Overview	Builtin/icons/24/home.png	color(250, 214, 138, 255)	color(
Administration/Users	User Management	Builtin/icons/16/users3.png	color(255, 255, 255, 255)	color(0, 0, 0, 255)	User Management	Builtin/icons/24/users3.png	color(250, 214, 138, 255)	color(
Administration/Users	Schedule Management	Builtin/icons/16/calendar.png	color(255, 255, 255, 255)	color(0, 0, 0, 255)	Schedule Management	Builtin/icons/24/calendar.png	color(250, 214, 138, 255)	color(
Administration	Roster Management	Builtin/icons/16/clock.png	color(255, 255, 255, 255)	color(0, 0, 0, 255)	Roster Management	Builtin/icons/24/clock.png	color(250, 214, 138, 255)	color(

Vision - Comments Panel

General



Derek Jan 8, 19 4:20 AM	Can we expedite this? Not having any inverse reactive current is really a becoming a problem, starting to get some sinusoidal deplenaration.
Bob Jan 6, 19 4:00 AM	NOTICE: This turbo encabulator is currently offline. Use unit 2 if you need unilateral phase detractors. New cardinal gram-meter on order, should arrive in 6 days.
Bob Jan 6, 19 3:45 AM	Found it. One of the cardinal gram-meters was unsynchronized. Ordering a new one.
Jane	It's happening on our shift too. Maybe it's the hydrocoptic marzel veins?

Component Palette Icon:

Comments Panel

INDUCTIVE UNIVERSITY

Comments Panel

Watch the Video

Looking for documentation on the legacy Comments Panel component? Please see the Legacy Comments Panel page.

Not sure which version you are looking at? The Legacy version of this component has several properties that the new one does not: "Insert Query 1", " Insert Query 2", "Delete Query", "Unstick Query", and "Download Attachment Query".

Description

The comments panel is used to power a blog-style comments system within your project. This can be useful for ad-hoc collaboration and communication between shifts, remote users, etc. This component is driven by a dataset that should be bound to a SQL query. Unlike most components, this component has built-in functionality to alter an external database. It expects three tables in the database, and that they are queried properly on the data property.

You can opt out of this three-table default system by simply making use of the Extension Functions on the component. See below for more details.

Behavior Description

Three-Table (Default) Configuration

The following section assumes the default configuration: all Extension Functions on the component are disabled.

Required Database Tables

The default behavior of the component expects three database tables be present under the same database connection, and each table needs to have certain columns with specific names.

Table: Notes

Stores all of the notes across the board.

Column Name	Description	Data Type
id	An auto-incrementing integer that is the primary key. This maps to the ID field in the dataset.	Integer
whoID	A mapping to the Username field in the dataset	Integer
tStamp	A mapping to the Timestamp field in the dataset	Date or Datetime
note	A mapping to the NoteText field in the dataset	Varchar
filename	A mapping to the AttachmentFilename in the dataset	Varchar
sticky	A mapping to the Sticky field in the dataset	Boolean or Integer
attachment	A column to hold the attachment data. LongBlobs do not exist in MSSQL, so a varbinary type must be used	LongBlob or Varbinary (depending on database)

Table: ItemNotes

Used to associate notes with other things. This allows you to have different sets of notes for different screens /objects.

Column Name	Description	Data Type
accountId	An automatically generated UUID for the Comment Panel instance. You can use the accountId in a WHERE clause on the data property so that the component only shows notes from a particular Comments Panel in the project.	Varchar
noteId	An integer that maps to the ID column on the Notes table	Integer

Tables: Users

A user mapping table that assigns an ID to each user on the table. This is easiest to do if a database authentication profile is used as the _users table automatically creates the required columns, but non-database authentication profiles can be used as long as the table is manually created and maintained.

Column Name	Description	Data Type
id	An integer that is inserted into the whoID column on the Notes table	Integer
username	The username of the user that created the note	Varchar

Configuring the Component

This component expects that its data property is populated with the following columns. The dataset in the Data property is very **specific** , and expects certain datatypes at precise positions. The order of **expected column positions** is listed below. Should the order of datatypes in the dataset differ from the order below, the names of the columns must match the **column names** below. Aliasing can be used to modify the names of the columns in the dataset.

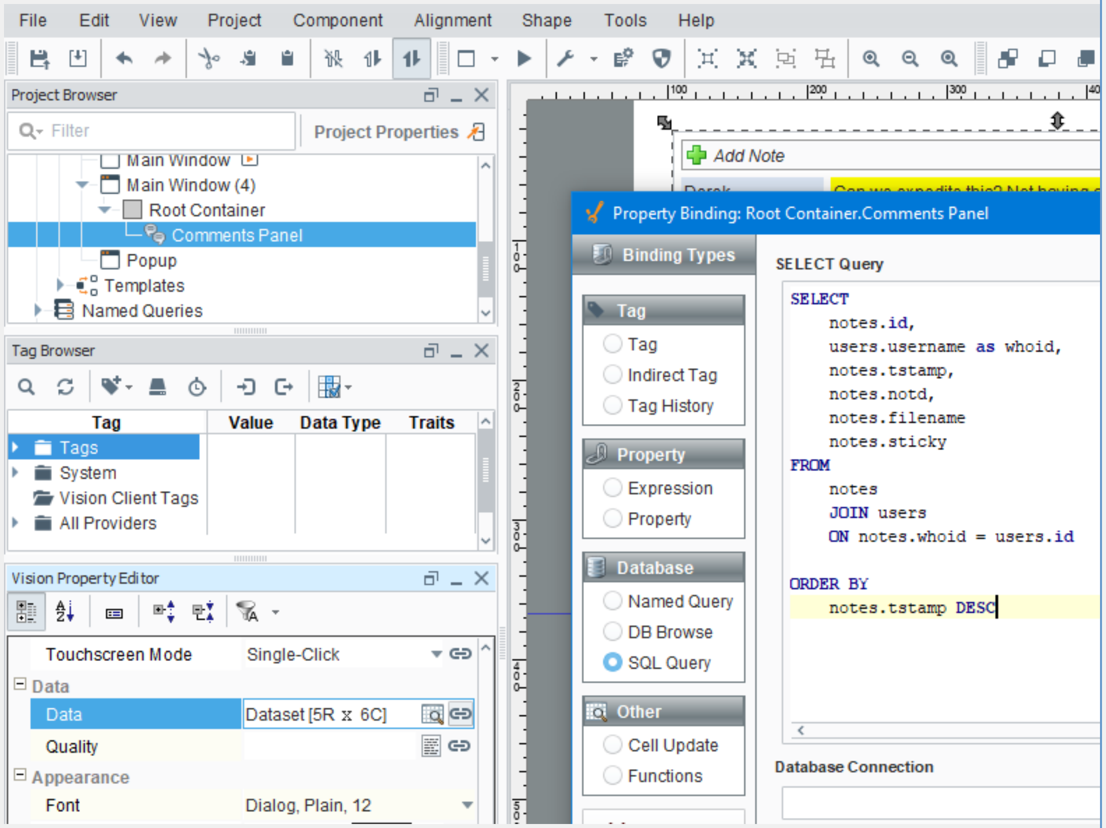
The names do need to be exact, but different names can be used as long as the query that builds the dataset uses aliases. The data type for each column in your notes table must match the table below.

Column Name	Description	Data Type	Expected Column Position
id	an integer that should be the primary key for the notes table. Used for deleting and looking up attachments	integer	0
username	the user who added the note	string /varchar	1
timestamp	when the note was added	dateTime	2
notetext	The text of the note itself	string /varchar	3
attachmentname	filename for a file attached to the note	string /varchar	4
issticky	0 or 1 indicating whether or not the note is "sticky", which means it gets highlighted and put at the top	boolean or integer	5

Example

The following query returns note data from the above tables, and displays the data on a Comments Panel component. This query should be placed in a SQL Query binding on the Data property

```
   SELECT

  notes.id,

  users.
  username
  as whoid,

  notes.
  tstamp,

  notes.
  note,

  notes.
  filename,

  notes.
  sticky

  FROM

  notes

  JOIN users
          ON
  notes.
  whoid =
  users.id

  ORDER BY

  notes.
  tstamp DESC
```



Property Binding: Root Container.Comments Panel

SELECT Query

```
SELECT
    notes.id,
    users.username as whoid,
    notes.tstamp,
    notes.notd,
    notes.filename
    notes.sticky
FROM
    notes
    JOIN users
    ON notes.whoid = users.id

ORDER BY
    notes.tstamp DESC
```

Binding Types: Tag: Tag, Indirect Tag, Tag History; Property: Expression, Property; Database: Named Query, DB Browse, [✓ SQL Query]; Other: Cell Update, Functions

Touchscreen Mode	Single-Click
Data	Dataset [5R x 6C]
Quality	
Font	Dialog, Plain, 12

By default, users can remove their own comments, and comments can have files attached.

Custom Configuration
Enabling the Extension Functions on the component will allow for custom functionality on the component. Some examples are: • Store all note data on a single database table - simply modify each Extension Function to run queries against a single database table • Save the attachment to a shared drive instead of a database column - modify insertNote to save the attachment to a hard drive. • Allow users to delete all notes by role - check the role of the user in canDelete and return True if the user has a specific role.

Properties

Name	Description	Property Type	Scripting	Category
Add Note Text	The word(s) used for the "Add Note" button.	String	.addNoteText	Appearance
Attach File Text	The word(s) used for the "Attach File" link.	String	.attachText	Appearance
Attachments Enabled	Controls whether or not files can be attached to notes.	boolean	.attachmentsEnabled	Behavior
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	border	.border	Common
Cancel Text	The word(s) used for the "Cancel" button.	String	.cancelText	Appearance
Data	Fill this DataSet in with the notes for the desired entity. Columns are: ID, Username, Timestamp, Note, Filename, IsSticky.	Dataset	.data	Data
Database Connection	Name of the database connection to run the queries against. Leave blank to use project's default connection.	String	.datasource	Behavior
Date Format	The format string to use for the date of the note.	String	.dateFormat	Appearance
Display Mode	Horizontal display mode will layout so that the comment header will be positioned to the left of the comment. Vertical display mode will have the comment header above the comment.	int	.displayMode	Behavior
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.foreground	Appearance

Header Color	The background color of the header notes. See Color Selector.	Color	.headersColor	Appearance
Maximum Attachment Size	The maximum attachment size in bytes that will be accepted. A value of 0 means no limit.	long	.maxAttachmentSize	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Note Color	The background color for notes. See Color Selector.	Color	.noteColor	Appearance
Padding	The amount of padding between the notes.	int	.padding	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Skip Audit	If true, update queries originating from this component will skip the audit system. Can be important when attachments are turned on.	boolean	.skipAudit	Behavior
Sticky Header Color	The background color of the header for sticky notes. See Color Selector.	Color	.stickyHeaderColor	Appearance
Sticky Note Color	The background color for sticky notes. See Color Selector.	Color	.stickyNoteColor	Appearance
Sticky Text	The word(s) used for the "Sticky" checkbox.	String	.stickyText	Appearance
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

- Description

  Called when a note is added.

- Parameters

  component self - A reference to the component that is invoking this function

  string note - The text contents of the note

  string filename - The full filepath to the the attachment

  string sticky - A boolean indicating whether this note should be flagged as stickied

- Return

  Nothing

- Description

  Called when a user clicks the 'delete' link on a note.

- Parameters

  component self - A reference to the component that is invoking this function

  integer id - The id of the note

- Return

  Nothing

- Description

  Called when a user clicks the 'unstick' link on a note.

- Parameters

  component self - A reference to the component that is invoking this function

  integer id - The id of the note

- Return

  Nothing

- Description

  Called when a user attempts to download an attachment from a note.

- Parameters

  component self - A reference to the component that is invoking this function

  integer id - The id of the note

- Return

  Nothing

- Description

  Returns whether or not a note with the given id can be deleted. Notes that return True will show a 'delete' link.

- Parameters

  component self - A reference to the component that is invoking this function

  integer id - The id of the note

- Return

  boolean - Notes with a True return can be deleted by the user, False return can not be deleted.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers

Examples

The following examples may need to be modifed to match the table and column names in your database.

These examples are written for a MySQL database connection. If you are using a different database, some things may need to be changed. For example, using MS SQL Server requires:

- the python value None may not be used when inserting into a byte array. NULL must be used in its place.
- binary data must be converted to a varbinary type when inserting. See the examples below

insertNote: using default table configuration

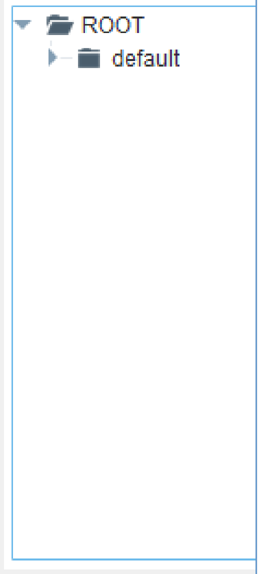
```
# Inserts a note using the three default tables: notes, users, and itemNotes.
# Also stores only the file name in the database instead of the full path to the file.
# Assumes a User ID is used in the notes table.

# determine the ID for the logged in user
user = system.db.runScalarPrepQuery("SELECT id from users where username = ?", [system.
security.getUsername()])

# determine if a file is being attached
if filename is None:
        # a file was not attached, provide a blank for the bytes
        attachmentBytes = None
else:
        # get the bytes of the file at the path the user selects
        attachmentBytes = system.file.readFileAsBytes(filename)

        # splits the file name from the file path. This way we can show just the file name on
the component
        # Using '\' as a delimiter, but python requires two since it's an escape character
        pathAndFile = filename.rsplit('\\', 1)
        filename = pathAndFile[1]

# build the query
#MySQL query
query = "INSERT INTO Notes (note, whoid, tstamp, attachment, filename, sticky) VALUES (?, ?,
CURRENT_TIMESTAMP, ?, ?, ?)"
#MSSQL Server query
# We're converting the binary data into a VARBINARY datatype, and checking for a NULL in the
attachment query.
#if attachmentBytes == None:
#        query = "INSERT INTO Notes (note, whoid, tstamp, attachment, filename, sticky)
VALUES (?, ?, CURRENT_TIMESTAMP, NULL, ?, ?)"
#else:
#        query = "INSERT INTO Notes (note, whoid, tstamp, attachment, filename, sticky)
VALUES (?, ?, CURRENT_TIMESTAMP, CONVERT(VARBINARY(MAX),?), ?, ?)"


# Set arguments and run the query
arguments = [note, user, attachmentBytes, filename, sticky]
insertId = system.db.runPrepUpdate(query, arguments, getKey=1))

# insert a row onto the itemNotes table
# replace 'MYID' with the proper code - this is based on how you are dividing the notes.
# this ID could be an area, page, or machine code, or anything else that you may want to
organize on.
myId = 'MYID'
system.db.runPrepUpdate("INSERT INTO ItemNotes (AccountId, NoteId) VALUES (?, ?)", [myId,
insertId])
```

insertNote: using a single table

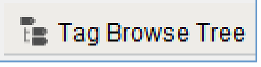
```
# Similar to the above example, but only a single database table is required.
# Assumes a User Name is used in the notes table.

# determine the name for the logged in user
user = system.security.getUsername()

# determine if a file is being attached
if filename is None:
        # a file was not attached, provide a blank for the bytes
        attachmentBytes = None

else:
        # get the bytes of the file at the path the user selects
        attachmentBytes = system.file.readFileAsBytes(filename)

        # splits the file name from the file path. This way we can show just the file name on
the component
        # Using '\' as a delimiter, but python requires 2 since it's an escape character
        pathAndFile = filename.rsplit('\\', 1)
        filename = pathAndFile[1]

# build the query
#MySQL query
query = "INSERT INTO Notes (note, whoid, tstamp, attachment, filename, sticky) VALUES (?, ?,
CURRENT_TIMESTAMP, ?, ?, ?)"
#MSSQL Server query
#We're converting the binary data into a VARBINARY datatype, and checking for a NULL in the
attachment query.
#if attachmentBytes == None:
#        query = "INSERT INTO Notes (note, whoid, tstamp, attachment, filename, sticky)
VALUES (?, ?, CURRENT_TIMESTAMP, NULL, ?, ?)"
#else:
#        query = "INSERT INTO Notes (note, whoid, tstamp, attachment, filename, sticky)
VALUES (?, ?, CURRENT_TIMESTAMP, CONVERT(VARBINARY(MAX),?), ?, ?)"

# Set arguments and run the query
arguments = [note, user, attachmentBytes, filename, sticky]
system.db.runPrepUpdate(query, arguments)
```

Vision - Tag Browse Tree

General



Component Palette Icon:



Description

The Tag Browse Tree component is similar to the Tag Browser in the Designer, allowing tags to be browsed in both the Designer and the Client, and dragged on to other components like the Easy Chart. Unlike the Tag Browser, tags can not be edited, tag properties can not be displayed, and UDT definitions can not be displayed. Tags in the component can be refreshed through scripting by calling refresh().

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Font	Font of text on this component.	Font	.font	Appearance
Include Historical Tags	Whether or not to display historical tags.	boolean	.showHistorical	Realtime Tag Tree Settings
Include Realtime Tags	Whether or not to display non-historical tags.	boolean	.showRealtime	Realtime Tag Tree Settings
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common

Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data

Root Node Path	The path of the root of this tree structure, or "" if no selection. When intentionally setting the root node, the exact syntax changes depending on what the **Tag Tree Mode** property is set to:
	
	Realtime Tag Tree: [TagProvider]FolderPath/
	
	The example below is using the "default" tag provider, and a folder named "machine_1"
	
	Example
	`[default]machine_1/`
	
	Historical Tag Tree: [DatabaseConnection/GatewayName:TagProvider]FolderPath/
	
	The example below is using a database connection named "DB", the system name of the Gateway is "ignition", the tag provider is "default" and will set the path to a folder named "machine_1"
	
	Example
	`[DB/ignition:default]machine_1/`
		String	.rootNodePath	Data

Selected Paths	Contains the paths that should be selected on the tree which should be in the format of a single string column.	Dataset	.selectedPaths	Data
Selection Mode	What kind of selection regions does the tree allow. Options are Single, Multiple - Contiguous, and Multiple - Discontiguous.	int	.selectionMode	Behavior
Show Root Handles	Whether or not to show handles next to parent nodes.	boolean	.showRootNodeHandles	Appearance
Show Root Node	Whether or not to show the root node of the tree.	boolean	.showRootNode	Appearance
Tag Tree Mode	Choose whether the tree is built using tags from the default provider or the historical provider.	int	.treeMode	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

- Description

  Called for each tag loaded into tag browse tree. Return false to hide this tag from the tree.

  Note that this is called for each **Tag**, not any folders that appear in the component.

- Parameters

  Component self- A reference to the component that is invoking this function.

  Tag Object tag - The tag itself.

- Return

  Boolean

- Description

  Returns a popup menu that will be displayed when the user triggers a popup menu (right click) on the tree. Use system.gui.createPopupMenu to create the popup menu.

- Parameters

  Component self- A reference to the component that is invoking this function.

  Tag Object clickedTag - The tag of the clicked on tree path.

  List selectedTags - The tags of the selected paths of the tree.

- Return

  JPopupMenu

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

⚠ This event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers

Examples

Code Snippet

```
# The following code shows a right-click popup menu.
# Add these lines after the """ """ section of the createPopupMenu extension function.
# Note how lines below are indented, the first def command should line up with the
# indentation of the """ """ section of the Extension Function.

        def showValue(self):
                value = str(clickedTag.getCurrentValue().value)
                system.gui.messageBox(value)


        def showLastChange(self):
                lastChange = str(clickedTag.getCurrentValue().timestamp)
                system.gui.messageBox(lastChange)

        itemsDict = {"Show Value": showValue, "Show Last Change":showLastChange}
        JPopupMenu = system.gui.createPopupMenu(itemsDict)
        return JPopupMenu
```

Vision - Charts Palette

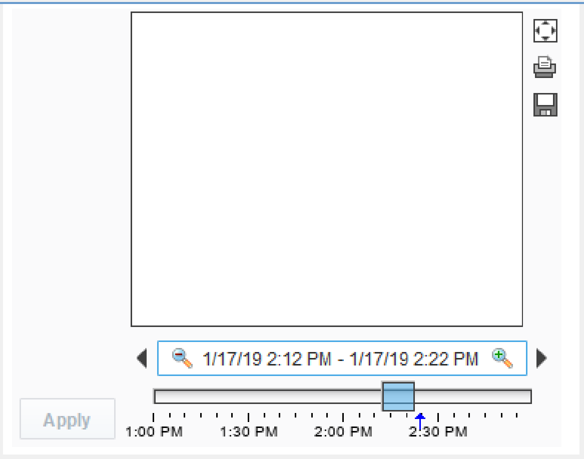
Chart Components

The following components give you various charts for displaying data.

In This Section ...

Vision - Easy Chart

General



Component Palette Icon:

Easy Chart

Description

Description

This component is used to make powerful and runtime-configurable time-series charts. It is configured by defining a set of pens and axes. Each pen represents a series of data. Pens can be many different styles, such as line, area, bar, and shape. This chart automatically creates controls for picking the time range and for hiding or displaying pens.

Features

- Easy configuration
- User-selectable set of pens
- Automatic time-selection controls
- SQL Query and/or SQLTags Historian data sources
- Automatic SPC and calculated pen support
- Zoom, Pan, X-Trace modes
- Any number of Y-axes and subplots
- Realtime or Historical

Pens

There are three kinds of pens in the Easy Chart:

1. Tag Historian Pens: These pens pull their data from the Historian system.
2. Database Pens: These pens will automatically create SQL SELECT queries to pull data from a database table. Typically, this is a table that is the target of a Historical Transaction Group.
3. Calculated Pens: These pens display a calculated dataset based off another pen, such as a moving average or Statistical Process Control (SPC) function such as the Upper Control Limit (UCL).

Modes: Realtime vs Historical

The Easy Chart can operate in three different modes. These modes affect the range of data that is displayed, the controls the user is shown, and whether or not the chart polls for data.

1. Historical Mode. In this mode, the user is shown a Vision - Date Range component to pick the range of data to fetch and display. The initial values of this component are set through properties on the chart. In historical mode, the chart does not poll.
2. Realtime Mode. In this mode, the user is given the opportunity to pick the amount of time in the past to display. For example, the last 5 minutes or the last 2 hours. The chart will poll at a rate according to the Poll Rate parameter.
3. Manual Mode. In this mode, the chart will use the values if its Start Date and End Date parameters to govern what data is displayed. Polling is controlled by having the Poll Rate at zero (polling off) or greater than zero.

Basic Chart Configuration

The Easy Chart has many properties, like other components, that control its behavior. Things like its Mode, Polling Rate, etc are configured via the properties. All of the setup for adding pens, axes, subplots, and so forth is done through its Customizer. You can also drag and drop Historian-enabled tags onto the chart directly in the Designer to add those tags as chart pens. For an example, see Using the Vision Easy Chart.

Y-Axes

The easy chart supports any number of Y-axes. To add an axis, go to the Axes tab of the chart customizer. When adding an axis, you get a number of options such as the type (numeric or logarithmic), label, color, autorange vs fixed range, and auto-ticks vs fixed ticks. You can also modify the position of the axis, but note that by default the Chart's Auto Axis Positioning property is enabled, which means that the chart will balance the axes automatically between left and right depending on demand. As pens are turned on and off by the user, only the axes that are used by visible pens are shown.

After you add your axes, you edit any pens that you want to use your new axes. Simply choose the new axis in the axis dropdown of the pen editing window.

Subplots

The Subplots feature lets you break up the chart's plot area into multiple distinct subplots that share the X axis, but have their own Y axes. This is often useful for digital data, as shown in the screenshot above. By default the chart has 1 subplot (the main plot). To add a new subplot, simply hit the add button in the Subplots tab of the chart customizer.

Subplots have relatively few options. The Weight option determines how much room the subplot gets relative to the other subplots. For example, in the screenshot above subplot #1's weight is 5, and subplot #2's weight is 1, leading to a 5-to-1 distribution of space. Just like axes, once you add your subplots you should go back to your pens and modify you pens' subplot property for any pens you want to appear on the subplot.

Pen Groups

You can put your pens in groups to break up the pens into some logical separation. For instance, in the screenshot above there are three pen groups: C1, C2, and Valves. The group name is used as the titled border for the pens' grouping container. Groups also have another purpose, but it is more advanced and most people won't have to worry about it. For more, read the Dynamic Pens section below.

Advanced Configuration

Dynamic Pens

In is often the case that you'll want to make one chart window that services many similar pieces of equipment. For instance, if you have 30 tanks and they all have the same datapoints, you want to be able to use one window for all 30 of them and simply pass the tank number into the chart window as a parameter. There are actually a number of ways to accomplish this, each method suitable for different scenarios.

Database pens have 2 ways to be made dynamic. The first is the Chart's Where Clause property. This is a snippet of SQL where clause syntax, like "machine_num = 28" that will be included for alldatabase pens in their queries. The second is to use a dynamic group. Any group can be made a dynamic group in the customizer. For each dynamic group, the easy chart will get a special dynamic property associated with that group. That property is another snippet of SQL where clause that will be applied to all database pens in that group.

The other way to make your pens (and anything else about the chart) dynamic at runtime is to use dynamic configuration. Read on...

Dynamic Configuration

The Easy Chart is not just meant to be easy to configure, but also very powerful. In particular, there is an emphasis on the ability to make any configuration change dynamically in a client - not just statically in the Designer. While a bit of scripting or clever property binding may be required, the technique is very powerful. This is achieved by storing all of the settings that you alter in the customizer in a set of expert-level dataset properties. So altering the datasets alters the chart configuration. You can inspect these various datasets, which hold the pens, axes, and subplot information, to see their format. They all look up information by column name (case-insensitive). So, if you have pen configuration stored in a database, you can bind an indirect SQL Query binding to alter the chart's pen set at runtime.

Properties

Name	Description	Property Type	Scripting	Category
3D X Offset	The offset to use in the x direction for the '3D Line' pen style.	int	.xOffset3D	Pen Style Options
3D Y Offset	The offset to use in the y direction for the '3D Line' pen style.	int	.yOffset3D	Pen Style Options
Allow Color Changes	If true, pen colors can be set to different values.	boolean	.allowColorChanges	Behavior
Allow Tag History Interpolation	If enabled and the query mode is not raw, the data will be interpolated for time spans with no data available.	boolean	tagHistoryAllowInterpolation	Tag History
Auto Apply	If true, user changes to pen visibility will occur immediately.	boolean	.autoApply	Behavior
Auto Axis Positioning	If true, axes alternate automatically between left and right, rather than being placed explicitly.	boolean	.autoPositionAxes	Behavior
Auto Color List	The list of colors to use if auto pen coloring is enabled.	Color[]	.autoColorList	Behavior
Auto Pen Coloring	If true, pens are assigned different colors automatically.	boolean	.autoColorPens	Behavior
Axes	This Dataset defines all axes that can be used by the pens.	Dataset	.axes	Chart Configuration
Axis Font	The font for axis labels.	Font	.axisLabelFont	Appearance
Background Color	The background color of the component. See Color Selector.	Color	.background	Appearance
Bar Margin	The margin to use for the 'Bar' pen style.	double	.barMargin	Pen Style Options
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. Note: The border is unaffected by rotation.	Border	.border	Appearance
Box Fill	For historical-mode date range. The fill color for the selection box. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.boxFill	Historical Range
Button Size	The size of the utility button icons.	int	.utilityButtonSize	Utility Buttons
Bypass Tag History Cache	If true, tag history queries will not use the client history cache.	boolean	.tagHistoryBypassCache	Tag History
Calculated Pens	This Dataset defines the calculated pens for the chart.	Dataset	.calcPens	Chart Configuration

Chart Border	The border for the chart itself.	Border	.chartBorder	Appearance
Chart Mode	Affects the mode that the chart operates in; Manual Mode, Historical Mode, Realtime Mode. **Integer Value** / **Corresponding Mode** 0 — Manual 1 — Historical 2 — Realtime	int	.chartMode	Behavior
Chart Title	Sets an optional title to be displayed above the chart.	String	.title	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
DB Pens	This Dataset defines all of the database pens for the chart.	Dataset	.pens	Chart Configuration
Date Editor Background	The background color for the date editor. See Color Selector.	Color	.editorBackgroundColor	Appearance
Date Editor Foreground	The foreground color for the date editor. See Color Selector.	Color	.editorForegroundColor	Appearance
Date Range	Affects the position of the date range control.	int	.dateRangeLocation	Layout
Date Range Border	The border for the date range control, if visible.	Border	.dateRangeBorder	Appearance
Date Style	The style to display dates in. For international support.	int	.dateStyle	Historical Range
Digital Gap	The size of the gap to use between digital pens.	double	.digitalGap	Pen Style Options
Empty Group Name	The group name to use for pens that are not in a pen group.	String	.emptyGroupName	Behavior
End Date	For manual-mode. The end date to use for selecting pen data	Date	.endDate	Data
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See Color Selector.	Color	.foreground	Appearance
Gap Threshold	The relative threshold to use for determining continuity breaks for the 'Discontinous Line' pen style.	double	.gapThreshold	Pen Style Options
Gridline Color	The color of the gridlines. See Color Selector.	Color	.gridlineColor	Appearance
Gridline Dash Pattern	Enter a string of comma-delimited numbers which indicate the stroke pattern for a dashed line. For instance, "3,5" means three pixels on, five pixels off.	String	.gridlineDashPattern	Appearance

Gridline Width	The width (thickness) of the gridlines.	float	.gridlineWidth	Appearance
Group Pens	If true, pens will be grouped by their group name.	boolean	.penGrouping	Behavior
High Density Color	For historical-mode date range. The color used to indicate high data density. See Color Selector.	Color	.highDensityColor	Historical Range
Horiz Gap	The horizontal spacing to use for the pen checkboxes.	int	.hGap	Layout
Ignore Bad Quality Data	If true, causes the system to ignore any bad quality data.	boolean	tagHistoryIgnoreBadData	Tag History
Invert Time Axis	If true, the time axis values will increase from the right to left or from top to bottom depending on the Plot Orientation.	boolean	.invertTimeAxis	Layout
Legend	Where the legend should appear, if any.	int	.legend	Layout
Max Selection	For historical-mode date range. The maximum size of the selected date range.	String	.maxSelectionSize	Historical Range
Maximize Plot	If true, displays maximized plot.	boolean	.currentlyMaximized	Layout
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Outer Range End	For historical-mode date range. The end date for the outer range.	Date	.outerRangeEnd	Historical Range
Outer Range Start	For historical-mode date range. The start date for the outer range.	Date	.outerRangeStart	Historical Range
Pen Control Border	The border for the pen control panel, if visible.	Border	.penBorder	Appearance
Pen Control Mode	The style in which the pen control panel alters the chart configuration. In heavyweight mode, unchecked pens are not queried, so checking and unchecking pens refreshes the chart. In lightweight mode, all pens are constantly queried, so checking and unchecking pens is quick.	int	.penControlMode	Behavior
Pen Control?	Controls whether or not end-users can turn on and off pens.	boolean	.allowPenManipulation	Behavior
Plot Background	The background color for all plots, unless they override it. See Color Selector.	Color	.plotBackground	Appearance
Plot Orientation	The plot orientation for all plots.	int	.plotOrientation	Layout
Plot Outline	The color to use for the plot outline. See Color Selector.	Color	.plotOutlineColor	Appearance
Poll Rate	The rate (in milliseconds) at which this chart's queries poll. Historical charts don't use this property.	int	.pollRate	Behavior

Properties Loading	The number of properties currently being loaded. (Read only. Usable in bindings and scripting.)	int	.propertiesLoading	Uncategorized
Realtime Text	For realtime-mode date range. The text to display on the realtime date control.	String	.rtLabel	Realtime Range
Selected X Value	The selected domain axis value for X-Trace and Mark modes. (Read only. Usable in bindings and scripting.)	String	.selectedXValue	Uncategorized
Selection Highlight	For historical-mode date range. The focus highlight color for the selection box. See Color Selector.	Color	.selectionHighlight	Historical Range
Show Density	For historical-mode date range. If true, a data density histogram will be shown in the date range.	boolean	.showHistogram	Historical Range
Show Loading	If true, an animated indicator will be shown when data is loading.	boolean	.showLoading	Behavior
Show Maximize Button?	If true, a small maximize button will be displayed next to the chart.	boolean	.showMaximize	Utility Buttons
Show Popup?	If true, a popup menu will be shown on right-click that allows the user to change mode, print, save, etc.	boolean	.showPopup	Behavior
Show Print Button?	If true, a small print button will be displayed next to the chart..	boolean	.showPrint	Utility Buttons
Show Save Button?	If true, a small save button will be displayed next to the chart.	boolean	.showSave	Utility Buttons
Show Tooltips?	If true, tooltips showing point values will be displayed on the chart.	boolean	.tooltips	Behavior
Show Warnings	If true, warnings generated during chart configuration will be printed to the console.	boolean	.showWarnings	Behavior
Sort Pens	If true, pens visibility checkboxes will be sorted.	boolean	.alphabetizePens	Layout
Start Date	For manual-mode. The start date to use for selecting pen data.	Date	.startDate	Data
Startup Range	For historical-mode date range. If startup mode is Automatic, this will be the starting range of time available for selection.	String	.startupRange	Historical Range
Startup Selection	For historical-mode date range. If startup mode is Automatic, this will be the starting selected range.	String	.startupSelection	Historical Range
Subplot Gap	The gap between subplots.	double	.subplotGap	Layout
Subplots	This Dataset defines all subplots' relative size and color.	Dataset	.subplots	Chart Configuration
Tag History Resolution	The number of datapoints to request for tag history pens. -1 means raw data, 0 means automatic, which uses the width of the chart.	int	.tagHistoryResolution	Tag History

Tag History Resolution Mode	The mode used for the number of requested points. Fixed will use the Tag History Resolution Size, Natural will return a value per scanclass execution, Chart Width will be based on the actual width of the chart component, and Raw will be the raw data.	int	tagHistoryResolutionMode	Tag History
Tag Pens	This Dataset defines all of the Tag History pens for the chart.	Dataset	.tagPens	Chart Configuration
Tick Density	For historical-mode date range. This is multiplied by the width to determine the current ideal tick unit.	float	.tickDensity	Historical Range
Tick Font	The font for tick labels.	Font	.axisTickLabelFont	Appearance
Time Style	The style to display times of day. For international support.	int	.timeStyle	Historical Range
Title Font	The font for the optional chart title.	Font	.titleFont	Appearance
Today Color	For historical-mode date range. The color of the "Today Arrow" indicator. See Color Selector.	Color	.todayIndicatorColor	Historical Range
Total Datapoints	The number of datapoints being displayed by the graph. (Read only. Usable in bindings and scripting.)	int	.datapoints	Uncategorized
Track Margin	For historical-mode date range. The amount of room on either side of the slider track. May need adjusting of default font is changed.	int	.trackMargin	Historical Range
Unit	For realtime-mode date range. The selected unit of the realtime date control.	int	.unit	Realtime Range
Unit Count	For realtime-mode date range. The number of units back to display.	int	.unitCount	Realtime Range
Validate Scan Class Executions	Causes the tag history query to verify the scan class execution records, generating bad data for the time periods where the scanclasses did not execute.	boolean	tagHistoryValidateScanclass	Tag History
Vert Gap	The vertical spacing to use for the pen checkboxes.	int	.vGap	Layout
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Where Clause	A snippet of where clause that will be applied to all pens, like "TankNum = 2".	String	.globalWhereClause	Data
X Axis AutoRange?	If true, the X axis will automatically fit the range of available data, if false, it will display a fixed range based on the start date and end date.	boolean	.xAxisAutoRange	Behavior
X Axis Label	The label shown on the X Axis (time axis).	String	.xAxisLabel	Appearance
X Axis Margin	A margin for the upper and lower ends of the x axis, expressed as a percentage of the total range.	double	.xAxisMargin	Behavior
X Axis Visible	Should the x-axis be displayed?	boolean	.xAxisVisible	Appearance
X-Trace Large Number Format	The large decimal format for the x-trace value in the Easy Chart.	String	.xTraceLargeNumberFormat	Appearance

X-Trace Number Format Threshold	If the magnitude of the to-be-formatted value is below this threshold, then the X-Trace Small Number Format will be used.	double	.xTraceNumberFormatThreshold	Appearance
X-Trace Small Number Format	The small decimal format for the x-trace value in the Easy Chart.	String	.xTraceSmallNumberFormat	Appearance
X-Trace Track Mouse	If set enabled, and the chart is set to X-Trace mode, the X-Trace will auto track the mouse position while the cursor is over the component. This is particularly useful when displaying the Easy Chart on a touchscreen.	boolean	xTraceTrackMouse	Appearance

Scripting

Scripting Functions

- Description

  This function save the chart's datasets as an Excel file. Returns a String of the complete file path chosen by the user, or None if the user canceled the save.

- Parameters

  String filename - The default file name for the Save dialog.

- Return

  String

- Description

  This function will print the chart.

- Parameters

  Nothing

- Return

  Nothing

- Description

  Sets the current mode for the chart.

- Parameters

  Int mode - The mode to set the chart to. The mode options are as follows:

  0 : Zoom Mode. This is the default mode, where the user can draw a zoom rectangle with the mouse pointer.

  1 : Pan Mode. This mode lets the user use the mouse pointer to pan the chart to the left and right.

  3 : Mark mode. This mode lets the user click near a datapoint to annotate the point with its X and Y value.

  4 : X-Trace mode. This mode lets the user click and drag on the chart to see all values that fall along that X value.

- Return

  Nothing

- Description

  returns an Array List of datasets, representing the time series data of each type of pen.

- Parameters

  None

- Return

  Array List of datasets. Each dataset represents timeseries data for set of pens. The order of the datasets are listed below.

- Index order of datasets

Index	Dataset
0	Tag Pens
1	Calculated Pens
2	Database Pens

Python - Accessing the Tag Pens Dataset

```
# This example will extract the Tag Pens series data that is already
present in an Easy Chart, and pass it to a Power Table on the same window.
# This script could be placed on the Easy Chart's propertyChanged event.

# Filter on the name of the property
if event.propertyName == 'tagPens':

        # Wrap our dataset behavor in a function, so we can pass it to
system.util.invokeLater
        def func():
                chart = event.source


                # Extract the datasets
                datasets = chart.exportDatasets()


                # Pass the first dataset (index 0 contains data for Tag
Pens) to the Power Table
                event.source.parent.getComponent('Power Table').data =
datasets[0]

        # Using invokeLater to provide a delay. We want this to run after
the chart has finished loading the new tag.
        system.util.invokeLater(func, 1000)
```

Extension Functions

- Description

  Provides an opportunity to perform further chart configuration via scripting. Doesn't return anything.

- Parameters

  Component self - A reference to the component that is invoking this function.

  JFreeChart chart- A JFreeChart object. Refer to the JFreeChart documentation for API details.

- Return

  Nothing

- Description

  Provides an opportunity to configure the x-trace label. Return a string to override the default label.
- Parameters

  Component self - A reference to the component that is invoking this function.

  JFreeChart chart - A JFreeChart object. Refer to the JFreeChart documentation for API details.

  String penName - The name of the pen the x-trace label applies to.

  int yValue - The y-value of the pen at the x-trace location.
- Return

  Nothing
- Description

  Called when the user has dropped rows from a power table on the chart. The source table must have dragging enabled.
- Parameters

  Component self - A reference to the component that is invoking this function.

  Component sourceTable - A reference to the table that the rows were dragged from.

  List rows - An array of the row indicies that were dragged, in the order they were selected.

  Dataset rowData - A dataset containing the rows that were dragged.
- Return

  Nothing

- Description

  Called when the user has dropped tags from the tag tree onto the chart. Normally, the chart will add pens automatically when tags are dropped, but this default behavior will be suppressed if this extension function is implemented.

- Parameters

  Component self - A reference to the component that is invoking this function.

  List paths - A list of the tag paths that were dropped on the chart.

- Return

  Nothing

Example - Pen Name Replacement

```
#This will take a tag that gets dropped from a Tag Browse Tree set in Realtime
Tag Tree mode,
#and will replace the underscores in the name of the tag "_" and replace them
with spaces.
tagPens = self.tagPens

for tag in paths:
        tagPath = tag.replace("default", "~")
        splitTag = tag.split("/")
        name = splitTag[-1].replace("_", " ")

        newRow = [name, tagPath, "MinMax", "Default Axis", 1, 1, system.gui.color
(255, 85, 85, 255), "", 1, 1, 0, 1, 0, "", 0, 0, 0, 1, 0, 0]

        self.tagPens = system.dataset.addRow(tagPens, newRow)
```

Example - Group Name

```
#This will take a tag that gets dropped from a Tag Browse Tree set in Realtime
Tag Tree mode,
#and will place it into a Pen Group titled "My Group Name".

tagPens = self.tagPens
groupName = "My Group Name"
for tag in paths:

        newRow = [name, tagPath, "MinMax", "Default Axis", 1, 1, system.gui.color
(255, 85, 85, 255), "", 1, 1, 0, 1, 0, "groupName", 0, 0, 0, 1, 0, 0]

        self.tagPens = system.dataset.addRow(tagPens, newRow)
```

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

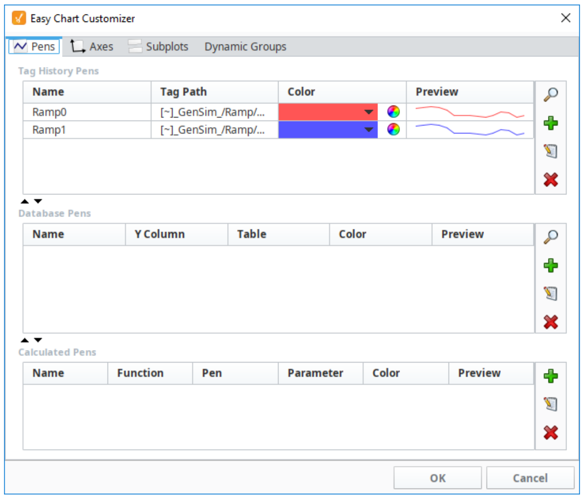
Refer to the Vision - Easy Chart Customizer and the Using the Vision Easy Chart sections of the manual for examples and tutorials on how to use the Easy Chart Customizer. With the customizer, you can set up:

- Axes
- Subplots
- Pen Groups
- Pen Display
- Offsets
- Calculated Pens
- Ad-Hoc Charting
- Indirection

Examples

There are no examples associated with this component.

Vision - Easy Chart Customizer


Easy Chart Customizer

Pens | Axes | Subplots | Dynamic Groups

Tag History Pens

Name	Tag Path	Color	Preview
Ramp0	[~]_GenSim_/Ramp/...	🟥	
Ramp1	[~]_GenSim_/Ramp/...	🟦	

Database Pens

Name	Y Column	Table	Color	Preview

Calculated Pens

Name	Function	Pen	Parameter	Color	Preview

OK | Cancel

Description

The Easy Chart component allows you to display the history of your Tags on a chart. When you drag and drop Tags onto an Easy Chart, it automatically trends the data for you. It has a special customizer that has some default settings to help you get started.

Customizers

The Easy Chart Customizer allows you to easily modify the chart to your own style. You can add pens and modify the contents of your pens, and create new axes, subplots, and pen groups. When you open the customizer, you'll notice four tabs at the top of the window: Pens, Axes, Subplots, and Dynamic Groups. Each have their own properties.

Shown below is each tab in the Easy Chart Customizer listing all its properties along with a brief description.

The Pens tab is where you can add new pens, create custom names for your pens, and group pens. There are three types of pens, and each functions in a similar manner, but what makes them different is how their data is collected. Each pen type has a few unique properties and is listed at the bottom of the table.

- **Tag pens** - Pens are driven by the Tag history system. (Data from any historical provider can be used).
- **Database pens** - Pens that are driven by an SQL query. They can query for data in any connected SQL database.
- **Calculated pens** - Pens that derive their data from calculations performed on other pens.

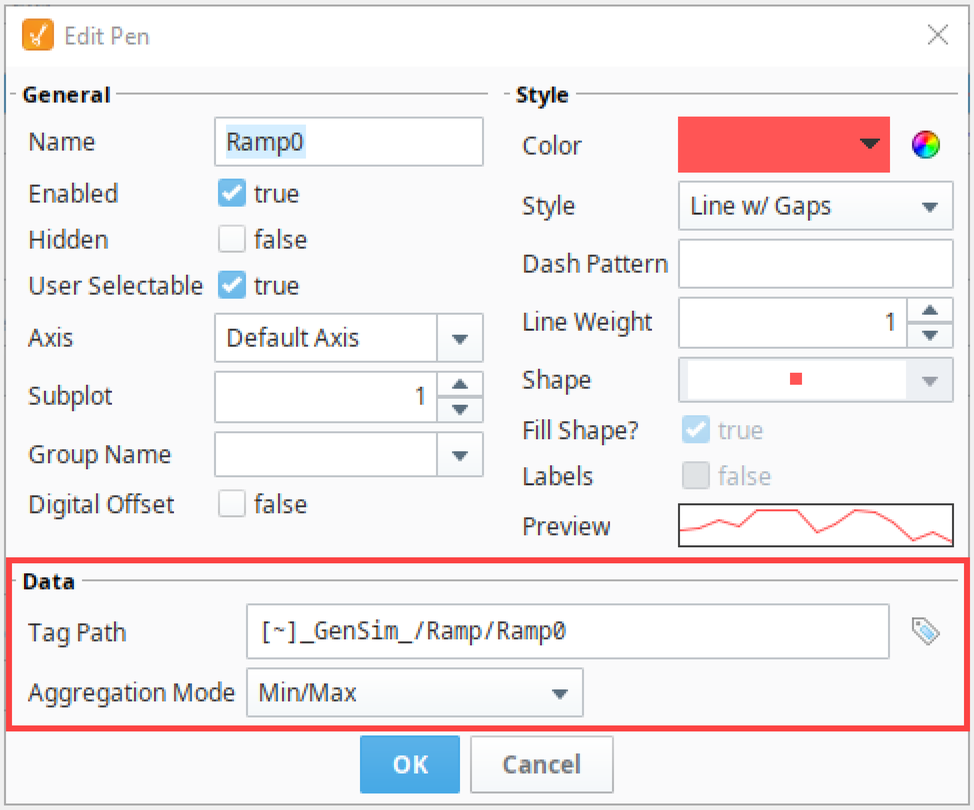
Action	Description
🔍	Add pen (Browse for Tags).
➕	Add a pen manually.
✎	Edit pen.
❌	Delete pen.

Property	Description
Edit Pen Panel	
Name	The name of the pen is what the user will see in the legend and the pen panel.
Enabled	If false, this pen will not show up on the chart and the data will not be generated. The user will be able to enable it via the pen control panel.

Hidden	If true, the pen will not show up on the chart or the pen control panel. The data will be generated.
User Selectable	If false, the pen will show up on the chart, but not the pen control panel.
Axis	Select the Y axis this pen will use.
Subplot	Putting pens on separate subplots can increase chart clarity.
Group Name	The group name is used for logical grouping in the pen panel and for advanced dynamic grouping.
Digital Offset	If true, a small gap will be placed between this and other digital pens so they don't overlap each other.
Color	Pen color.
Style	The style of the pen determines how it looks in the chart.
Dash Pattern	Uses a dash pattern like "5,5" to specify 5 pixels on, 5 pixels off.
Line Weight	The thickness of the pen's line.
Shape	If the renderer style uses shapes, this will be the shape for each point.
Fill Shape	If true, the shape will be filled in rather than an outline.
Labels	If true, shows a label of the value above each bar.
Preview	Field where you can view the pen style.
Tag History Pens Properties	
Tag Path	String-based path where the Tag is located.
Aggregation Mode	Type of calculation (i.e., Constant, UCL, UWL, Avg, LWL, LCL, MovingAvg, Multiply, Min, Max).
Database Pens Properties	
Volume Column	The name of the column for the pen's value (Y value).
Time Column	The name of the column for the pen's timestamp (X value).
Table Name	The name of the table where the pen will be found.
Datasource	The name of the datasource to use for this pen (MySQL).
Where Clause	You can specify a snippet of WHERE clause here, like "TankNum = 16."
Run Diagnostics	Test this pen for data configuration for validity.
Calculated Pens Properties	
Function	Function is the type of calculation (i.e., Constant, UCL, UWL, Avg, LWL, LCL, MovingAvg, Multiply, Min, Max).
Driving Pen	Dedicated pen that will drive the value.

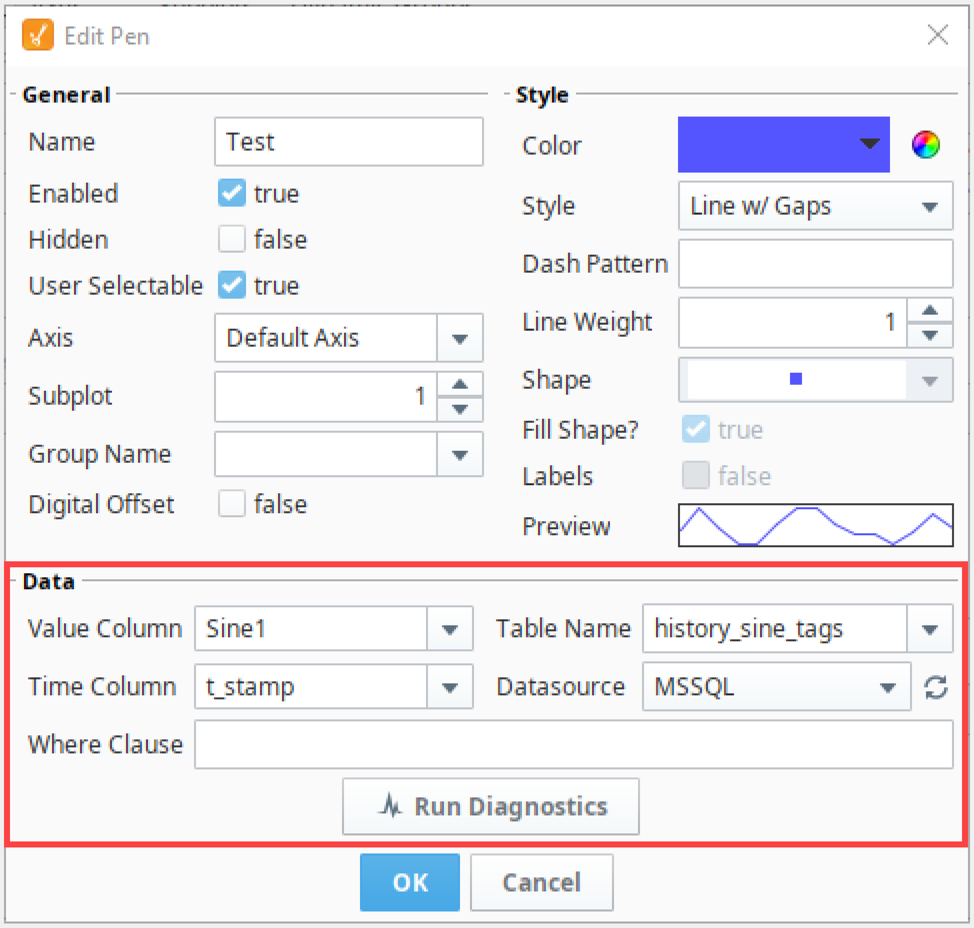
Parameter	Value which is the horizontal line drawn on the graph. The parameter type can be different for the Function used: • Constant Value - constant value of the pen.(Used with the Constant function). • Window Size - the size of the moving average window, specified as a multiplier of the chart's date range. It's the percentage of time that you're going to do the moving average on. (Used with MovingAvg function). • Factor - multiply by 'X' factor (Used with Multiply function). • Secondary pen - another pen added to the chart to show the sum and/or the difference. (Used with the Sum and Difference functions).

Edit Pen Panel for Tag History Pens



Name: Ramp0
Enabled: true
Hidden: false
User Selectable: true
Axis: Default Axis
Subplot: 1
Group Name:
Digital Offset: false
Color:
Style: Line w/ Gaps
Dash Pattern:
Line Weight: 1
Shape:
Fill Shape?: true
Labels: false
Tag Path: [~]_GenSim_/Ramp/Ramp0
Aggregation Mode: Min/Max

Edit Pen Panel for Database Pens


Edit Pen

General

Name	Test
Enabled	✓ true
Hidden	false
User Selectable	✓ true
Axis	Default Axis
Subplot	1
Group Name	
Digital Offset	false

Style

Color	
Style	Line w/ Gaps
Dash Pattern	
Line Weight	1
Shape	
Fill Shape?	✓ true
Labels	false
Preview	

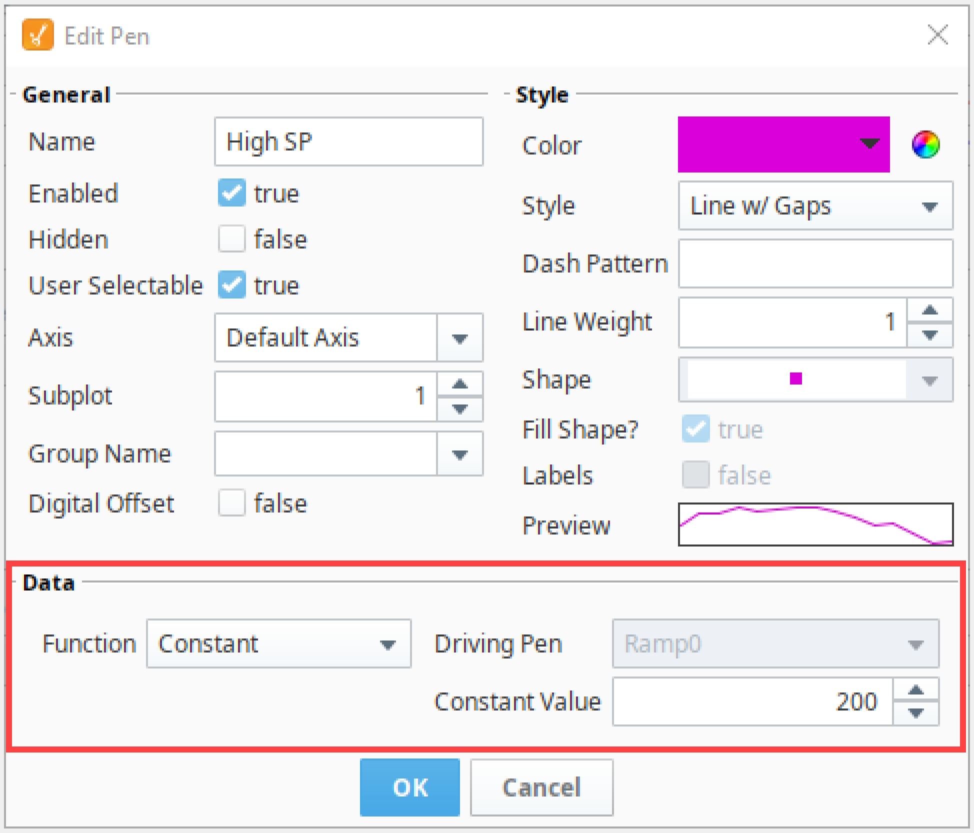
Data

Value Column	Sine1	Table Name	history_sine_tags
Time Column	t_stamp	Datasource	MSSQL
Where Clause			

Run Diagnostics

OK Cancel

Edit Pen Panel for Calculated Pens


Edit Pen

General

Name	High SP
Enabled	✓ true
Hidden	false
User Selectable	✓ true
Axis	Default Axis
Subplot	1
Group Name	
Digital Offset	false

Style

Color	
Style	Line w/ Gaps
Dash Pattern	
Line Weight	1
Shape	
Fill Shape?	✓ true
Labels	false
Preview	

Data

Function	Constant	Driving Pen	Ramp0
		Constant Value	200

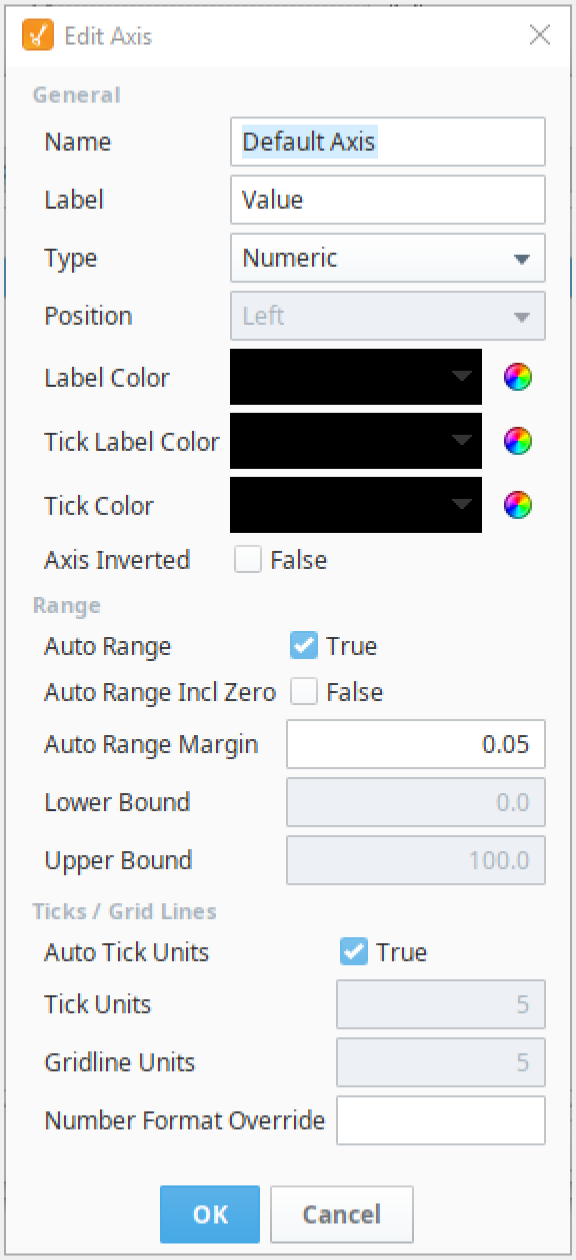
OK Cancel

For more information, refer to the following sections:

- Easy Chart - Pen Names and Groups
- Easy Chart - Calculated Pens
- Using the Vision Easy Chart

The Axes tab is where you can configure multiple axes on the Easy Chart component.

Property	**Description**
Name	The name of the axis is what pens use to refer to it.
Label	The label will be displayed on the chart next to the axis.
Type	The type of axis determines the plotting behavior. (i.e., Numeric, Logarithmic, Symbol)
Position	The position of the axis, if automatic, axis positioning is turned off.
Label Color	Color of the label.
Tick Label Color	Color of the tick label.
Tick Color	Color of the tick mark.
Axis Inverted	If true, inverts the axis.
Auto Range	If true, the axis will automatically scale itself to the data, rather than display a fixed range.
Auto Range Incl Zero	If true, forces the auto range to include zero.
Auto Range Margin	The extra margin (as percent of the total range) for the top and bottom of an auto range axis.
Lower Bound	The lower bound of a non-auto-ranging axis.
Upper Bound	The upper bound of a non-auto-ranging axis.
Auto Tick Units	If true, the distance between the tick marks and the gridlines will be automatically calculated rather than a fixed number.
Tick Units	If false, this amount will be used as the distance between tick marks.
Gridline Units	If false, this amount will be used as the distance between gridlines.
Number Format Override	Specifies a number format pattern to use for tick labels. Leave blank for automatic number formatting.


Edit Axis

General

Name	Default Axis
Label	Value
Type	Numeric
Position	Left
Label Color	⬛
Tick Label Color	⬛
Tick Color	⬛
Axis Inverted	☐ False

Range

Auto Range	✓ True
Auto Range Incl Zero	☐ False
Auto Range Margin	0.05
Lower Bound	0.0
Upper Bound	100.0

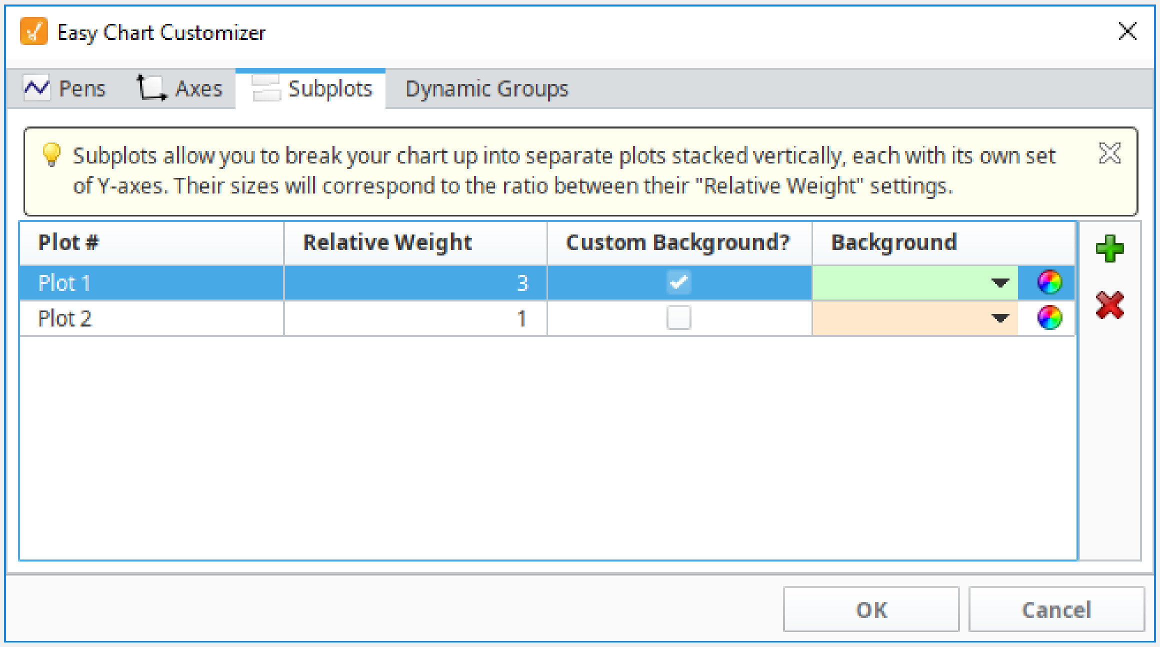
Ticks / Grid Lines

Auto Tick Units	✓ True
Tick Units	5
Gridline Units	5
Number Format Override	

For more information, refer to the Easy Chart - Axes.

The Subplot tab is where you can break up a chart's plot area into multiple distinct subplots that share the X axis, and also where you can add additional subplots.

Property	Description
Plot Number	Number of plots in a chart plot area.
Relative Weight	Ratio between all subplots. (If you have two subplots, and Plot 1's weight is 3 and Plot 2's weight is 1, then Plot 1 will be 3 times larger than Plot 2).
Custom Background	If false, the default background is white.
Background	Color of the plot area's background.



Subplots allow you to break your chart up into separate plots stacked vertically, each with its own set of Y-axes. Their sizes will correspond to the ratio between their "Relative Weight" settings.

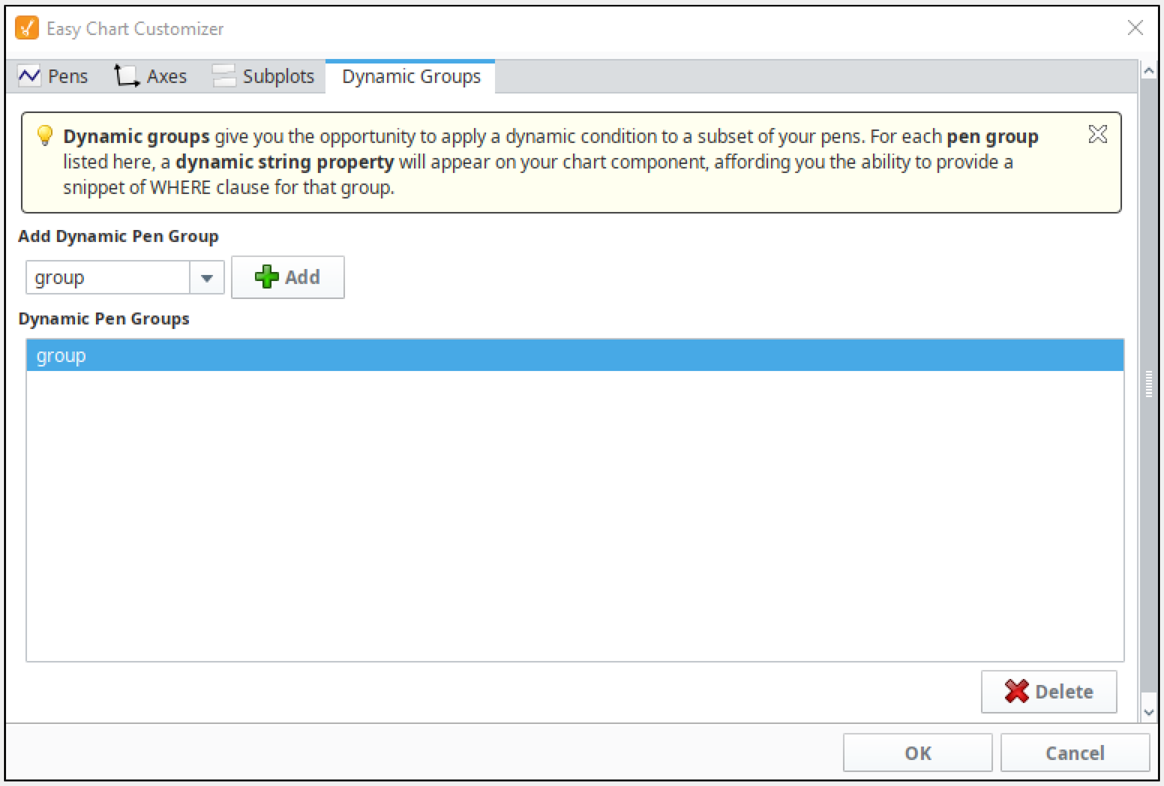
Plot #	Relative Weight	Custom Background?	Background
Plot 1	3	✓	🟩
Plot 2	1	☐	🟧

In the Pens Tab

Once you add a subplot, go to the Pens Tab, edit your pen, and put your pen into a different subplot.

For more information, go to Easy Chart - Subplots.

Dynamic Groups are used with Database pens. They allow you to apply a dynamic condition, like using a WHERE clause, to a subset of pens. For each pen group, a dynamic string property will appear in the Property Editor under Custom Properties of your Easy Chart component. You can create a WHERE Clause that will search the database and return values if the pens meet a true condition.



Dynamic groups give you the opportunity to apply a dynamic condition to a subset of your pens. For each **pen group** listed here, a **dynamic string property** will appear on your chart component, affording you the ability to provide a snippet of WHERE clause for that group.

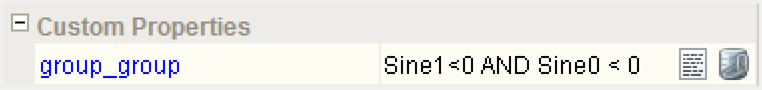
Add Dynamic Pen Group

group

Dynamic Pen Groups

group

Property Editor - Custom Properties - Where Clause for Dynamic Group Property



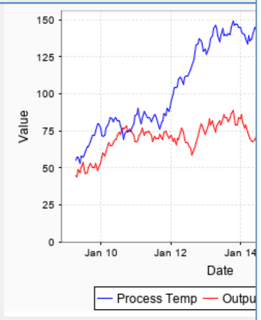
Custom Properties

group_group	Sine1 <0 AND Sine0 < 0

To learn more about Dynamic Groups, refer to the Vision - Easy Chart section.

Vision - Chart

General



Component Palette Icon:

Chart

Description

The Chart component (also called the Classic Chart when contrasted with the Easy Chart) provides a flexible way to display either timeseries or X-Y charts that are powered by any number of datasets. Typically, these datasets are bound to SQL Query Bindings in Vision.

Features

- SQL Query and/or SQLTags Historian data sources
- Zoom, Pan, X-Trace modes
- Any number of Y-axes and subplots
- Realtime or Historical
- Many different rendering styles

Configuration

The basic idea behind configuring the classic chart is simple: add datasets, and fill them in with data in a format that the chart understands. You can add datasets to the chart using the chart's customizer. You then use standard property bindings to put data into these charts. Commonly you'll use a SQL Query Bindings in Vision. Since these datasets are just normal dynamic properties, you can also access them via scripting.

The Customizer also lets you add additional X and Y axes. There are various types of axes, and they each have a large number of properties. Lastly, you can configure additional properties for each dataset, such as which axes it maps to, its visual style, subplot, etc.

Datasets

Each dataset should define one or more "series" (a.k.a "pens"). The format for these datasets is quite simple. Each series in a dataset shares common X-values, defined by the first column. Each additional column are the Y-values for a series.

Binding Techniques

The classic chart can be used to make almost any kind of chart, with some effort. Historical, realtime, dynamic pen selection, etc., is all possible. Your job is just to fill the datasets with the pertinent data, and the chart will display it. The most common idea is to make the chart dynamic by varying the date range that the dataset's SQL Query bindings run. This is easy to do by adding a Date Range component and using Indirect Tag Binding.

Chart Type: XY vs Category

The classic chart is typically in XY Plot mode. This means that the X-axis is either date or numeric, and the Y-axes are numeric. If your X-axis is categorical (names, not numbers), you can switch the Chart Type property to Category Chart in the Property Editor. Don't be surprised when you get a few errors - you'll need to go and switch your X-axis to be a Category Axis, and fill your dataset in with valid category data, that is, String-based X-values. This is most often used with the Bar Renderer (see the Vision - Chart Customizer).

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is not affected by rotation.	Border	.border	Common
Chart Orientation	The orientation of the domain axis of the chart.	int	.orientation	Appearance
Chart Title	An optional title that will appear at the top of the chart.	String	.title	Appearance
Chart Type	Choose the type for this chart: XY (Numeric X-axis) or Category (String X-axis).	int	.chartType	Behavior
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Extract Order	Extract order for how category datasets should be interpreted.	int	.extractOrder	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Plot Background	The background color for all plots, unless they override it.	Color	.plotBackground	Appearance
Properties Loading	The number of properties currently being loaded. (Read only. Usable in bindings and scripting.)	int	.propertiesLoading	Uncategorized
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Selected Datapoint	The currently selected datapoint. (Read only. Usable in bindings and scripting.)	String	.selectedData	Uncategorized
Selected X Value	The selected domain axis value for X-Trace and Mark modes. (Read only. Usable in bindings and scripting.)	String	.selectedXValue	Uncategorized
Selection Enabled?	If true, the user will be able to select datapoints on the chart. The selected datapoint will be highlighted, and the selectedData property will reflect it.	boolean	.selectionEnabled	Behavior
Selection Highlight Color	The color of the selection highlight.	Color	.selectionHighlightColor	Appearance

Selection Highlight Width	The line width of the selection highlight.	float	.selectionHighlightWidth	Appearance
Show Legend?	If true, a legend will be shown for the series displayed in the chart.	boolean	.legend	Appearance
Show Popup?	If true, a popup menu will be shown on right-click that allows the user to change mode, print, save, etc.	boolean	.showPopup	Behavior
Show Tooltips?	If true, tooltips showing point values will be displayed.	boolean	.tooltips	Behavior
Subplot Mode	The axis that subplots share if more than one subplot.	int	.subplotMode	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

- Description

  Provides an opportunity to perform further chart configuration via scripting.
- Parameters

  Component self- A reference to the component that is invoking this function.

  JFreeChart chart- A JFreeChart object. Refer to the JFreeChart documentation for API details.
- Return

  Nothing
- Description

  Provides an opportunity to configure the x-trace label. Return a string to override the default label.
- Parameters

  Component self- A reference to the component that is invoking this function.

  JFreeChart chart **-** A JFreeChart object. Refer to the JFreeChart documentation for API details.

  String penName - The name of the pen the x-trace label applies to.

  int yValue - The y-value of the pen at the x-trace location
- Return

  Nothing

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

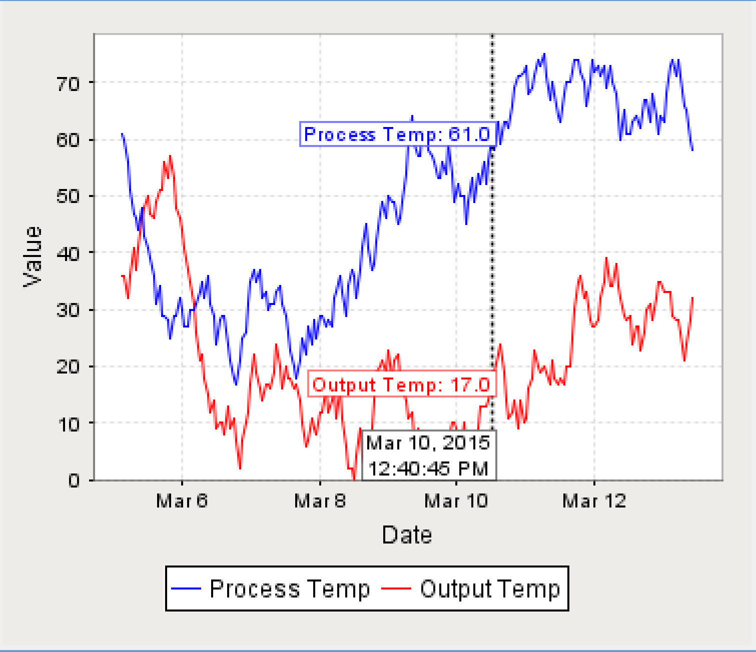
Customizers

The Chart component uses its own customizer called the Vision - Chart Customizer. You can add datasets and additional XY axes to a chart using the tabs in the chart customizer. You can configure additional properties for each dataset, like what axes it maps to as well as select from a host of visual styles. It also has six axis types to choose from, each with an extensive list of properties.

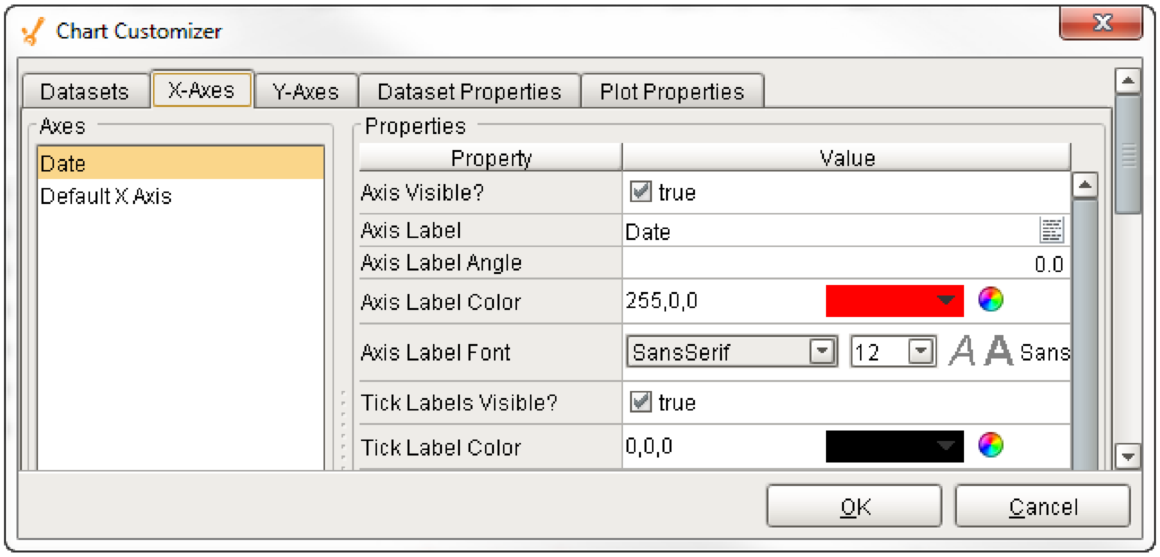
The customizer already has some default styles in place to help you get started, but you can modify these default settings to your own style. Refer to the Vision - Chart Customizer section for property descriptions and examples of chart axis types.

- Vision - Chart Customizer
- Vision Component Customizers
- Understanding Component Customizers

Example



Vision - Chart Customizer



Property	Value
Axis Visible?	✓ true
Axis Label	Date
Axis Label Angle	0.0
Axis Label Color	255,0,0 🟥
Axis Label Font	SansSerif 12
Tick Labels Visible?	✓ true
Tick Label Color	0,0,0 ■

Description

The Chart component, also known as the Classic Chart, can be used to make almost any kind of chart. It provides a flexible way to display XY charts using a host of built-in properties. All you need to do to create a chart is add datasets, fill them in with data, configure a property binding, and setup the chart properties using the customizer.

Customizer

The Chart component has its own special customizer called the Chart Customizer. When you open the customizer, you'll notice five tabs at the top: Dataset, X-Axes, Y-Axes, Dataset Properties, and Plot Properties. Each tab has its own set of properties and defaults.

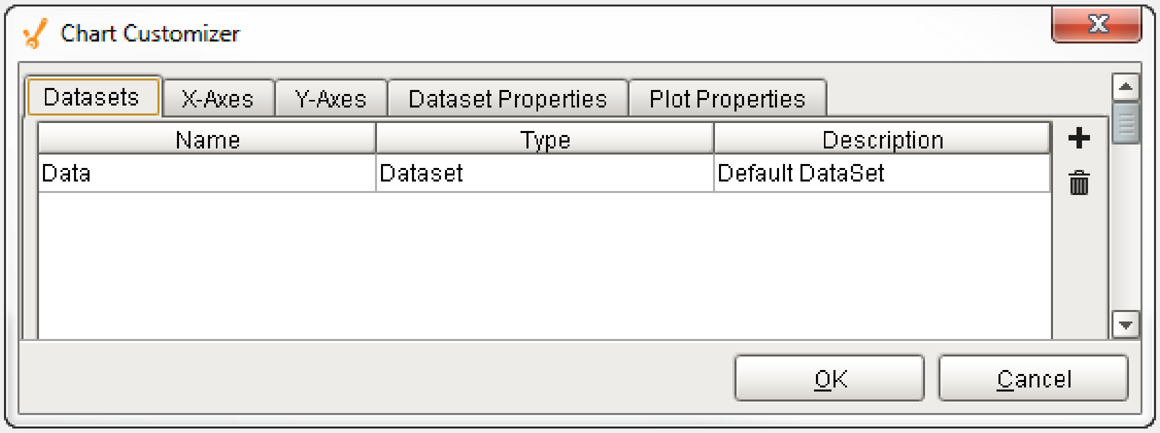
To get started, first add your dataset(s) and any additional XY axes using the appropriate tabs in the customizer. You can configure additional properties for each dataset, like what axes the data maps to, as well as select from a host of visual styles.

There are six types of axes to choose from when configuring a chart, each having its own list of properties: Number Axis, Date Axis, Category Axis, Logarithmic Axis, Elapsed Axis, and Symbols Axis. Most of the X and Y axes properties are used in the customizer, and some properties are specific to the axis type and have their own unique properties

The customizer already has some default styles in place to help you get started, but you can modify these default settings to your own style using the XY properties, Axes Type, Renderer and Plot styles. If you don't like one style, try another.

Shown below is each tab in the Chart Customizer with all its properties, description, and what axes type it supports. Note: Not all properties are available for all axes type charts.

The Dataset tab is where you setup, add, and remove datasets.



Name	Type	Description
Data	Dataset	Default DataSet

Property	Description
Data	Default dataset property.
✚ Add	Adds a new dataset. Click the plus icon a new row will be added. Enter the dataset Name and Description.
🗑 Delete	Deletes an existing dataset. Click the Delete icon to delete an existing dataset.
Name	Name of the dataset. Double click in the field to rename the dataset.
Type	Default type is "Dataset."
Description	Describes the dataset.

The X-Axes tab is where X-Axis properties are configured. You can also add and delete X axes here.

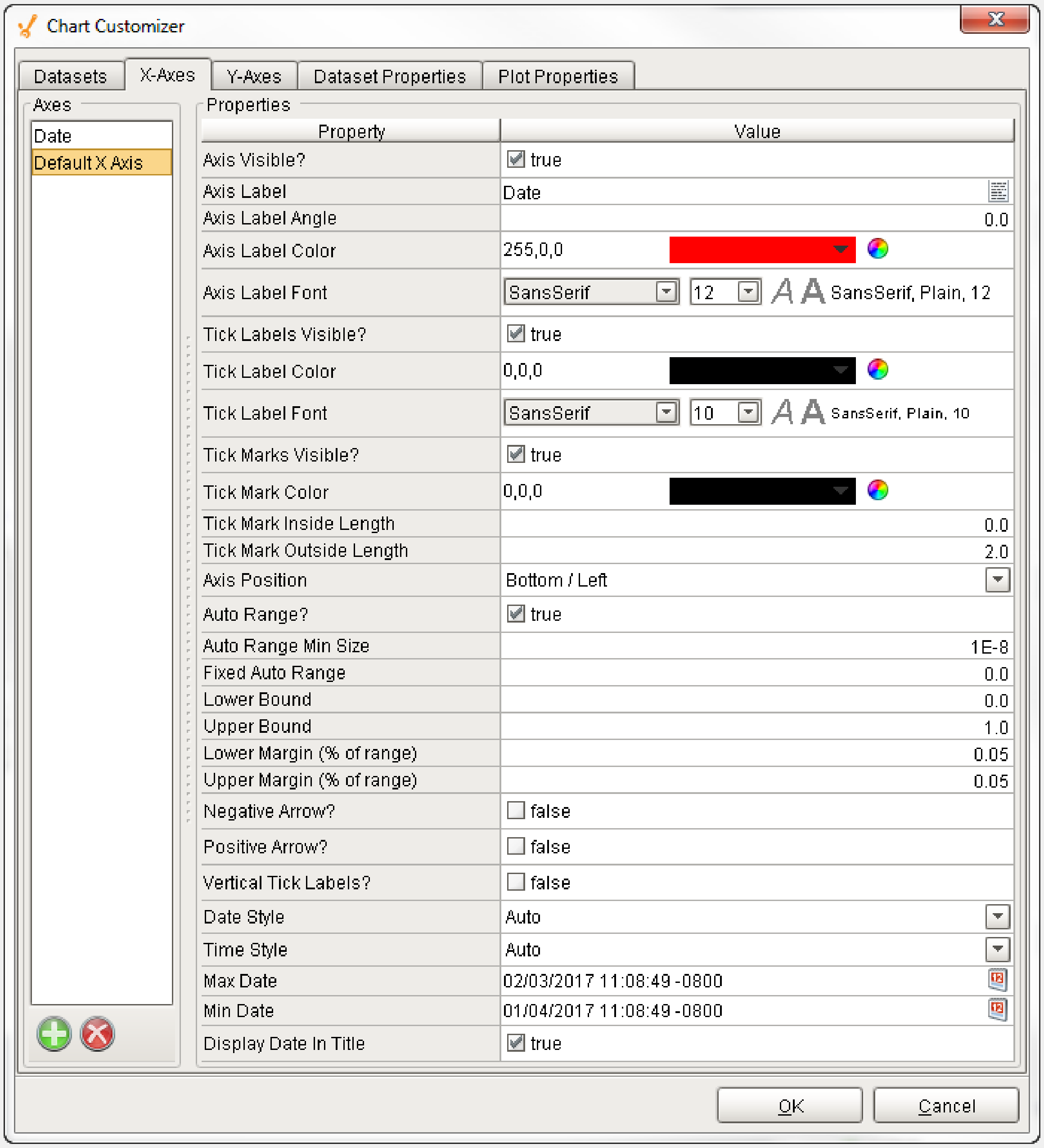


Chart Customizer

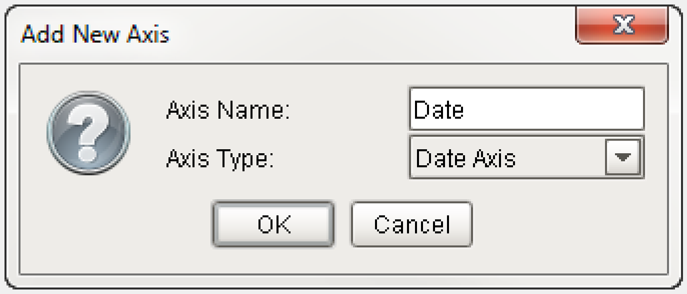
Datasets | X-Axes | Y-Axes | Dataset Properties | Plot Properties

Axes: Date, Default X Axis

Property	Value
Axis Visible?	✓ true
Axis Label	Date
Axis Label Angle	0.0
Axis Label Color	255,0,0 🟥
Axis Label Font	SansSerif 12 SansSerif, Plain, 12
Tick Labels Visible?	✓ true
Tick Label Color	0,0,0 ⬛
Tick Label Font	SansSerif 10 SansSerif, Plain, 10
Tick Marks Visible?	✓ true
Tick Mark Color	0,0,0 ⬛
Tick Mark Inside Length	0.0
Tick Mark Outside Length	2.0
Axis Position	Bottom / Left
Auto Range?	✓ true
Auto Range Min Size	1E-8
Fixed Auto Range	0.0
Lower Bound	0.0
Upper Bound	1.0
Lower Margin (% of range)	0.05
Upper Margin (% of range)	0.05
Negative Arrow?	☐ false
Positive Arrow?	☐ false
Vertical Tick Labels?	☐ false
Date Style	Auto
Time Style	Auto
Max Date	02/03/2017 11:08:49 -0800
Min Date	01/04/2017 11:08:49 -0800
Display Date In Title	✓ true

OK | Cancel

Property	Description	Supports Axes Types

➕	Add X axis. When you add an X axis, you can select from one of the following axis types: Number, Date, Category, Logarithmic, Elapsed, and Symbol. Click the green plus icon, select an Axis Type, enter an Axis Name, and click OK.  Add New Axis Axis Name: Date Axis Type: Date Axis 	All
✖	Delete an existing axis. Select the axis, and click the Delete icon.	All
Axis Visible	If false, the axis will be hidden.	All
Axis Label	Name of the axis.	All
Axis Label Angle	Angle of the value on the axis label.	All
Axis Label Color	Color of axis label.	All
Axis Label Font	Font type and size of text on axis label.	All
Tick Labels Visible	If false, the tick labels will be hidden.	All
Tick Label Color	Color of tick labels.	All
Tick Label Font	Font type and size of text on tick labels.	All
Tick Marks Visible	If false, the tick marks will be hidden.	All
Tick Mark Color	Color of tick marks.	All
Tick Mark Inside Length	Length of tick marks inside the chart.	All
Tick Mark Outside Length	Length of tick marks outside the chart.	All
Axis Position	Depends on the axis selected. X-axis label alternates between top and bottom. Y-axis label alternates between left and right. You many need to change both X and Y axis properties to get your intended axis position.	All
Auto Range	If true, the value axis range will be determined automatically. If false, the specified Lower and Upper bounds will be used.	All

Auto Range Min Size	If true, the minimum value range is used.	Date, Number, Logarithmic, Symbol, Elapsed
Fixed Auto Range	Sets an axis up for dynamic graphs.	Date, Number, Logarithmic, Symbol, Elapsed
Lower Bound	Lower bound value. Used only when Auto Range is false.	Date, Number, Logarithmic, Symbol, Elapsed
Upper Bound	Upper bound value. Used only when Auto Range is false.	Date, Number, Logarithmic, Symbol, Elapsed
Lower Margin (% of range)	Lower margin represented as a percentage. Used only when Auto Range is true.	Date, Number, Logarithmic, Symbol, Elapsed
Upper Margin (% of range)	Upper margin represented as a percentage. Used only when the Auto Range is true.	Date, Number, Logarithmic, Symbol, Elapsed
Negative Arrow	If true, negative arrow is visible.	Date, Number, Logarithmic, Symbol, Elapsed
Positive Arrow	If true, positive arrow is visible.	Date, Number, Logarithmic, Symbol, Elapsed
Vertical Tick Labels	Vertical orientation for tick labels.	Date, Number, Logarithmic, Symbol, Elapsed
Auto Range Includes Zero	If true, the range includes a zero.	Date, Number, Logarithmic, Symbol, Elapsed
Auto Range Sticky Zero	If true, the zero is on both the XY axes.	Date, Number, Logarithmic, Symbol, Elapsed
Number Format Override	Overwrites the current number format.	Date, Number, Logarithmic, Symbol
Date Style	The style of the date displayed on the axis.	Date
Time Style	The style of the time displayed on the axis.	Date
Max Date	Max value in a series of dates.	Date
Min Date	Min value in a series of dates.	Date
Display Date in Title	If true, the date will be displayed in the title when the range is zoomed into the hour range.	Date
Label Angle	The angle for the value axis labels.	Category
"1e#"-style tick labels	If true, uses scientific notation format (i.e.,1e5, 1e6, etc.,).	Logarithmic

"10^n"-style tick labels	If true, uses power notation format (i.e., 10 to the "X" power).	Logarithmic
Symbols String	Sequence of characters such as a literal constant. (i.e., On,Off,Auto)	Symbols
Grid Bands Visible	If true, grid bands will be hidden.	Symbols
Grid Bands Color	Color of grid bands.	Symbols
Grid Bands Alternate Color	Backup color of grid bands.	Symbols
Format String	Specified sequence of characters.	Elapsed

The Y-Axes tab is where Y-Axis properties are configured. You can also add and delete Y axes here.

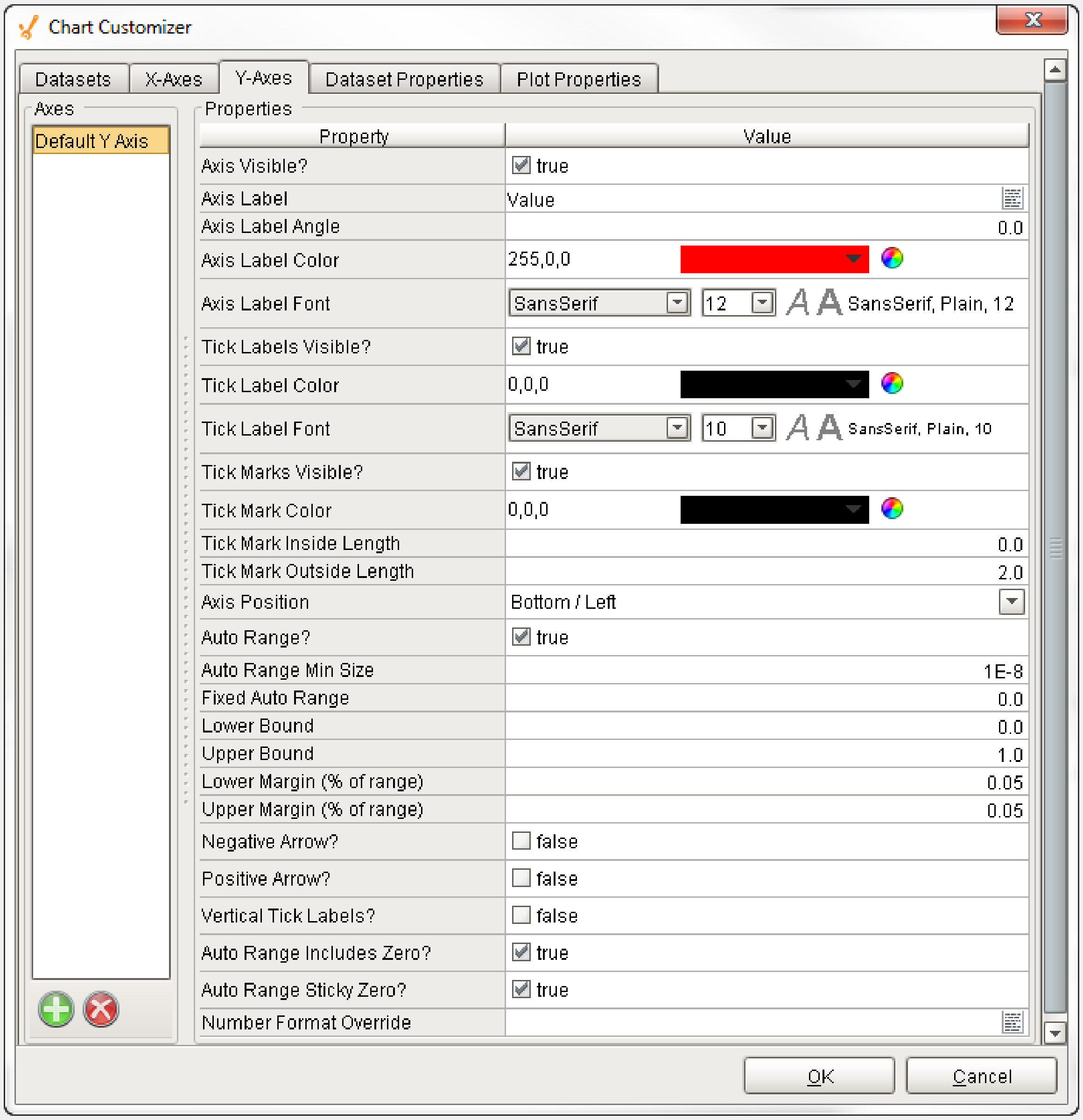


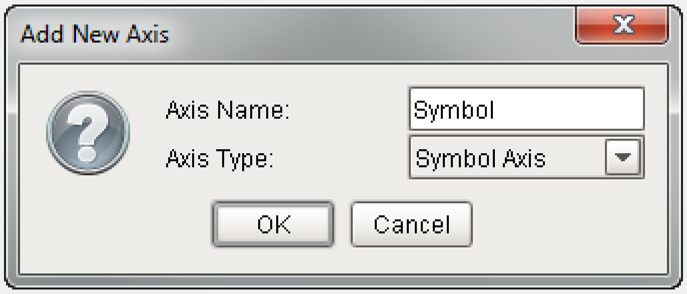
Chart Customizer

Datasets | X-Axes | Y-Axes | Dataset Properties | Plot Properties

Axes: Default Y Axis

Properties

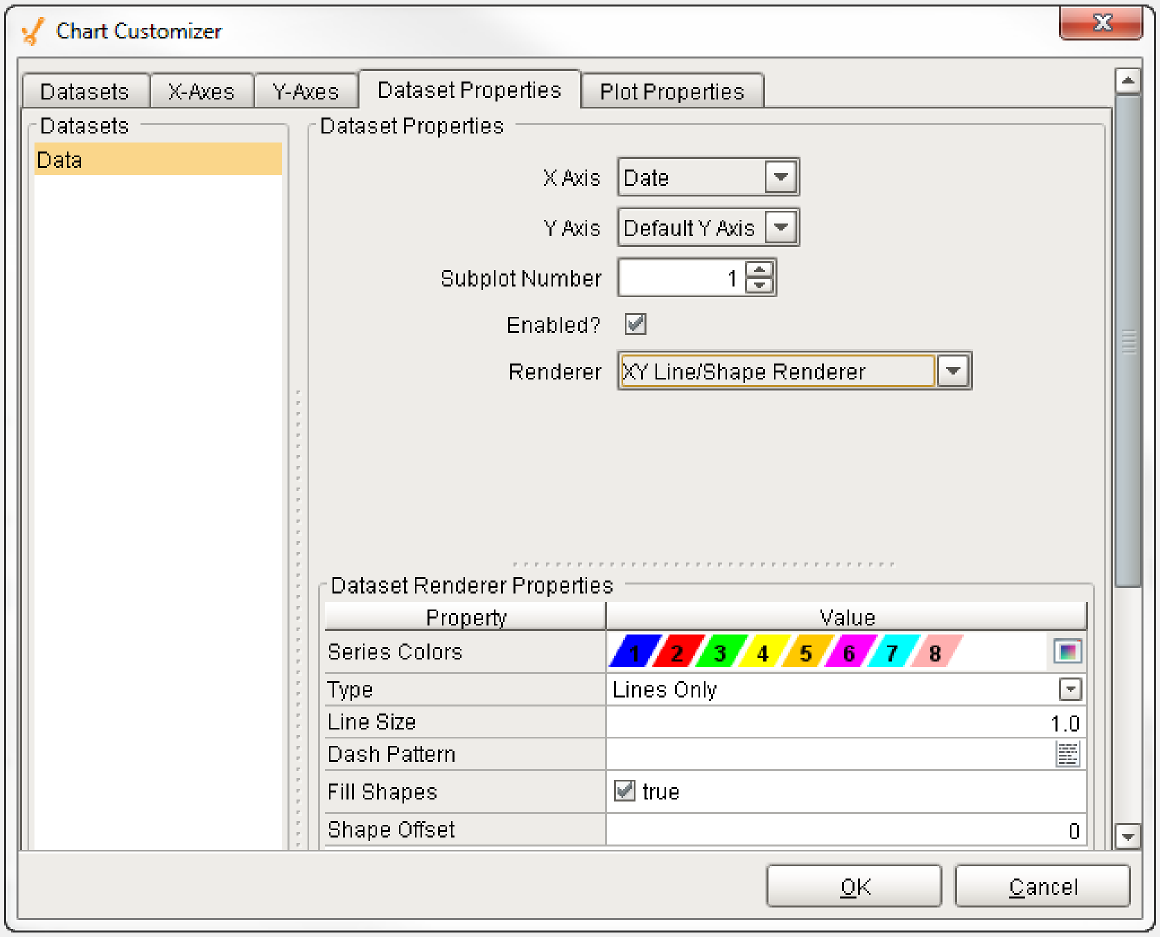
Property	Value
Axis Visible?	✓ true
Axis Label	Value
Axis Label Angle	0.0
Axis Label Color	255,0,0 🟥
Axis Label Font	SansSerif 12 SansSerif, Plain, 12
Tick Labels Visible?	✓ true
Tick Label Color	0,0,0 ⬛
Tick Label Font	SansSerif 10 SansSerif, Plain, 10
Tick Marks Visible?	✓ true
Tick Mark Color	0,0,0 ⬛
Tick Mark Inside Length	0.0
Tick Mark Outside Length	2.0
Axis Position	Bottom / Left
Auto Range?	✓ true
Auto Range Min Size	1E-8
Fixed Auto Range	0.0
Lower Bound	0.0
Upper Bound	1.0
Lower Margin (% of range)	0.05
Upper Margin (% of range)	0.05
Negative Arrow?	☐ false
Positive Arrow?	☐ false
Vertical Tick Labels?	☐ false
Auto Range Includes Zero?	✓ true
Auto Range Sticky Zero?	✓ true
Number Format Override	

Property	**Description**	**Axes Types Supports**
➕	Add axis. When you add a Y axis, you can select from one of the following axis types: Number, Date, Category, Logarithmic, Elapsed, and Symbol. Click the green plus icon, select an Axis Type, enter an Axis Name, and click OK.  Add New Axis Axis Name: Symbol Axis Type: Symbol Axis 	All
❌	Delete an existing axis. Select an axis, and click the Delete icon.	All

Axis Visible	If false, the axis will be hidden.	All
Axis Label	Name of the axis.	All
Axis Label Angle	Angle of the value on the axis label.	All
Axis Label Color	Color of axis label.	All
Axis Label Font	Font type and size of text on axis label.	All
Tick Labels Visible	If false, the tick labels will be hidden.	All
Tick Label Color	Color of tick labels.	All
Tick Label Font	Font type and size of text on tick labels.	All
Tick Marks Visible	If false, the tick marks will be hidden.	All
Tick Mark Color	Color of tick marks.	All
Tick Mark Inside Length	Length of tick marks inside the chart.	All
Tick Mark Outside Length	Length of tick marks outside the chart.	All
Axis Position	Depends on the axis selected. X-axis label alternates between top and bottom. Y-axis label alternates between left and right. You many need to change both X and Y axis properties to get your intended axis position.	All
Auto Range	If true, the value axis range will be determined automatically. If false, the specified Lower and Upper bounds will be used.	All
Auto Range Min Size	If true, the minimum value range is used.	Date, Number, Logarithmic, Symbol, Elapsed
Fixed Auto Range	Sets an axis up for dynamic graphs.	Date, Number, Logarithmic, Symbol, Elapsed
Lower Bound	Lower bound value. Used only when Auto Range is false.	Date, Number, Logarithmic, Symbol, Elapsed
Upper Bound	Upper bound value. Used only when Auto Range is false.	Date, Number, Logarithmic, Symbol, Elapsed
Lower Margin (% of range)	Lower margin represented as a percentage. Used only when Auto Range is true.	Date, Number, Logarithmic, Symbol, Elapsed

Upper Margin (% of range)	Upper margin represented as a percentage. Used only when the Auto Range is true.	Date, Number, Logarithmic, Symbol, Elapsed
Negative Arrow	If true, negative arrow is visible.	Date, Number, Logarithmic, Symbol, Elapsed
Positive Arrow	If true, positive arrow is visible.	Date, Number, Logarithmic, Symbol, Elapsed
Vertical Tick Labels	Vertical orientation for tick labels.	Date, Number, Logarithmic, Symbol, Elapsed
Auto Range Includes Zero	If true, the range includes a zero.	Date, Number, Logarithmic, Symbol, Elapsed
Auto Range Sticky Zero	If true, the zero is on both the XY axes.	Date, Number, Logarithmic, Symbol, Elapsed
Number Format Override	Overwrites the current number format.	Date, Number, Logarithmic, Symbol
Date Style	The style of the date displayed on the axis.	Date
Time Style	The style of the time displayed on the axis.	Date
Max Date	Max value in a series of dates.	Date
Min Date	Min value in a series of dates.	Date
Display Date in Title	If true, the date will be displayed in the title when the range is zoomed into the hour range.	Date
Label Angle	The angle for the value axis labels.	Category
"1e#"-style tick labels	If true, uses scientific notation format (i.e.,1e5, 1e6, etc.,).	Logarithmic
"10^n"-style tick labels	If true, uses power notation format (i.e., 10 to the "X" power).	Logarithmic
Symbols String	Sequence of characters such as a literal constant. (i.e., On,Off,Auto)	Symbols
Grid Bands Visible	If true, grid bands will be hidden.	Symbols
Grid Bands Color	Color of grid bands.	Symbols
Grid Bands Alternate Color	Backup color of grid bands.	Symbols
Format String	Specified sequence of characters.	Elapsed

The Dataset tab is where you can modify the visual styles of your chart. You can configure your chart with subplots, experiment with different renderer types and property types to change how the data is displayed until you find what best meets your requirements. Note: Not all Renderer properties are available for each axis type.



Dataset Properties

X Axis: Date
Y Axis: Default Y Axis
Subplot Number: 1
Enabled? ✓
Renderer: XY Line/Shape Renderer

Dataset Renderer Properties

Property	Value
Series Colors	1 2 3 4 5 6 7 8
Type	Lines Only
Line Size	1.0
Dash Pattern	
Fill Shapes	✓ true
Shape Offset	0

Dataset Tab Properties

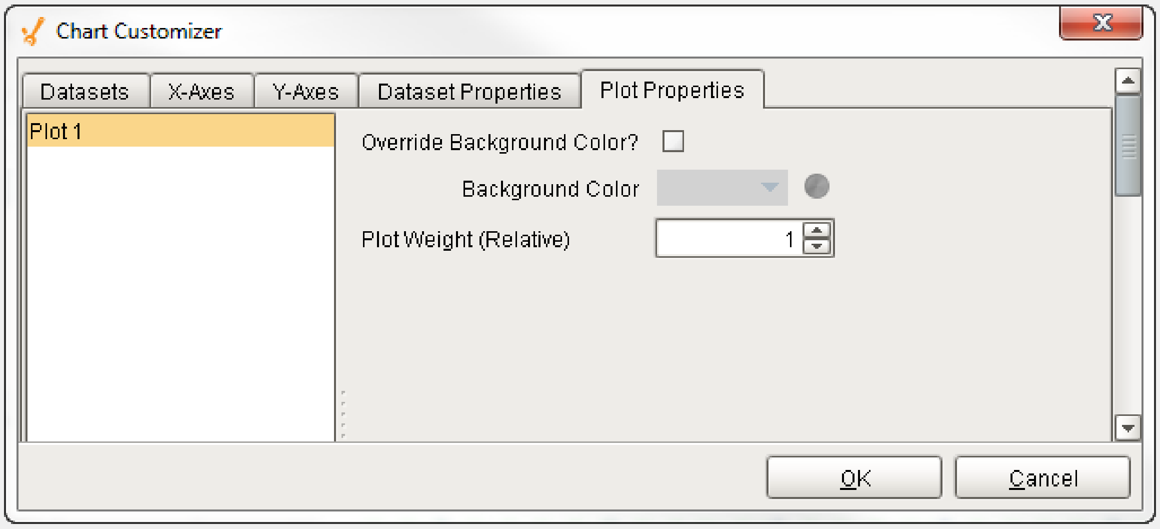
Property	**Description**	**Axes Types Supports**
Dataset	Collection of data in tabular form. Data from the dataset drives the chart.	All
X Axis	Horizontal axis.	All
Y Axis	Vertical axis.	All
Subplot Number	Number of plot areas on one chart.	All
Enabled	If true, the chart is displayed with the selected renderer properties.	All

Renderer	The visual style of the data presented on the chart. Select from various renderer styles: • XY Line/Shape Renderer • XY Bar Renderer • XY Area Renderer • XY Step Renderer • XY Step Area Renderer • XY Dot Renderer • Category Line/Shape Renderer • Category Bar Renderer	All
Series Colors	An ordered list of the colors to draw series in.	All
Type	Type of XY Item Renderer.	All
Line Size	The thickness of the line.	All
Dash Pattern	The pattern used for dashed lines.	All
Fill Shapes	If false, there is only an outline of the shape, no fill color.	All
Shape Offset	The offset into the standard shape list to start this renderer at. Offsets and their shapes are listed below.	All

Offset	Shape
0	Square
1	Circle
2	Upward triangle
3	Diamond
4	Horizontal rectangle
5	Downward triangle
6	Horizontal ellipse
7	Rightward triangle
8	Vertical rectangle
9	Leftward triangle

Margin	The percentage by which the bars are trimmed using the XY Bar Renderer.	All
Shadows	If true, draws shadows under the bars using the XY Bar Renderer.	All
Outline	If true, draws an outline around the area using the XY Area Renderer.	All
Draw Lines	If true, lines will be drawn to connect the datapoints using the Category Line/Shape Renderer.	All
Draw Shapes	If true, shapes will be drawn to connect each datapoint if using the Category Line /Shape Renderer.	All

The Plot Properties tab allows you to break up the chart plot area into multiple distinct subplots.



Plot 1

Override Background Color? ☐

Background Color

Plot Weight (Relative) 1

Property Name	Description	
Plot	The chart area displaying data.	All
Override Background Color	If enabled, allows you to change the background color.	All
Background Color	Background color of the chart.	All
Plot Weight (Relative)	The chart ratio between subplots.	All

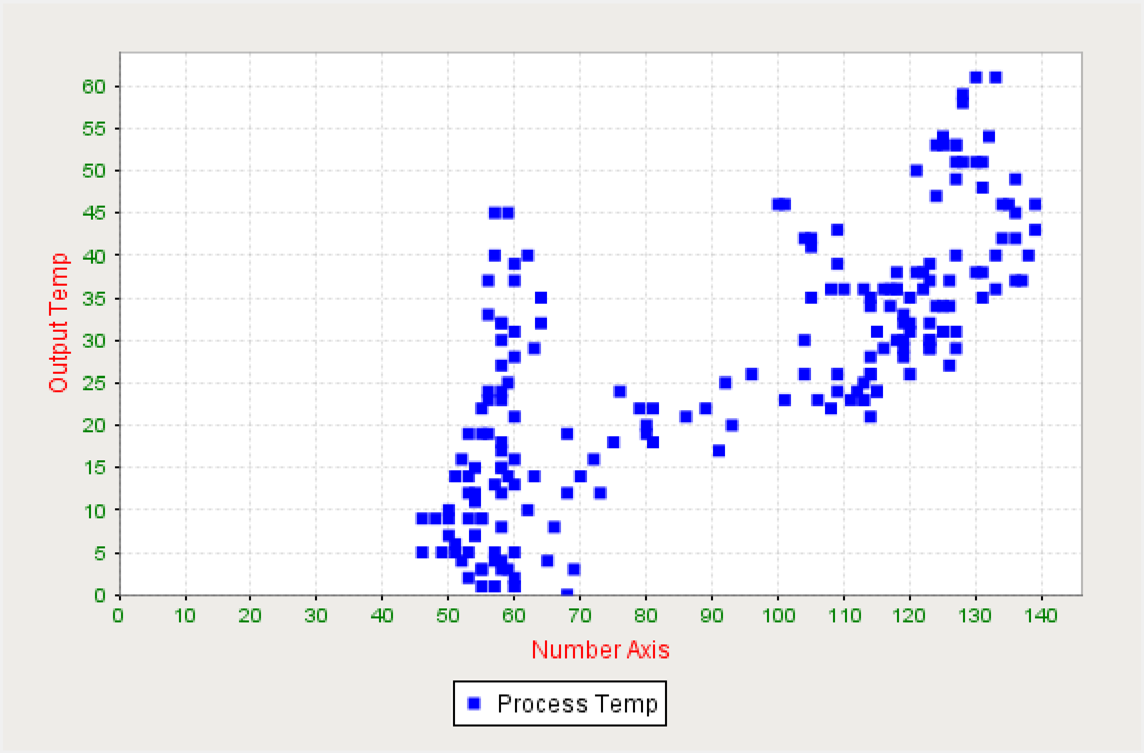
References

- Vision - Chart
- Component Customizers
- Understanding Component Customizers

Axis Type Examples

The Chart Customizer has six different axis types to choose from when configuring a chart, each with its own list of properties. Note: Some customizer properties are specific to the axis type and have their own unique properties. Examples of all axis types are shown below along with the property settings used to create each chart.

Number Axis Chart

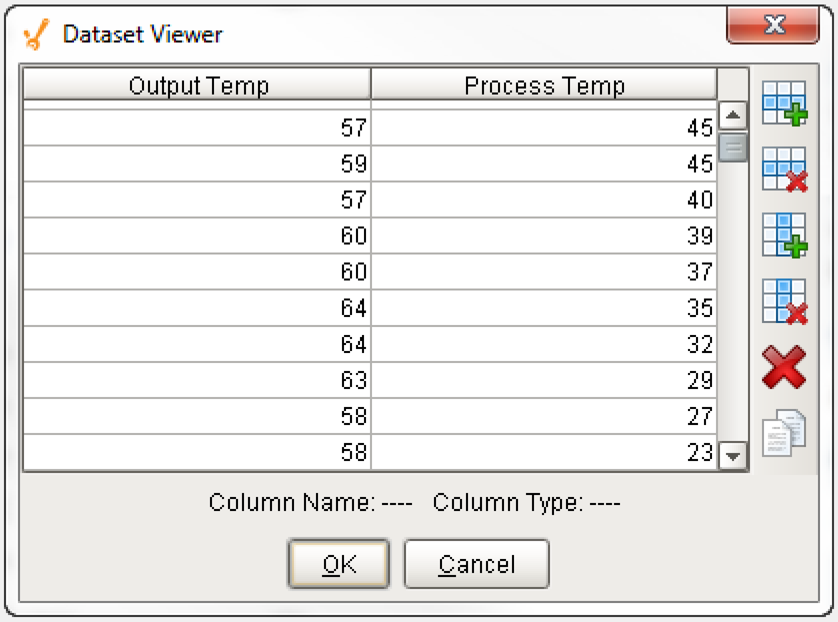


Binding Type	
Tag	Tag History

Chart Customizer Property Settings

Datasets Tab	
Property Name	**Value**
Datasets	Data
X-Axes Tab	
Axes	Number
X Axis Label	Number Axis
Axis Label Color	Red
Tick Label Color	Green
Y-Axes Tab	
Axes	Default Y Axis
Y Axis Label	Output Temp
Axis Label Color	Red
Tick Label Color	Green
Dataset Properties Tab	
X Axis	Number
Y Axis	Default Y Axis
Renderer	XY Line/Shape Renderer
Type	Shapes Only

Data Property Dataset



Dataset Viewer

Output Temp	Process Temp
57	45
59	45
57	40
60	39
60	37
64	35
64	32
63	29
58	27
58	23

Column Name: ---- Column Type: ----

Date Axis Chart

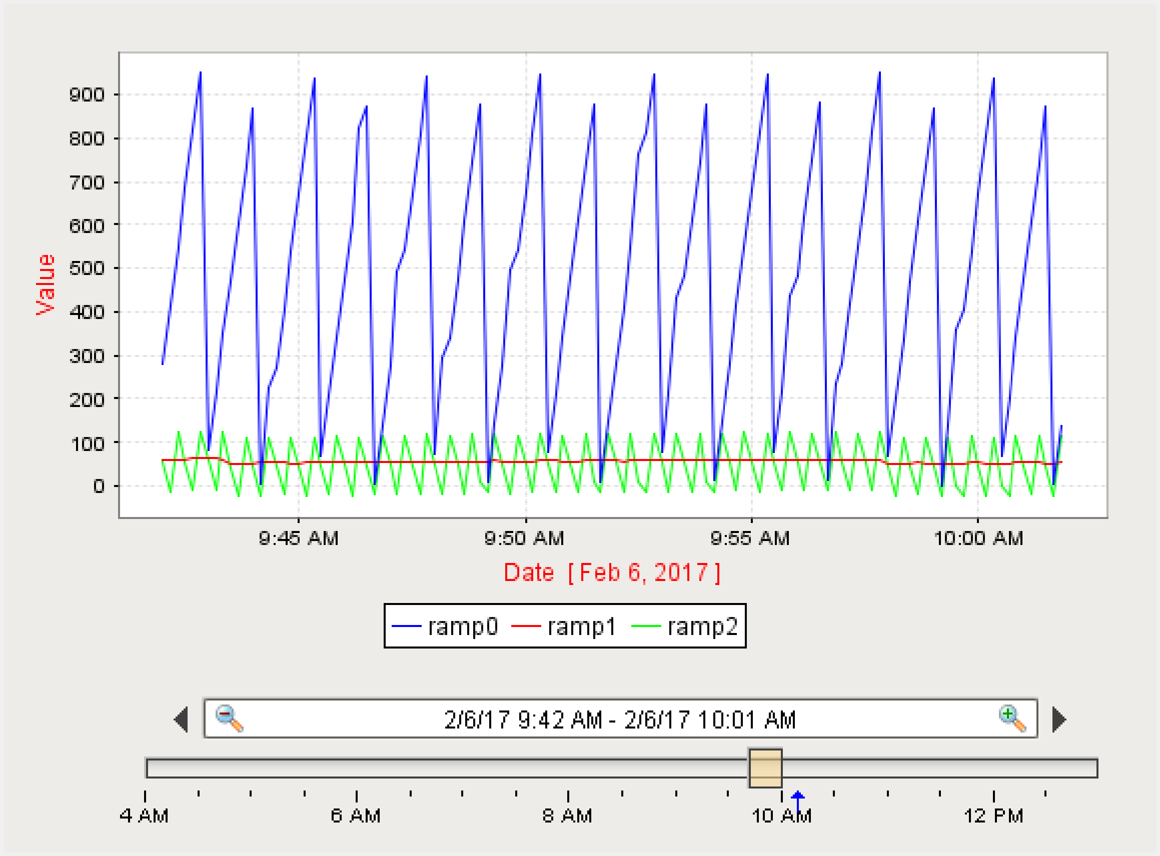
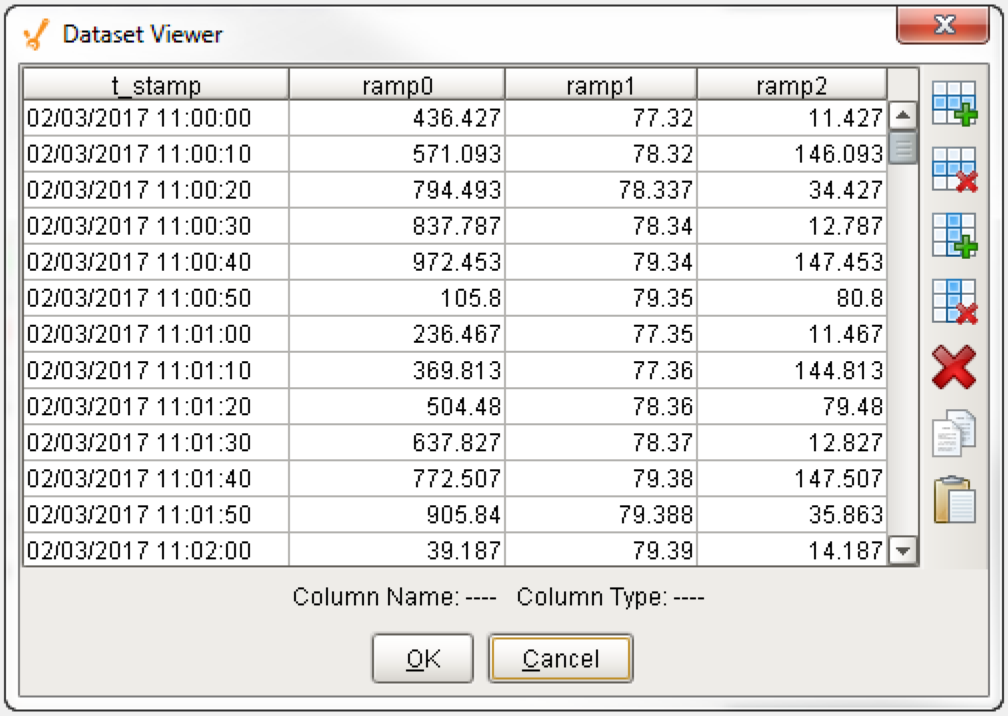


Chart Customizer Property Settings

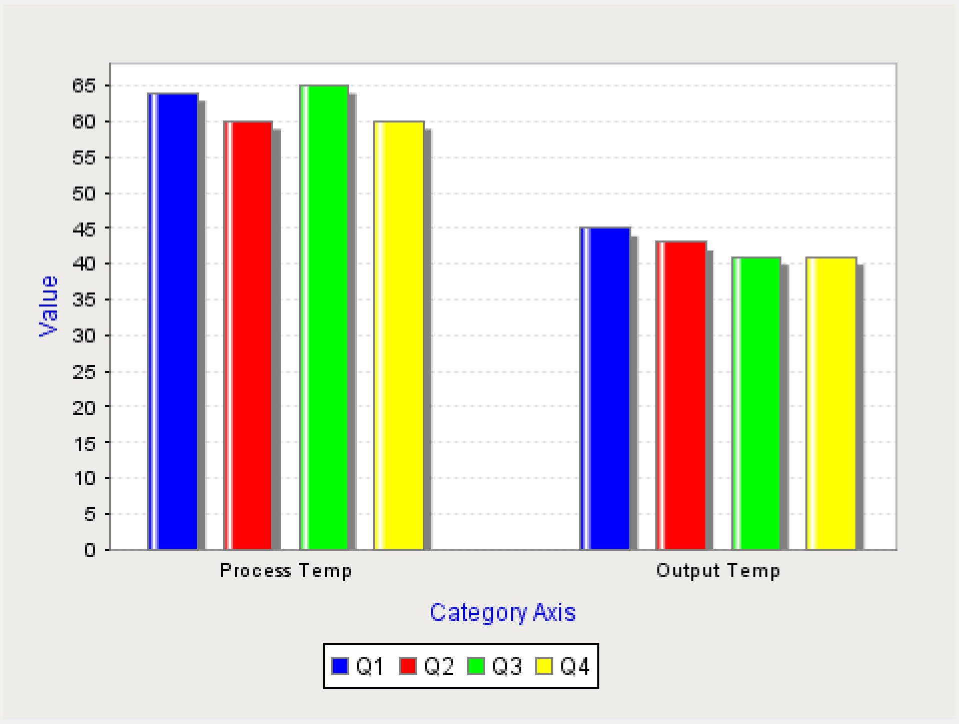
Datasets Tab	
Propert Name	**Description**
Dataset	Data
X-Axes Tab	
Axes	Date
Axis Label	Date
Axis Label Color	Red
Y-Axes Tab	
Axes	Default Y Axis
Axis Label	Value
Axis Label Color	Red
Dataset Properties Tab	
Datasets	Data
X Axis	Date
Y Axis	Default Y Axis
Renderer	XY Line/Shape Render
Type	Lines Only

Data Property Dataset



t_stamp	ramp0	ramp1	ramp2
02/03/2017 11:00:00	436.427	77.32	11.427
02/03/2017 11:00:10	571.093	78.32	146.093
02/03/2017 11:00:20	794.493	78.337	34.427
02/03/2017 11:00:30	837.787	78.34	12.787
02/03/2017 11:00:40	972.453	79.34	147.453
02/03/2017 11:00:50	105.8	79.35	80.8
02/03/2017 11:01:00	236.467	77.35	11.467
02/03/2017 11:01:10	369.813	77.36	144.813
02/03/2017 11:01:20	504.48	78.36	79.48
02/03/2017 11:01:30	637.827	78.37	12.827
02/03/2017 11:01:40	772.507	79.38	147.507
02/03/2017 11:01:50	905.84	79.388	35.863
02/03/2017 11:02:00	39.187	79.39	14.187

Category Axis Chart



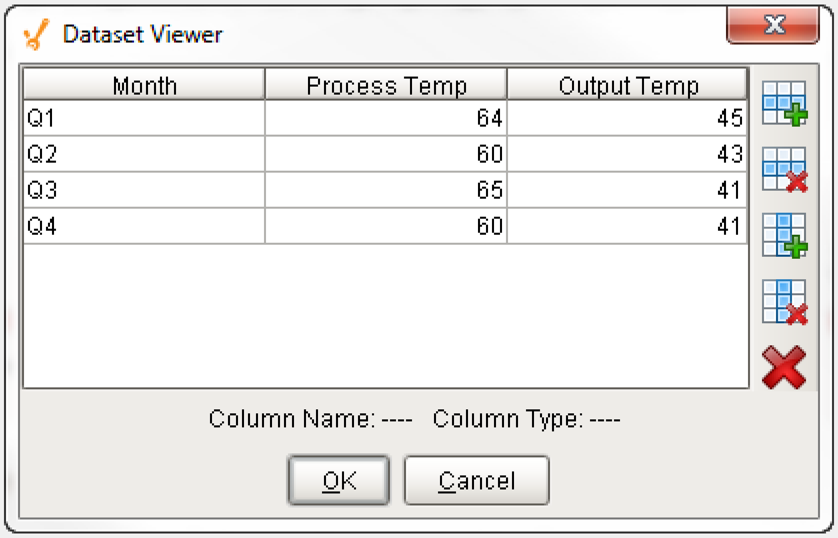
Property Editor Setting

Behavior	
Property	**Value**
Chart Type	Category

Chart Customizer Property Settings

Datasets Tab	
Property Name	**Value**
Dataset	Data
X-Axes Tab	
Axes	Category
Axis Label	Category Axis
Axis Label Color	Blue
Y-Axes Tab	
Axes	Default Y Axis
Axis Label	Value
Axis Label Color	Blue
Dataset Properties Tab	
Datasets	Data
X Axis	Category
Y Axis	Default Y Axis
Renderer	Category Bar Renderer
Style	Bar

Data Property Dataset



Month	Process Temp	Output Temp
Q1	64	45
Q2	60	43
Q3	65	41
Q4	60	41

Column Name: ---- Column Type: ----

Logarithmic Axis Chart

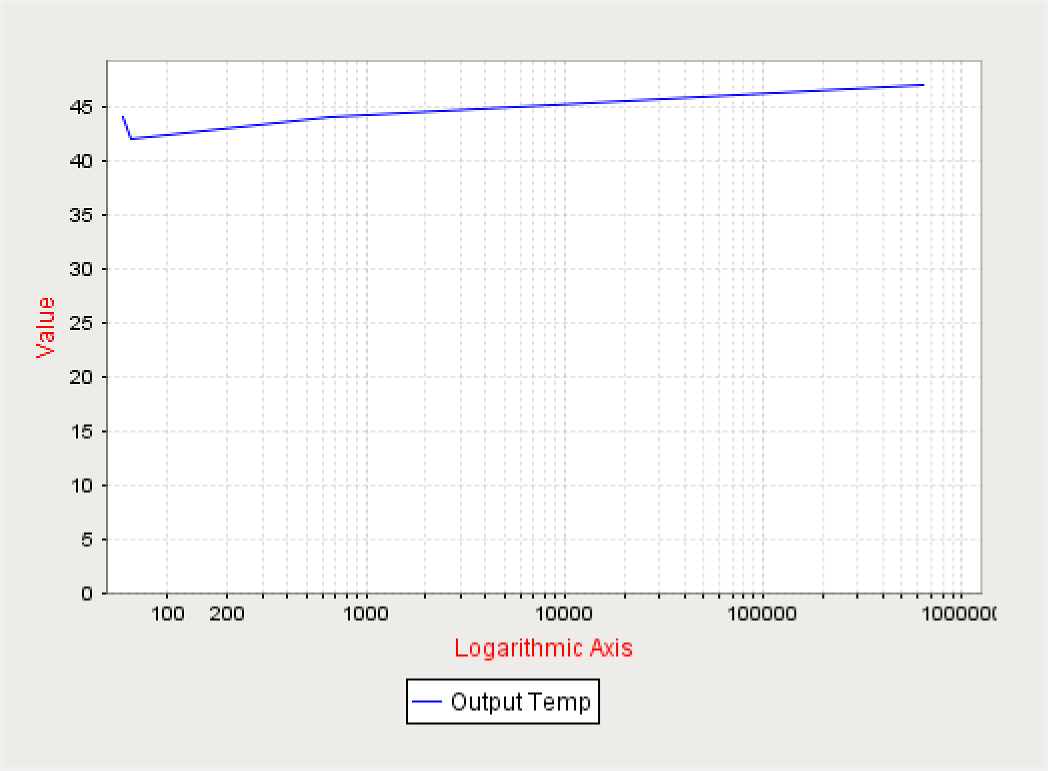
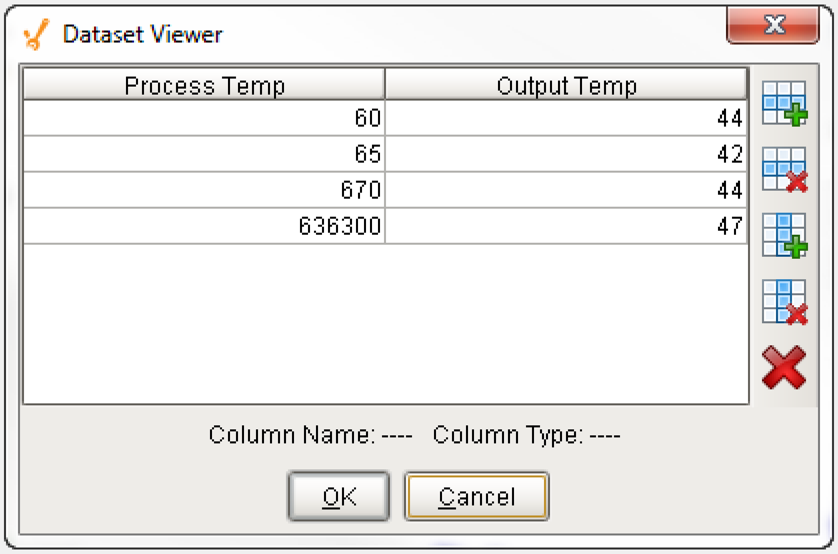


Chart Customizer Property Settings

Datasets Tab	
Property	**Value**
Datsets	Data
X-Axes Tab	
Axes	Logarithmic
Axis Label	Logarithmic Axis
Axis Label Color	Red
Y-Axes Tab	
Axes	Default Y Axis
Axis Label	Value
Axis Label Color	Red
Dataset Properties Tab	
Datasets	Data
X Axis	Logarithmic
Y Axis	Default Y Axis
Renderer	XY Line/Shape Renderer
Type	Lines Only

Data Property Dataset



Dataset Viewer

Process Temp	Output Temp
60	44
65	42
670	44
636300	47

Column Name: ---- Column Type: ----

Symbols Axis Chart

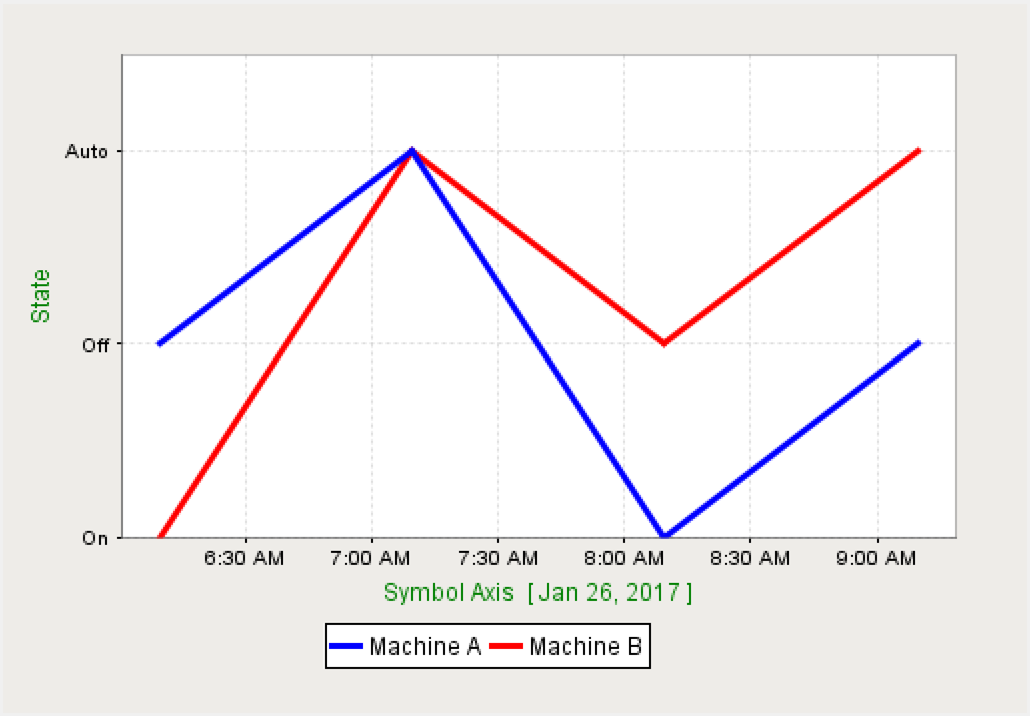
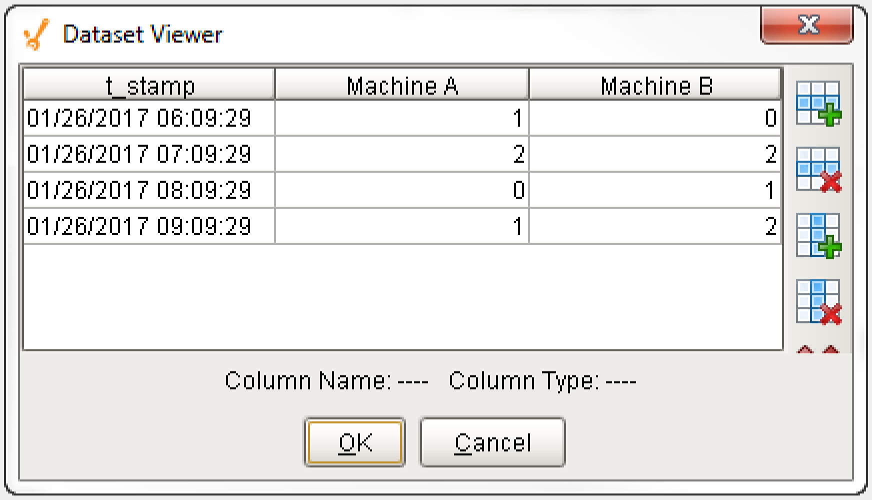


Chart Customizer Property Settings

Datasets Tab	
Property Name	**Value**
Dataset	Data
X-Axes Tab	
Axes	Default Axis
Axis Label	Symbol Axis
Axis Label Color	Green
Y-Axes Tab	
Axes	Symbol
Axis Label	State
Axis Label Color	Green
Symbols String	On,Off,Auto
Dataset Properties Tab	
Datasets	Data
X Axis	Default X Axis
Y Axis	Symbol
Renderer	XY Line/Shape Renderer
Type	Lines Only
Line Size	3

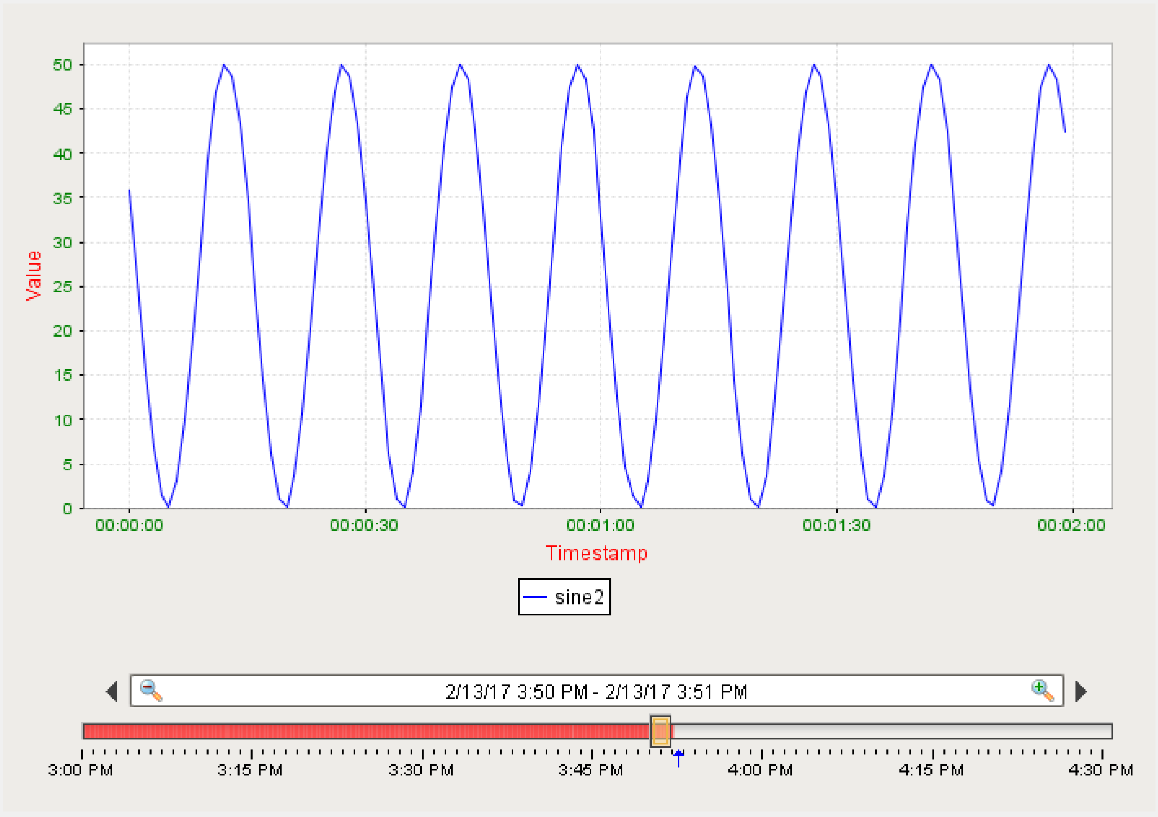
Data Property Dataset



t_stamp	Machine A	Machine B
01/26/2017 06:09:29	1	0
01/26/2017 07:09:29	2	2
01/26/2017 08:09:29	0	1
01/26/2017 09:09:29	1	2

Column Name: ---- Column Type: ----

Elapsed Time Axis Chart

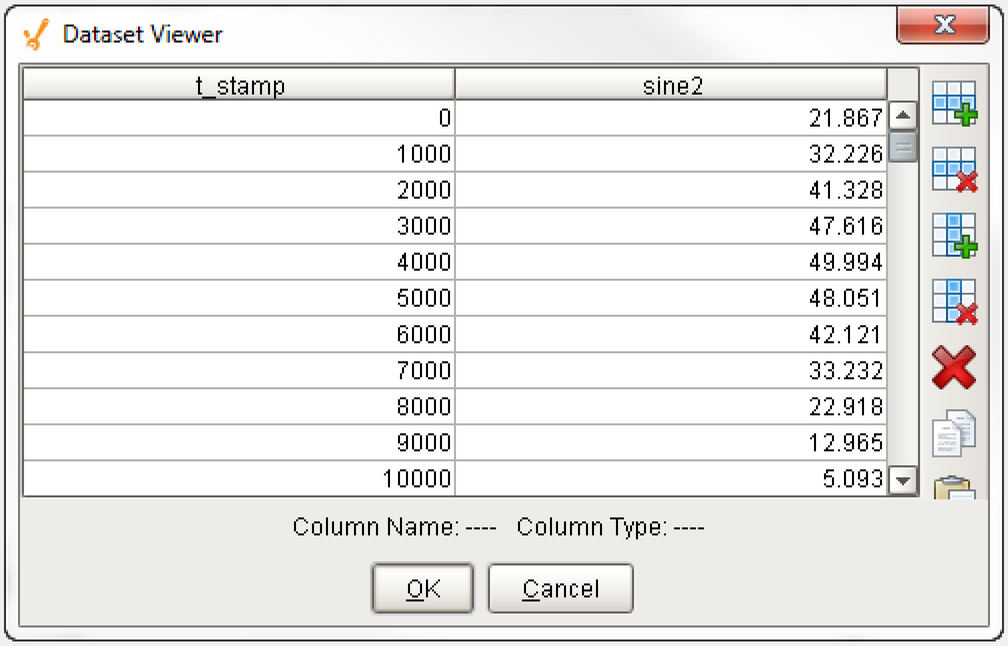


Binding Type	
Database	SQL Query

Chart Customizer Property Settings

Datasets Tab	
Property Name	**Value**
Dataset	Data
X-Axes Tab	
Axes	Elapsed Time
Axis Label	Timestamp
Axis Label Color	Red
Tick Label Color	Green
Upper Bound	60,000
Tick Size (ms)	30,000
Y-Axes Tab	
Axes	Default Y Axis
Axis Label	Value
Axis Label Color	Red
Tick Label Color	Green
Dataset Properties Tab	
Datasets	Data
X Axis	Elapsed
Y Axis	Default Y Axis
Renderer	XY Line/Shape Renderer
Type	Lines Only

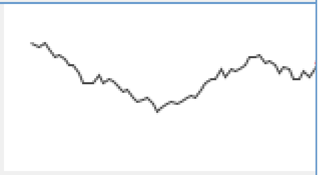
Data Property Dataset



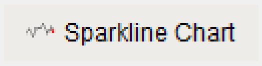
t_stamp	sine2
0	21.867
1000	32.226
2000	41.328
3000	47.616
4000	49.994
5000	48.051
6000	42.121
7000	33.232
8000	22.918
9000	12.965
10000	5.093

Vision - Sparkline Chart

General



Component Palette Icon:



Description

The sparkline chart is a minimalistic chart component that displays a line-chart history for a single datapoint. Sparklines were invented by Edward Tufte as a way to show a great deal of contextual information in a very small amount of space. Sparklines are typically used to display the recent history (up to current time) of a datapoint so that the viewer can quickly discern the recent trend of a datapoint: is it rising? falling? oscillating? etc..

To use a sparkline, bind its Data property either to a Tag Historian realtime query, or to a database query. There should be two columns in this dataset: the first one a date column, the second a number. Each row will become a datapoint on the chart, and the dataset must be sorted by time in ascending order.

Instead of using axes to convey scale, the sparkline can display a band of color across the back of the chart which indicates the desired operating range of the datapoint. In this way, it is instantly obvious when a value is in its expected range, above that range, or below. The sparkline automatically configures its internal axes based on the data given to it. To give it a fixed range, simply fill in the Range Highand Range Low properties.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is not affected by rotation.	Border	.border	Common
Border Inset	The amount of space to inset the chart inside its border.	double	.borderInset	Appearance
Chart Max	The value that corresponds to the upper edge of the chart. (Read only. Usable in bindings and scripting.)	Double	.chartMax	Uncategorized
Chart Min	The value that corresponds to the lower edge of the chart. (Read only. Usable in bindings and scripting.)	Double	.chartMin	Uncategorized
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Data	The history data to draw in the sparkline chart.	Dataset	.data	Data
Desired High	The high value of the desired operating range. If left blank (null), no desired range band will be shown.	Double	.desiredHi	Data

Desired Low	The low value of the desired operating range. If left blank (null), no desired range band will be shown.	Double	.desiredLo	Data
Desired Range Color	The color of the desired operating range band. Only used if the desired operating range is configured. See Color Selector.	Color	.desiredRangeColor	Appearance
First Marker Color	The color of the first value marker. See Color Selector.	Color	.firstMarkerColor	Markers
First Marker Size	The size of the first value marker.	double	.firstMarkerSize	Markers
First Marker Style	The style of the first value marker.	int	.firstMarkerStyle	Markers
First Value	The first (oldest) value in the dataset. (Read only. Usable in bindings and scripting.)	Double	.firstValue	Uncategorized
High Marker Color	The color of the high value marker. See Color Selector.	Color	.hiMarkerColor	Markers
High Marker Size	The size of the high value marker.	double	.hiMarkerSize	Markers
High Marker Style	The style of the high value marker.	int	.hiMarkerStyle	Markers
Last Marker Color	The color of the last value marker. See Color Selector.	Color	.lastMarkerColor	Markers
Last Marker Size	The size of the last value marker.	double	.lastMarkerSize	Markers
Last Marker Style	The style of the last value marker.	int	.lastMarkerStyle	Markers
Last Value	The last (most recent) value in the dataset. (Read only. Usable in bindings and scripting.)	Double	.lastValue	Uncategorized
Line Color	The color of the sparkline. See Color Selector.	Color	.foreground	Appearance
Line Width	The width of the sparkline.	float	.lineWidth	Appearance
Low Marker Color	The color of the low value marker. See Color Selector.	Color	.loMarkerColor	Markers
Low Marker Size	The size of the low value marker.	double	.loMarkerSize	Markers
Low Marker Style	The style of the low value marker.	int	.loMarkerStyle	Markers
Max Value	The largest value in the dataset. (Read only. Usable in bindings and scripting.)	Double	.maxValue	Uncategorized
Min Value	The smallest value in the dataset. (Read only. Usable in bindings and scripting.)	Double	.minValue	Uncategorized
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common

Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	int	.quality	Data
Range High	A fixed value for the upper edge of the chart. If left blank (null), the upper range will be calculated automatically.	Double	.rangeHi	Data
Range Low	A fixed value for the lower edge of the chart. If left blank (null), the lower range will be calculated automatically.	Double	.rangeLo	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

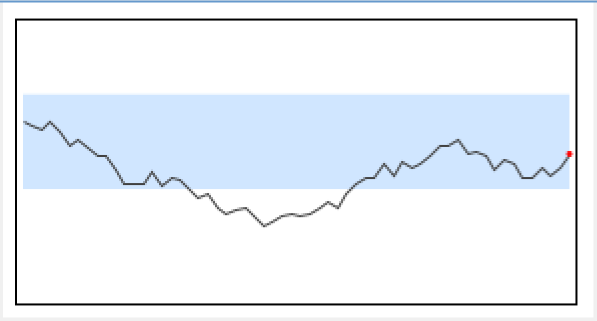
.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

This component does not have any custom properties.

Examples

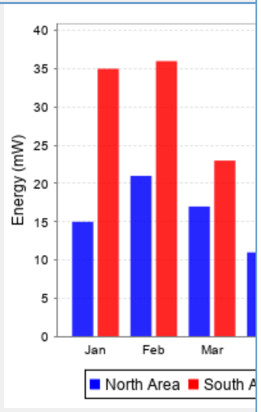
Sparkline Chart with Low and High Limits



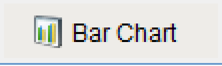
Property Name	Value
Desired Range Color	184,218,255
Range High	100
Range Low	0
Desired High	75
Desired Low	40

Vision - Bar Chart

General



Component Palette Icon:



Description

The Bar Chart is a very easy-to-use chart that provides a familiar bar representation of any numeric values. That is, the height of the bars is determined by some numeric value in the underlying dataset. It is often configured to display as a category chart. A category chart is a chart whose X-values are categories (strings, names, groupings, etc) rather than numeric values (numbers, dates).

Like most chart components (other than the Easy Chart), the Data property drives the chart. The first column in the Data dataset defines the names of the categories. The rest of the columns define the values for each of the series (if there is more than one series per category), and thus should be numeric. Note - if your data is 'turned on its side', meaning that the columns define the categories and rows define the series, then set the Extract Order to "By Column".

Properties

Name	Description	Property Type	Scripting	Category
Bar Label Color	The color for the bar labels. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.barLabelColor	Axes
Bar Label Font	The font for the bar labels.	Font	.barLabelFont	Axes
Bar Label Offset	The offset between the bar and the bar label.	double	.barLabelOffset	Axes
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Category Axis Label	The label for the category axis.	String	.categoryLabel	Axes

Category Axis Label Angle	The angle for the value axis' labels.	int	.catAxisLabelPosition	Axes
Category Axis Label Color	The color for the category axis label. See Color Selector.	Color	.catAxisLabelColor	Axes
Category Axis Label Font	The font for the category axis label.	Font	.catAxisLabelFont	Axes
Category Axis Lower Margin	The lower margin, as a percentage, of the category axis.	double	.catAxisLowerMargin	Axes
Category Axis Tick Color	The color for the category axis' ticks. See Color Selector.	Color	.catAxisTickColor	Axes
Category Axis Tick Font	The font for the category axis' ticks.	Font	.catAxisTickFont	Axes
Category Axis Upper Margin	The upper margin, as a percentage, of the category axis.	double	.catAxisUpperMargin	Axes
Category Margin	The margin between categories as a fraction of the total space.	double	.categoryMargin	Appearance
Chart Title	An optional title that will appear at the top of the chart.	String	.title	Appearance
Chart Type	Controls how the bar chart is displayed.	int	.rendererType	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Data	The data driving the chart.	Dataset	.data	Data
Extract Order	Controls whether the first row defines the categories or the series.	int	.extractOrder	Data
Foreground Transparency	The transparency of the bars (useful for 3D bars). Valid values are between 0 (0% opacity) and 1 (100% opacity).	float	.foregroundAlpha	Appearance
Gradient bars?	If true, bars will be painted with a gradient 'shine'.	boolean	.gradient	Appearance
Item Margin	The margin between bars in a category as a fraction.	double	.itemMargin	Appearance
Labels?	Always display labels?	boolean	.labels	Appearance
Legend Font	The font for the legend items.	Font	.legendFont	Axes
Legend?	If true, show a legend for the chart.	boolean	.legend	Appearance

Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Plot Background	The background color for the plot.	Color	.plotBackground	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Series Colors	The sequence of colors used for series in the bar chart. See Color Selector.	Color[]	.seriesColors	Appearance
Shadows?	If true, bars will have a drop-shadow beneath them.	boolean	.shadows	Appearance
Title Font	The font for the chart's title.	Font	.titleFont	Axes
Tooltips?	If true, show tooltips.	boolean	.tooltips	Behavior
Value Axis Auto-Range	If true, the value axis range will be determined automatically. If false, the specified upper and lower bounds will be used.	boolean	.valAxisAutoRange	Axes
Value Axis Label	The label for the value axis	String	.valueLabel	Axes
Value Axis Label Color	The color for the value axis label. See Color Selector.	Color	.valAxisLabelColor	Axes
Value Axis Label Font	The font for the value axis label.	Font	.valAxisLabelFont	Axes
Value Axis Lower Bound	The lower bound of the value axis. Used only when auto-range is false.	double	.valAxisLowerBound	Axes
Value Axis Tick Color	The color for the value axis' ticks. See Color Selector.	Color	.valAxisTickColor	Axes
Value Axis Tick Font	The font for the value axis' ticks.	Font	.valAxisTickFont	Axes
Value Axis Upper Bound	The upper bound of the value axis. Used only when auto-range is false.	double	.valAxisUpperBound	Axes
Value Axis Upper Margin	The upper margin, as a percentage, of the value axis. Only used when auto-range is true.	double	.valAxisUpperMargin	Axes
Vertical	Sets the orientation of the chart to vertical (true) or horizontal(false)	boolean	.vertical	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

The following feature is new in Ignition version **8.0.16**
Click here to check out the other new features

- Description

  Provides an opportunity to perform further chart configuration via scripting.

- Parameters

  Component self- A reference to the component that is invoking this function.

  JFreeChart chart- A JFreeChart object. Refer to the JFreeChart documentation for API details.

- Return

  Nothing

- Description

  Provides a chance to override the color of each bar. Can be used to have bar colors changed based upon bar value. Returning the value None will use the default bar color for the series.

- Parameters

  Component self - A reference to the component that is invoking this function.

  int series - The series index for this bar.

  int category - The category index for this bar.

  int value - The value (a number) of this bar.

  Color defaultColor - The color that the bar would be if this function wasn't invoked.

- Return

  Color

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

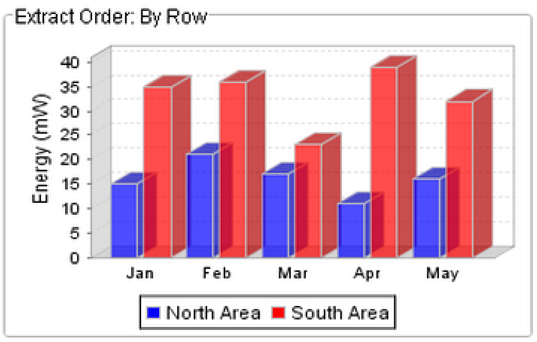
Customizers

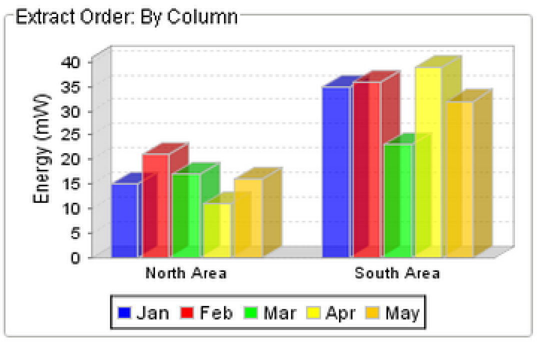
- Vision Component Customizers

Example

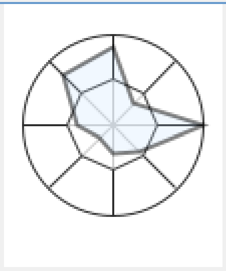
Extract Order Example
The following two charts demonstrate the effects of the extract order property on the given dataset

Label (String)	**North Area** (Integer)	**South Area** (integer)
Jan	15	35
Feb	21	36
Mar	17	23
Apr	11	39
May	16	32





Vision - Radar Chart

General

Component Palette Icon:
Radar Chart

INDUCTIVE UNIVERSITY

Radar Chart

Watch the Video

Description
Radar charts, also known as web charts, spider charts, spider plots, and a few other names, display a dataset as a two dimensional polygon. The plot is arranged as a set of spokes with equal angles between them. Each spoke represents a value axis for the variable it corresponds to. Each dataset is then drawn as a connected polygon, where the points of the polygon are arranged on the spokes according to their value. Each row of the dataset has a minimum and maximum column -- these values are used to determine the scale of the spoke for that variable, with the midpoint representing the desired value. The intended use of radar plots is to display realtime information in such a way that outliers can be quickly identified. This can be an efficient way to convey if a process is running on-spec or off-spec at a glance. The radar chart gets its data from a dataset. Each row in the dataset will become a single variable (spoke) on the chart. The dataset must have a columns labeled "Value", "Min", and "Max"; other columns will be ignored. To display realtime data on a radar chart, you can use a cell-update binding to bind individual values to tag values. You can also drop tags onto a radar chart, with the EngMin binding to min and EngMax binding to max. If there are no existing cell-update bindings, the tags will replace existing data, otherwise the tags will be added to the end of the dataset. Alternatively, you can have realtime information stored by a transaction group to a database table, and drive the radar chart's dataset with a query binding. Refer to Radar Chart to learn more.

Properties

Name	Description	Property Type	Scripting	Category
Actual Fill Color	Fill color for the actual polygon. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.actualFillColor	Appearance
Actual Stroke Color	Stroke color for the actual polygon. See Color Selector.	Color	.actualStrokeColor	Appearance
Actual Stroke Width	Stroke width for the actual polygon.	float	.actualStrokeWidth	Appearance
Background Color	The background color of the component. See Color Selector.	Color	.background	Appearance

Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Border Inset	The amount of area that the chart should be inset from the component bounds.	double	.borderInset	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Data	Contains the datapoints for the radar plot. Each row represents a spoke and point on the polygon.	Dataset	.data	Data
Desired Fill Color	Fill color for the desired polygon. See Color Selector.	Color	.desiredFillColor	Appearance
Desired Stroke Color	Stroke color for the desired polygon. See Color Selector.	Color	.desiredStrokeColor	Appearance
Desired Stroke Width	Stroke width for the desired polygon.	float	.desiredStrokeWidth	Appearance
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Show Desired Shape	Display the desired shape on the chart.	boolean	.showDesiredShape	Appearance
Spoke Color	The color to use for the chart's spokes and exterior ring. See Color Selector.	Color	.foreground	Appearance
Spoke Width	The line width for the chart's spokes and exterior ring.	float	.strokeWidth	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

⚠ This event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

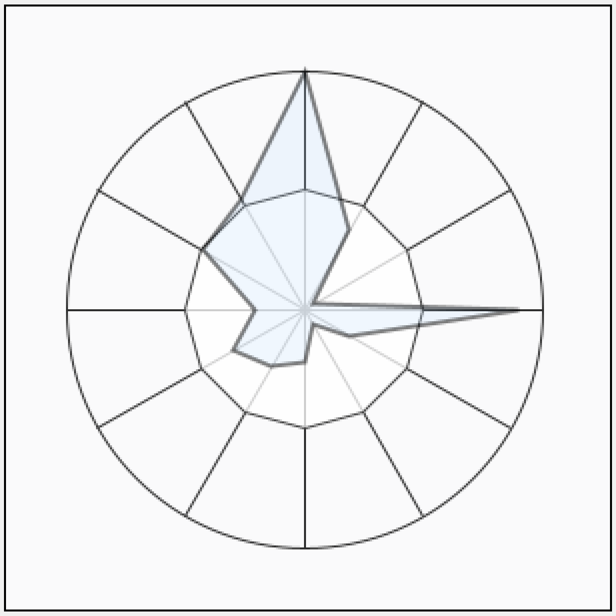
.source	The component that fired this event
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
. property Name	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

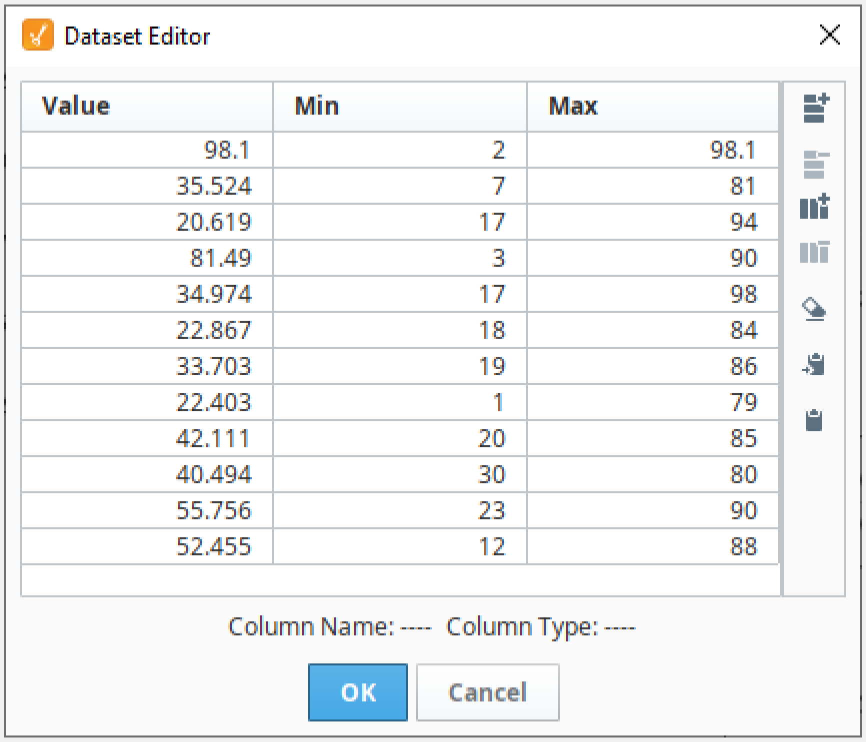
- Vision Component Customizers
- Style Customizer

Examples

Radar Charts display realtime information in such a way that outliers can be quickly identified. In this example, the Radar Chart plotted the values forming a polygon using the raw data in the code block below. You can quickly see where the process is out-of-spec and compare the values to where they should be.



Radar Chart - Dataset Editor



Dataset Editor

Value	Min	Max
98.1	2	98.1
35.524	7	81
20.619	17	94
81.49	3	90
34.974	17	98
22.867	18	84
33.703	19	86
22.403	1	79
42.111	20	85
40.494	30	80
55.756	23	90
52.455	12	88

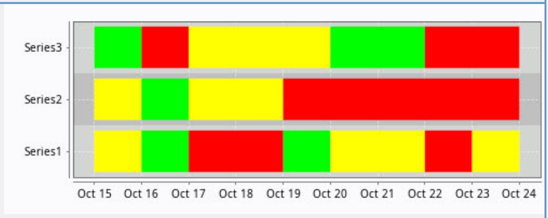
Column Name: ---- Column Type: ----

Radar Chart - Raw Data

```
"#TYPES"
"D","D","D"
"#ROWS","12"
"98.09962923575328","2.0","98.09962923575328"
"35.524092312648314","7.0","81.0"
"20.619468859704142","17.0","94.0"
"81.49014792489209","3.0","90.0"
"34.97383734960057","17.0","98.0"
"22.866686267453773","18.0","84.0"
"33.70266314329313","19.0","86.0"
"22.402620699908937","1.0","79.0"
"42.111234986669811","20.0","85.0"
"40.494873208734567","30.0","80.0"
"55.756456098723458","23.0","90.0"
"52.455123456944321","12.0","88.0"
```

Vision - Status Chart

General



Component Palette Icon:

Status Chart

Description

The Status Chart component allows you to visualize the status of one or more discrete datapoints over a time range. The X-axis is always a timeseries axis, and the Y-axis is a category axis, with one entry per data series. The chart is populated with a single dataset, the first column of which must be a datetime column.

Wide vs Tall Datasets

In Wide format, all of the columns but the first must be numeric. These "series" columns' headers will be used as the names on the y-axis. In Tall format, there should be exactly 3 columns. The first is the timestamp, the second is the series name, and the third is the value. For example:

Wide Format

t_stamp	Valve1	Valve2
2010-01-13 8:00:00	0	2
2010-01-13 8:02:00	0	2
2010-01-13 8:04:00	1	2
2010-01-13 8:06:00	1	1
2010-01-13 8:08:00	0	1

Tall Format

t_stamp	Name	Value
2010-01-13 8:00:00	Valve1	0
2010-01-13 8:00:00	Valve2	2
2010-01-13 8:02:00	Valve1	0
2010-01-13 8:02:00	Valve2	2
2010-01-13 8:04:00	Valve1	1
2010-01-13 8:04:00	Valve2	2
2010-01-13 8:06:00	Valve1	1
2010-01-13 8:06:00	Valve2	1
2010-01-13 8:08:00	Valve1	0
2010-01-13 8:08:00	Valve2	1

Color Mapping

Apart from getting the data into the series chart, the only other commonly configured option is the mapping of discrete values to colors. This is done in the Status Chart Customizer. Each named series can have its own mapping of colors, if desired. These mappings are stored in the expert-level dataset property Series Properties Data so they can be altered at runtime.

Properties

Name	Description	Property Type	Scripting	Category

Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Chart Title	Title of this chart.	String	.chartTitle	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Data Format	Format of the incoming data. In "wide" format, the first column of the dataset needs to be a timestamp, and every subsequent column represents one series in the chart. In "tall" format, the first column is a timestamp, the second column is a series name.	int	.dataFormat	Data
Date Style	The style to display dates in. For international support.	int	.dateStyle	Appearance
Domain Axis Color	Color used on the domain axis. See Color Selector.	Color	.domainAxisColor	Domain Axis
Domain Axis Font	Font used on the domain axis.	Font	.domainAxisFont	Domain Axis
Domain Axis Label	Label on the domain axis.	String	.domainAxisLabel	Domain Axis
Domain Axis Location	Location of the domain axis.	int	.domainAxisLocation	Domain Axis
Legend	Maps chart colors to descriptions.	dataset	.legend	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Properties Loading	The number of properties currently being loaded. (Read only. Usable in bindings and scripting.)	int	.propertiesLoading	Uncategorized
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Range Axis Color	Color used on the range axis. See Color Selector.	Color	.rangeAxisColor	Range Axis
Range Axis Font	Font used on the range axis.	Font	.rangeAxisFont	Range Axis
Range Axis Label	Label on the range axis.	String	.rangeAxisLabel	Range Axis
Range Axis Location	Location of the range axis.	int	.rangeAxisLocation	Range Axis

Range Axis Lower Margin	Lower margin of the range axis.	double	.rangeAxisLowerMargin	Range Axis
Range Axis Upper Margin	Upper margin of the range axis.	double	.rangeAxisUpperMargin	Range Axis
Series Data	Data about each series. Data can be in either "wide" or "tall" format.	Dataset	.data	Data
Series Properties Data	Properties for each series.	Dataset	.properties	Data
Series Spacing	Affects the amount of spacing between series. Can be between 0.0 and 1.0. The series present on this chart are given equal space to display themselves. Series spacing is the percentage of that space that they use to do so.	double	.seriesSpacing	Appearance
Show Domain Axis	Sets whether or not the domain axis is visible.	boolean	.domainAxisVisible	Domain Axis
Show Range Axis	Sets whether or not the range axis is visible.	boolean	.rangeAxisVisible	Range Axis
Time Style	The style to display times of day. For international support.	int	.timeStyle	Appearance
Title Color	Color of the chart title. See Color Selector.	Color	.titleColor	Appearance
Title Font	Font on the chart title.	Font	.titleFont	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting
This component does not have scripting functions associated with it.

Extension Functions

The following feature is new in Ignition version **8.0.16**
Click here to check out the other new features

- Description

  Provides an opportunity to perform further chart configuration via scripting.

- Parameters

  Component self- A reference to the component that is invoking this function.

  JFreeChart chart- A JFreeChart object. Refer to the JFreeChart documentation for API details.

- Return

  Nothing

- Description

  Return a formatted tool tip String

- Parameters

  Component self- A reference to the component that is invoking this function.

  int seriesIndex-The series index corresponding to the column in the series dataset.

  int selectedTimeStamp-The time stamp corresponding to the x value of the displayed tooltip. The time stamp is the number of seconds since the epoch.

  int timeDiff-The width of the current status interval measured in seconds since the epoch.

  int seletedStatus-The status value corresponding to the x value of the displayed tooltip.

  PyDataset data-The series dataset as a PyDataset.

  PyDataset properties-The series properties dataset as a PyDataset.

  string defaultString-The default tooltip string.

- Return

  String defaultString

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

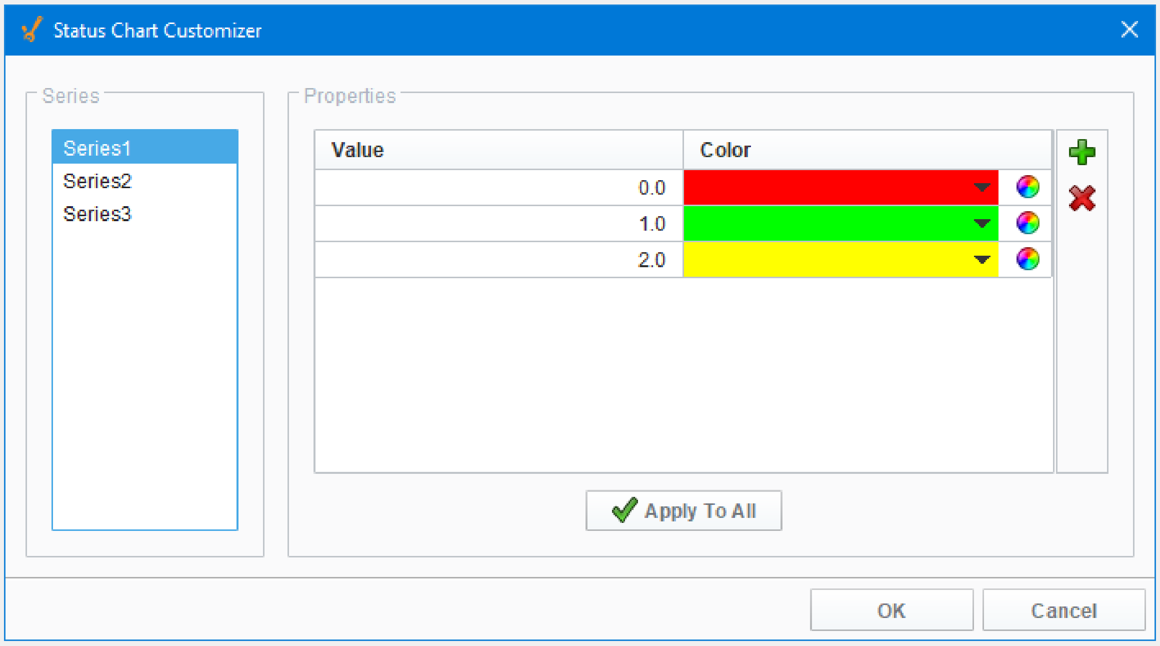
.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

The Status Chart component has its own customizer, used to set a number-to-color mapping for each series in the **Series Data** property.

Status Chart Customizer - Property Description

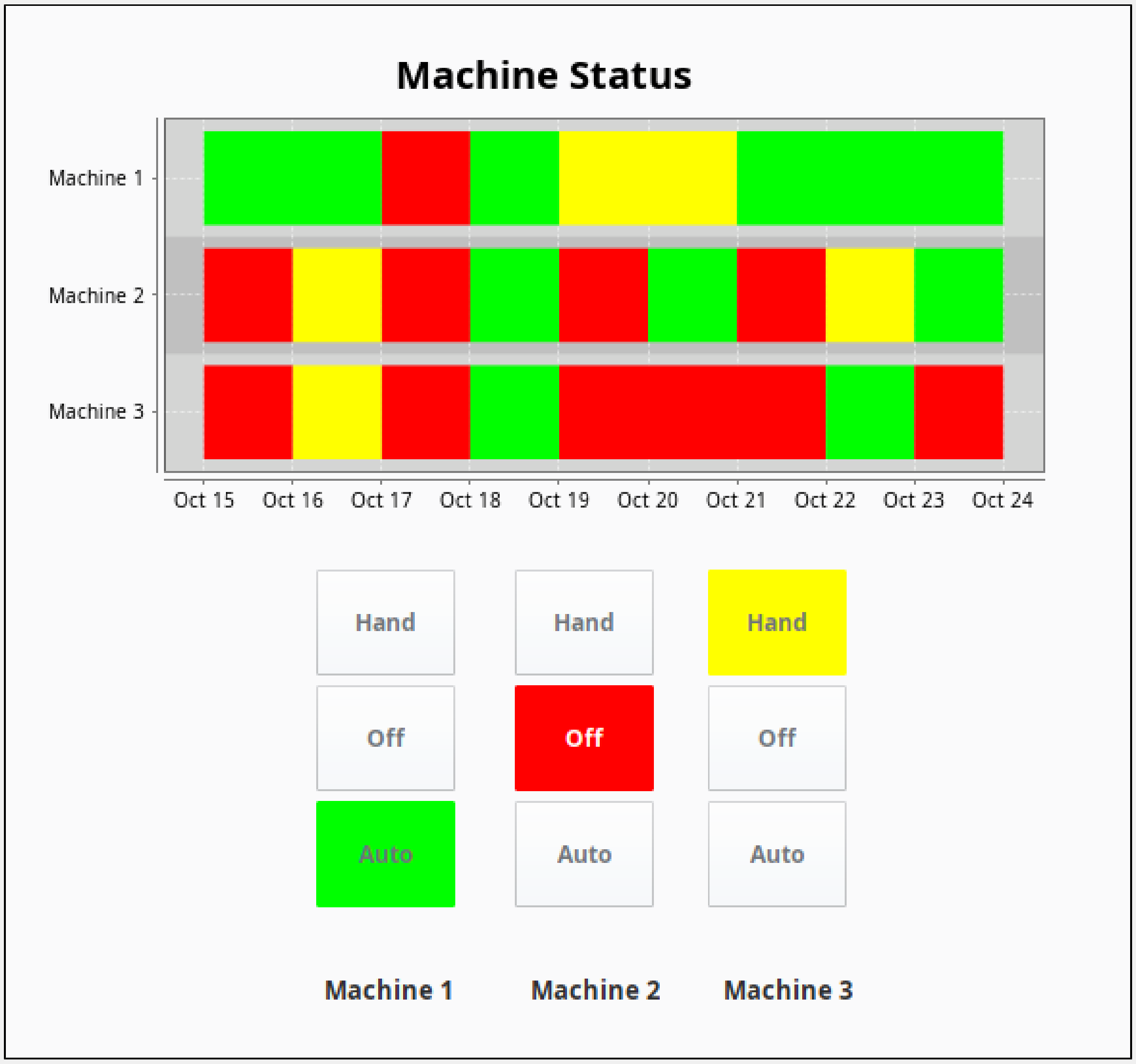
Property	Description
Series	Selectable list of all objects in the Series Data property. • **Wide** format: Each non-timestamp column. • **Tall** format: each unique value in the Name column.
Properties Table	The number-to-color mapping for the selected Series.
Value	A numeric value to match against.
Color	The color to display for the given value.
Apply To All	Set all of the Series mappings to the currently selected mapping.



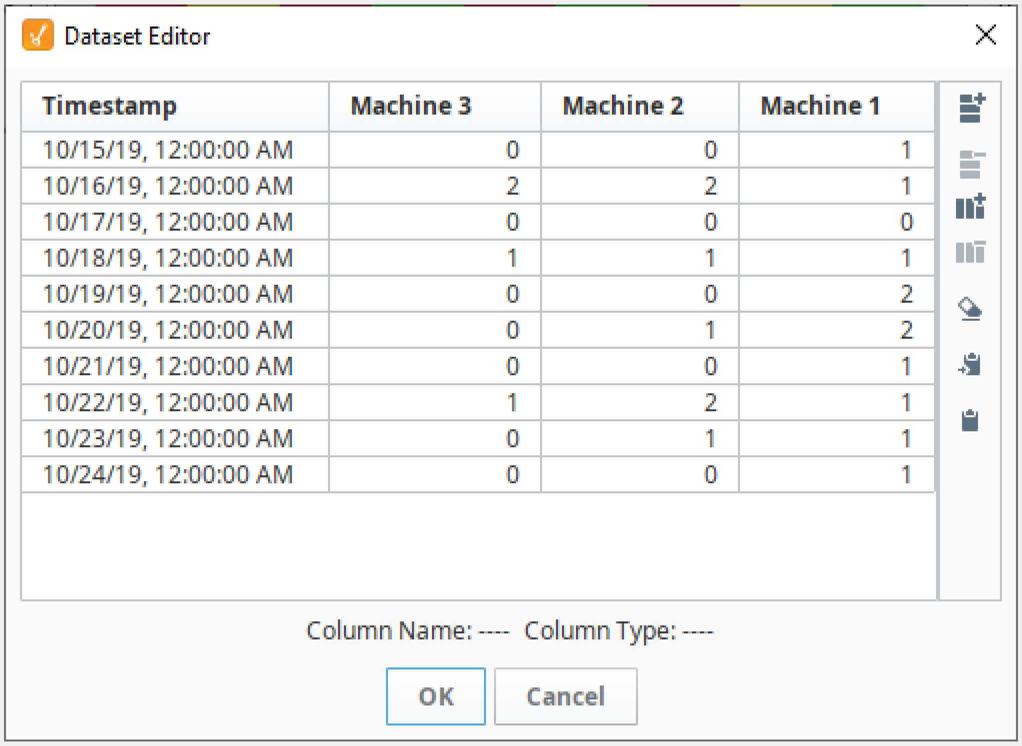
Value	Color
0.0	🟥
1.0	🟩
2.0	🟨

Examples

This example uses the Status Chart to display the state of each of the three machines over consecutive days using the Muli-State button. Tag History was turned on to record history HOA values. The Series Data property's dataset populates the Status Chart using a Tag History Binding. You can view the raw data by clicking on the Dataset Viewer icon to the right of the Series Data property. Each color represents a state for the machine and can be set in the Series Properties Data property. This example also has the raw data in the code block in case you want to try it for yourself.



Series Data - Dataset Viewer



Timestamp	Machine 3	Machine 2	Machine 1
10/15/19, 12:00:00 AM	0	0	1
10/16/19, 12:00:00 AM	2	2	1
10/17/19, 12:00:00 AM	0	0	0
10/18/19, 12:00:00 AM	1	1	1
10/19/19, 12:00:00 AM	0	0	2
10/20/19, 12:00:00 AM	0	1	2
10/21/19, 12:00:00 AM	0	0	1
10/22/19, 12:00:00 AM	1	2	1
10/23/19, 12:00:00 AM	0	1	1
10/24/19, 12:00:00 AM	0	0	1

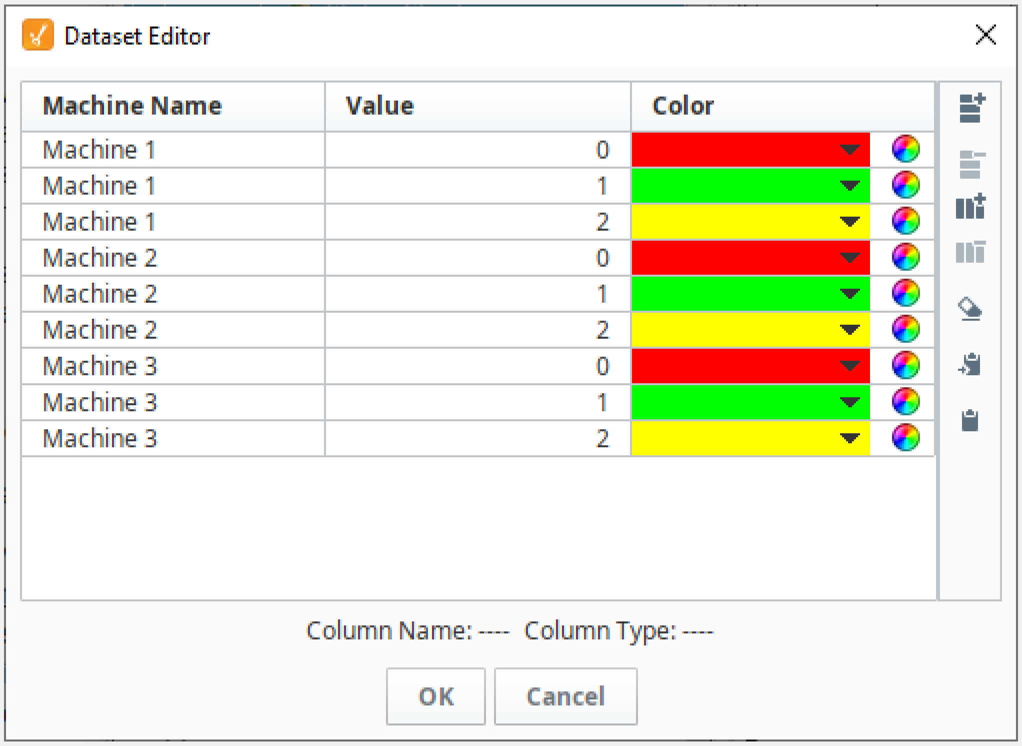
Column Name: ---- Column Type: ----

Series Raw Data

```
"#NAMES"
"Timestamp","Machine 3","Machine 2","Machine 1"
"#TYPES"
"date","I","I","I"
"#ROWS","10"
"2008-10-15 00:00:00.000","0","0","1"
"2008-10-16 00:00:00.000","2","2","1"
"2008-10-17 00:00:00.000","0","0","0"
"2008-10-18 00:00:00.000","1","1","1"
"2008-10-19 00:00:00.000","0","0","2"
"2008-10-20 00:00:00.000","0","1","2"
"2008-10-21 00:00:00.000","0","0","1"
"2008-10-22 00:00:00.000","1","2","1"
"2008-10-23 00:00:00.000","0","1","1"
"2008-10-24 00:00:00.000","0","0","1"
```

Series Properties Data - Dataset Viewer

Each machine has three states, and each of the three states (i.e., HOA) have different colors assigned representing a different state.



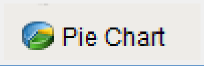
Machine Name	Value	Color
Machine 1	0	🟥
Machine 1	1	🟩
Machine 1	2	🟨
Machine 2	0	🟥
Machine 2	1	🟩
Machine 2	2	🟨
Machine 3	0	🟥
Machine 3	1	🟩
Machine 3	2	🟨

Column Name: ---- Column Type: ----

Series Properties Raw Data

```
"#NAMES"
"SeriesName","Value","Color"
"#TYPES"
"str","I","clr"
"#ROWS","9"
"Series1","0","color(255,0,0,255)"
"Series1","1","color(0,255,0,255)"
"Series1","2","color(255,255,0,255)"
"Series2","0","color(255,0,0,255)"
"Series2","1","color(0,255,0,255)"
"Series2","2","color(255,255,0,255)"
"Series3","0","color(255,0,0,255)"
"Series3","1","color(0,255,0,255)"
"Series3","2","color(255,255,0,255)"
```

Vision - Pie Chart

General
Apples Bananas Kiwis Or Grapefruit
Component Palette Icon:


Description

The Pie Chart component displays a familiar-looking pie chart. A Pie Chart displays a list of named items, each of which has a value that is part of a total. The total is the sum of the value of each item. The key to the Pie Chart component is the Data property, which contains the items that will be displayed as pie wedges. Typically, this dataset will be bound to a SQL Query Binding in Vision to pull dynamic data out of an external database.

Extract Order

Similar to other charts, the pie chart can actually accept data in two formats. You can tell the pie chart which format to use via its Extract Order property. The two extract orders are By Column or By Row. The following table shows the two styles for the data that created the pie chart in the screenshot.

By Column

Label	Value
Grapefruit	7
Apples	15
Bananas	56
Kiwis	19

By Row

Grapefruit	Apples	Bananas	Kiwis
7	15	56	19

Labels

In addition to the color-coded legend, the pie chart can annotate each wedge with a label. The format of the label is controlled via the Label Format property.

For example, the format string used in the screenshot is " {0} = {2} ({3}) " This is a pattern string that uses the following placeholders:

- {0} - the item label
- {1} - the item value
- {2} - the item percentage

Properties

Name	Description	Property Type	S
3D Depth Factor	The depth of a 3D pie as a factor of the chart height.	double	. c t
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. The border is unaffected by rotation.	Border	.
Chart Title	An optional title that will appear at the top of the chart.	String	.
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	. c c
Data	The data driving the chart.	Dataset	.

Enforce Circularity?	If true, the pie cannot be an oval, even if the overall chart is.	boolean	.
Extract Order	Controls whether or not a pie plot views columns as pies, or rows.	int	. e c
Foreground Transparency	The transparency of the pie (useful for 3D pies). Valid values are between 0 (0% opacity) and 1 (100% opacity).	double	. f c
Label Font	The font for labels items, if there are labels.	Font	. l
Label Format	Formatting String. '{0}' is the wedge name, '{1}' is the value, '{2}' is the percent.	String	. l r
Labels?	Should labels be displayed near sections?	boolean	.
Legend Font	The font for legend items, if there is a legend.	Font	. l r
Legend?	Should there be an item legend below the chart?	boolean	.
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	. t x
Name	The name of this component.	String	.
Outline Colors	The colors to use for the pie wedge outlines. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color[]	. c l
Outline Visible	The following feature is new in Ignition version **8.0.16** Click here to check out the other new features Whether to display an outline around the pie chart.	boolean	. c i
Outline Stroke	The width for the section outline stroke.	float	. c c
Plot Background	The background color for all plots, unless they override it. See Color Selector.	Color	. p g
Plot Insets	The following feature is new in Ignition version **8.0.16** Click here to check out the other new features The padding to use around the actual plot rendering area.	int	. p
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.
Rotation	Draw the wedges clockwise or counter-clockwise from the starting angle?	int	.
Section Colors	The colors to use for the pie wedge fills. See Color Selector.	Color[]	. s c
Selected Wedge	The currently selected wedge. (Read only. Usable in bindings and scripting.)	String	. s [

Selection Enabled?	If true, the user will be able to select wedges on the chart. The selected wedge will be highlighted, and the "selectedData" property will reflect it.	boolean	. s E
Selection Highlight Color	The color of the selection highlight. See Color Selector.	Color	. s H C
Selection Highlight Width	The line width of the selection highlight.	float	. s H W
Starting Angle	The start angle to draw the pie wedges.	int	. s
Style	Style of pie chart, standard, 3D, or ring.	int	.
Title Font	The font for the chart's title.	Font	.
Tooltip Format	Formatting String. '{0}' is the wedge name, '{1}' is the value, '{2}' is the percent.	String	. t r
Tooltips?	Should tooltips be displayed when the mouse hovers over sections?	boolean	.
Visible	If disabled, the component will be hidden.	boolean	.
Deprecated Properties			
Data Quality	The data quality code for any Tag bindings on this component.	int	. c t

Scripting

The following feature is new in Ignition version **8.0.16**
Click here to check out the other new features

- Description

  Provides an opportunity to perform further chart configuration via scripting.

- Parameters

  Component self- A reference to the component that is invoking this function.

  JFreeChart chart- A JFreeChart object. Refer to the JFreeChart documentation for API details.

- Return

  Nothing

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

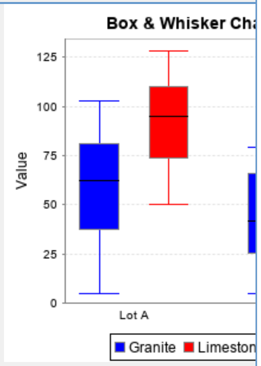
- Vision Component Customizers

Examples

Code Snippet

```
#The following code will print named and value of the selected wedge to the console.
#Alternatively, this can be used to write to a custom property of a table that is used to
create the 'Where' clause of a SQL query that populates a table.

selectedWedge = event.source.selectedData
print selectedWedge
```

Vision - Box and Whisker Chart

General

Component Palette Icon:
Box and Whisker C

Description
A Box and Whisker chart displays pertinent statistical information about sets of data. Each box represents a set of numbers. The upper and lower bounds of the box represent the 1st and 3rd quartiles. The line inside the box represents the median. The extends of the "whiskers" represent the max and min outliers. For a more detailed description, see http://mathworld.wolfram.com/Box-and-WhiskerPlot.html. The configuration for setting up a box and whisker chart, like most charts, is populating the Data property. The dataset for a box and whisker chart contains sets of numbers. Each column defines a series of values, for which a "box" will be calculated. The column headers define the name for the box. You may also have an optional first column that is a String column, which can break up the series into categories. To learn more, refer to Box and Whisker Chart.

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Category Axis Title	A text label to display on the category axis.	String	.categoryAxisTitle	Appearance
Chart Title	An optional title that will appear at the top of the chart.	String	.title	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Data	The data driving the chart.	Dataset	.data	Data
Fill Boxes?	Fill the boxes with their color?	boolean	.fillBoxes	Appearance
Font	Font of text on this component.	Font	.font	Appearance
Legend?	Show a legend on the chart?	boolean	.legend	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Plot Background	The background color for the plot. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.plotBackground	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Series Colors	The colors to paint each box in a series. See Color Selector.	Color[]	.seriesColors	Appearance
Tooltips?	Show tooltips on tasks?	boolean	.tooltips	Behavior
Value Axis Title	A text label to display on the value axis.	String	.valueAxisTitle	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

⚠ This event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
. property Name	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

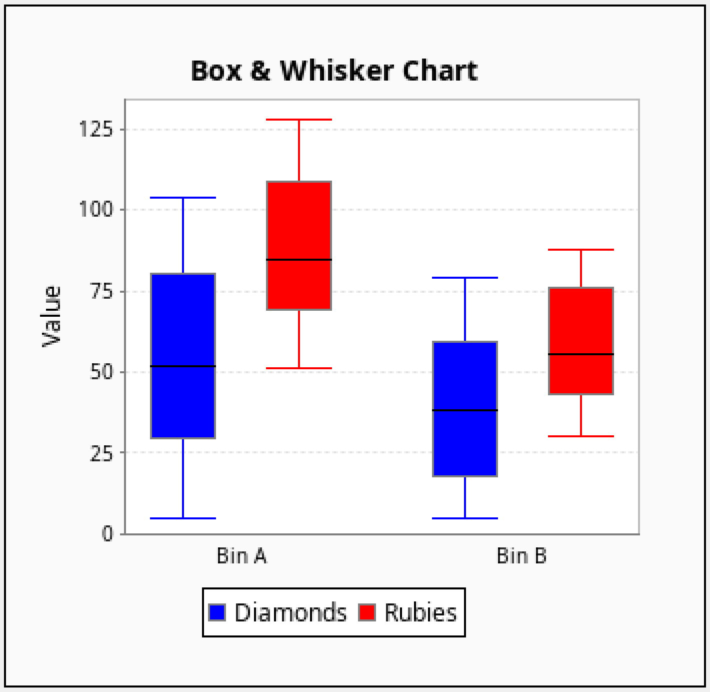
Customizers

This component does not have any custom properties.

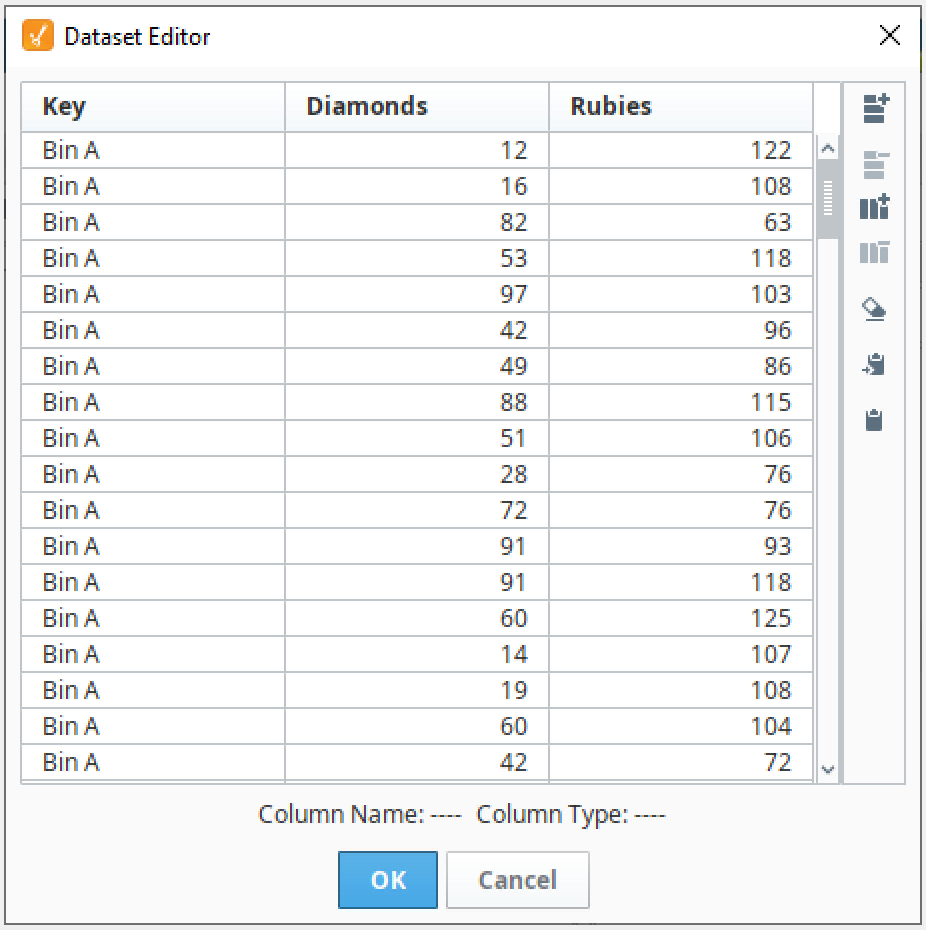
Example

This example uses the Box & Whisker Chart to display information about two sets of data, Bin A and Bin B, and both contain Diamonds and Rubies. The Box and Whisker Chart is displaying a large amount of data as you can tell from looking at the code block below. It displays high, low, and median values which is where 50% of the data falls. The dataset contains all the raw data and calculates the upper and lower bounds of each box which are the solid colored boxes, horizontal line inside the box which represents the median value, and the whiskers which represent the minimum and maximum values which are outside the solid color boxes.

The dataset populates the chart. You can view the data in the dataset by clicking on the dataset icon. This example also has the raw data in the code block in case you want to try it for yourself.



Box and Whisker - Dataset Editor



Dataset Editor

Key	Diamonds	Rubies
Bin A	12	122
Bin A	16	108
Bin A	82	63
Bin A	53	118
Bin A	97	103
Bin A	42	96
Bin A	49	86
Bin A	88	115
Bin A	51	106
Bin A	28	76
Bin A	72	76
Bin A	91	93
Bin A	91	118
Bin A	60	125
Bin A	14	107
Bin A	19	108
Bin A	60	104
Bin A	42	72

Column Name: ---- Column Type: ----

Box and Whisker Raw Data

```
"#NAMES"
"Key","Diamonds","Rubies"
"#TYPES"
"str","I","I"
"#ROWS","200"
"Bin A","12","122"
"Bin A","16","108"
"Bin A","82","63"
"Bin A","53","118"
"Bin A","97","103"
"Bin A","42","96"
"Bin A","49","86"
"Bin A","88","115"
"Bin A","51","106"
"Bin A","28","76"
"Bin A","72","76"
"Bin A","91","93"
"Bin A","91","118"
"Bin A","60","125"
"Bin A","14","107"
"Bin A","19","108"
"Bin A","60","104"
"Bin A","42","72"
"Bin A","97","69"
"Bin A","99","69"
"Bin A","95","119"
"Bin A","76","92"
"Bin A","84","101"
"Bin A","27","99"
"Bin A","33","101"
"Bin A","12","53"
"Bin A","90","83"
```

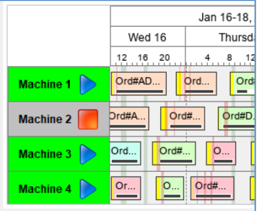
```
"Bin A","78","61"
"Bin A","101","61"
"Bin A","50","84"
"Bin A","93","126"
"Bin A","15","85"
"Bin A","43","117"
"Bin A","37","57"
"Bin A","79","81"
"Bin A","5","53"
"Bin A","65","75"
"Bin A","94","76"
"Bin A","79","80"
"Bin A","94","97"
"Bin A","45","58"
"Bin A","104","77"
"Bin A","29","74"
"Bin A","22","89"
"Bin A","20","115"
"Bin A","61","73"
"Bin A","5","70"
"Bin A","12","117"
"Bin A","36","118"
"Bin A","42","85"
"Bin A","92","87"
"Bin A","100","57"
"Bin A","42","72"
"Bin A","102","114"
"Bin A","7","90"
"Bin A","75","112"
"Bin A","36","92"
"Bin A","84","105"
"Bin A","80","69"
"Bin A","46","67"
"Bin A","48","77"
"Bin A","100","62"
"Bin A","32","72"
"Bin A","11","113"
"Bin A","23","127"
"Bin A","53","95"
"Bin A","67","108"
"Bin A","45","54"
"Bin A","47","51"
"Bin A","62","68"
"Bin A","86","72"
"Bin A","80","70"
"Bin A","77","113"
"Bin A","103","126"
"Bin A","21","57"
"Bin A","22","128"
"Bin A","11","77"
"Bin A","48","57"
"Bin A","73","118"
"Bin A","35","125"
"Bin A","57","52"
"Bin A","34","124"
"Bin A","66","68"
"Bin A","81","79"
"Bin A","43","78"
"Bin A","16","53"
"Bin A","81","109"
"Bin A","64","53"
"Bin A","94","59"
"Bin A","67","95"
"Bin A","67","57"
"Bin A","27","115"
"Bin A","18","120"
"Bin A","17","77"
"Bin A","56","87"
"Bin A","32","124"
"Bin A","30","57"
"Bin A","5","78"
"Bin A","68","82"
"Bin A","31","58"
"Bin B","66","74"
"Bin B","64","85"
```

```
"Bin B","29","86"
"Bin B","34","85"
"Bin B","16","36"
"Bin B","42","68"
"Bin B","26","33"
"Bin B","9","85"
"Bin B","27","74"
"Bin B","42","58"
"Bin B","6","72"
"Bin B","14","79"
"Bin B","40","54"
"Bin B","12","42"
"Bin B","21","34"
"Bin B","6","73"
"Bin B","46","43"
"Bin B","39","36"
"Bin B","67","42"
"Bin B","55","71"
"Bin B","42","42"
"Bin B","34","41"
"Bin B","24","54"
"Bin B","20","42"
"Bin B","66","75"
"Bin B","12","80"
"Bin B","75","84"
"Bin B","43","57"
"Bin B","62","50"
"Bin B","12","37"
"Bin B","65","32"
"Bin B","11","60"
"Bin B","5","32"
"Bin B","21","58"
"Bin B","36","53"
"Bin B","12","79"
"Bin B","37","78"
"Bin B","24","30"
"Bin B","73","87"
"Bin B","53","70"
"Bin B","70","82"
"Bin B","6","36"
"Bin B","65","72"
"Bin B","54","88"
"Bin B","10","47"
"Bin B","10","70"
"Bin B","63","41"
"Bin B","12","84"
"Bin B","77","47"
"Bin B","64","72"
"Bin B","72","84"
"Bin B","68","49"
"Bin B","23","88"
"Bin B","78","63"
"Bin B","40","57"
"Bin B","14","76"
"Bin B","7","45"
"Bin B","77","60"
"Bin B","19","86"
"Bin B","52","50"
"Bin B","64","88"
"Bin B","57","37"
"Bin B","50","69"
"Bin B","45","85"
"Bin B","27","51"
"Bin B","28","56"
"Bin B","54","54"
"Bin B","43","32"
"Bin B","11","68"
"Bin B","44","85"
"Bin B","22","55"
"Bin B","74","76"
"Bin B","51","83"
"Bin B","50","42"
"Bin B","65","77"
"Bin B","22","43"
"Bin B","34","36"
```

```
"Bin B","29","46"
"Bin B","33","51"
"Bin B","39","55"
"Bin B","17","43"
"Bin B","35","44"
"Bin B","50","31"
"Bin B","10","49"
"Bin B","78","38"
"Bin B","15","31"
"Bin B","45","78"
"Bin B","79","76"
"Bin B","22","55"
"Bin B","37","49"
"Bin B","10","50"
"Bin B","40","76"
"Bin B","40","44"
"Bin B","17","45"
"Bin B","16","87"
"Bin B","7","41"
"Bin B","67","77"
"Bin B","70","35"
"Bin B","69","52"
"Bin B","30","71"
```

Vision - Equipment Schedule

General



Component Palette Icon:

Equipment Schedu

Description

The Equipment Schedule view is a mix between the status chart, gantt chart, and a calendar view. It conveys a lot of information about equipment, including current status, production schedule, production status, scheduled and unexpected downtime.

The equipment schedule is powered by four datasets. Information is retrieved from the datasets by column name, case-insensitive. The order of the columns is not important. Optional columns may be omitted.

The "Items" Dataset

Describes the "items" or "cells" to display schedules for. Each entry in this dataset will become a row of the chart.

Name	Type	Optional	Description
ID	Any	N	The identifier for this item. May be any type, will referenced by each entry in the Scheduled Events dataset.
Label	String	N	The text to display in the header.
Foreground	Color	Y	Text color.
Background	Color	Y	Background color.
StatusImagePath	String	Y	A path to an image to display to the right of the header label.

The "Scheduled Items" Dataset

Lists the scheduled events for each item described in the "Items" dataset. Each scheduled event can have a colored lead, or change-over time, a label, a background color, and a progress.

Name	Type	Optional	Description
EventId	String	Y	An identifier for the event, used for event selection.
ItemId	Any	N	The ID of the item to correlate this event with. If no such item is found, the event won't be shown.
Label	String	N	The text ot display in the event's box.
StartDate	Date	N	The start-time for the event.
EndDate	Date	N	The end-time for the event.
Foreground	Color	Y	The text color of the event.
Background	Color	Y	The background color of the event.
LeadTime	Integer	Y	Time, in seconds, to display as lead time.
LeadColor	Color	Y	The color for the lead time, if any.
PctDone	Number	Y	A value from 0 to 100 to be displayed as a progress bar, use -1 to hide progress bar.

The "Downtime" Dataset

Entries in this dataset will be displayed as simple colored overlays on top of the events, correlated against an item defined in the "Items" dataset.

Name	Type	Optional	Description
ItemId	Any	N	The ID of the item to correlate this downtime event with. If no such item is found, the downtime event won't be shown.
StartDate	Date	N	The start-time for the downtime event.
EndDate	Date	N	The end-time for the downtime event.
Color	Color	Y	The color to use, typically transparent.
Layer	Integer	Y	0 or 1, with 0 meaning that the rectangle gets painted below the events, and 1 means it will be painted above the events.

The "Breaks" Dataset

Entries in this dataset will be displayed as colored underlays beneath all events.

Name	Type	Optional	Description
StartDate	Date	N	The start-time for the break event.
EndDate	Date	N	The end-time for the break event.
Color	Color	Y	The color to use.

Properties

Name	Description	Property Type	Scripting	Category

Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Break Events	Scheduled breaks, which will appear as downtime for all items.	Dataset	.breakEvents	Data
Current Time Color	The color of the current time indicator. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.nowColor	Appearance
Downtime Events	Downtime events correlated to a specific item.	Dataset	.downtimeEvents	Data
Drag Enabled	Controls whether or not scheduled events can be dragged for rescheduling.	boolean	.dragEnabled	Behavior
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
End Date	The end of the time range to display.	Date	.endDate	Data
Event Border	The normal border for a scheduled event.	Border	.eventBorder	Appearance
Event Font	The font to use for the event labels.	Font	.eventFont	Appearance
Event Margin	The margin to leave visible above and below a scheduled event.	int	.scheduledEventMargin	Appearance
Header Background	The color of the background for the header timeline. See Color Selector.	Color	.headerBackground	Appearance
Header Font	The font of the text in the header timeline.	Font	.headerFont	Appearance
Header Item Font	The font to use for the header items' labels.	Font	.itemFont	Appearance
Header Text Color	The color of the text in the header timeline. See Color Selector.	Color	.headerTextColor	Appearance
Items	The cells, or equipment items, to have their schedules displayed.	Dataset	.items	Data
Line Color	The color of separating lines in the schedule.	Color	.lineColor	Appearance
Name	The name of this component.	String	.name	Common
Progress Bar Background	The background color for the event progress bars. See Color Selector.	Color	.progressBackground	Appearance
Progress Bar Border	The border color for the event progress bars. See Color Selector.	Color	.progressBorder	Appearance

Progress Bar Fill	The color for 'done' portion the event progress bars. See Color Selector.	Color	.progressFill	Appearance
Resize Enabled	Controls whether or not scheduled events resized for duration changes.	boolean	.resizeEnabled	Behavior
Row Height	The height of each event's schedule row.	int	.lineHeight	Appearance
Schedule Background	The background color of the schedule area. See Color Selector.	Color	.scheduleBackground	Appearance
Scheduled Events	The scheduled events for all configured items.	Dataset	.scheduledEvents	Data
Selected Event Border	The border for a selected scheduled event.	Border	.selectedEventBorder	Appearance
Selected Event ID	The ID of the selected event.	String	.selectedEvent	Data
Start Date	The beginning of the time range to display.	Date	.startDate	Data
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

- Description

  Called when the user drags a segment on the schedule background.

- Parameters

  Component self - A reference to the component that is invoking this function.

  int itemID - The ID of the equipment item of the row where the user dragged.

  Date startDate - The datetime corresponding to where the user started dragging.

  Date endDate - The datetime corresponding to where the user ended dragging.

  Event Object event - The mouse event.

- Return

  Nothing

- Description

  Called when the user clicks on a scheduled event. Use event.clickCount to detect double clicks.

- Parameters

  Component self - A reference to the component that is invoking this function.

  int itemID - The ID of the equipment item of the event that was clicked on.

  int eventId - The ID of the event that was clicked on.

  Event Object event - The mouse event.

- Return

  Nothing

- Description

  Called when the user drags and drops a scheduled event. It is up to this script to actually alter the underlying data to reflect the schedule change.

- Parameters

  Component self - A reference to the component that is invoking this function.

  int eventId - The ID of the scheduled event that was moved.

  int oldItemId - The ID of the item this event was originally correlated against.

  int newItemId - The ID of the item whose schedule the event was dropped on.

  Date oldStartDate - The original starting datetime of the event.

  Date newStartDate - The new starting datetime of the event.

  Date newEndDate - The new ending datetime of the event.

- Return

  Nothing

- Description

  Called when the user right-clicks on a scheduled event. This would be the appropriate time to create and display a popup menu.

- Parameters

  Component self - A reference to the component that is invoking this function.

  int itemId - The ID of the equipment item of the event that was right-clicked on.

  int eventId - The ID of the event that was right-clicked on.

  Event Object event - The mouse event that caused the popup trigger.

- Return

  Nothing

- Description

  Called when the user drags the edge of an event to resize its time span. It is up to this script to actually alter the underlying data to reflect the schedule change.

- Parameters

  Component self - A reference to the component that is invoking this function.

  int eventId - The ID of the scheduled event that was resized.

  int itemId - The ID of the item this event is correlated against.

  Date oldStartDate - The original starting datetime of the event.

  Date oldEndData - The original ending datetime of the event.

  Date newStartDate - The new starting datetime of the event.

  Date newEndDate - The new ending datetime of the event.

- Return

  Nothing

- Description

  Called when the user right-clicks outside of an event. This would be the appropriate time to create and display a popup menu.

- Parameters

  Component self - A reference to the component that is invoking this function.

  int itemId - The item ID of the equipment line that was clicked on (if any).

  Event Object event - The mouse event that caused the popup trigger.

- Return

  Nothing

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
. property Name	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

This event is deprecated. Please use the onEventDropped extension function.

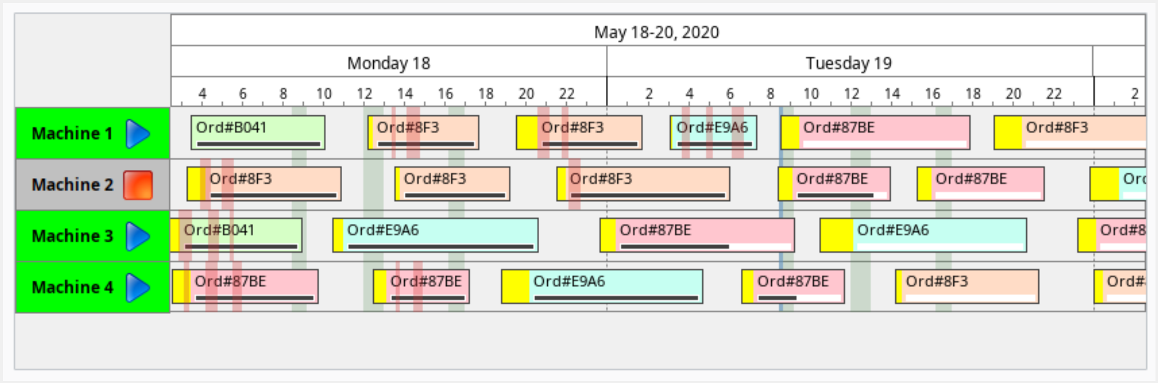
Customizers

- Vision Component Customizers

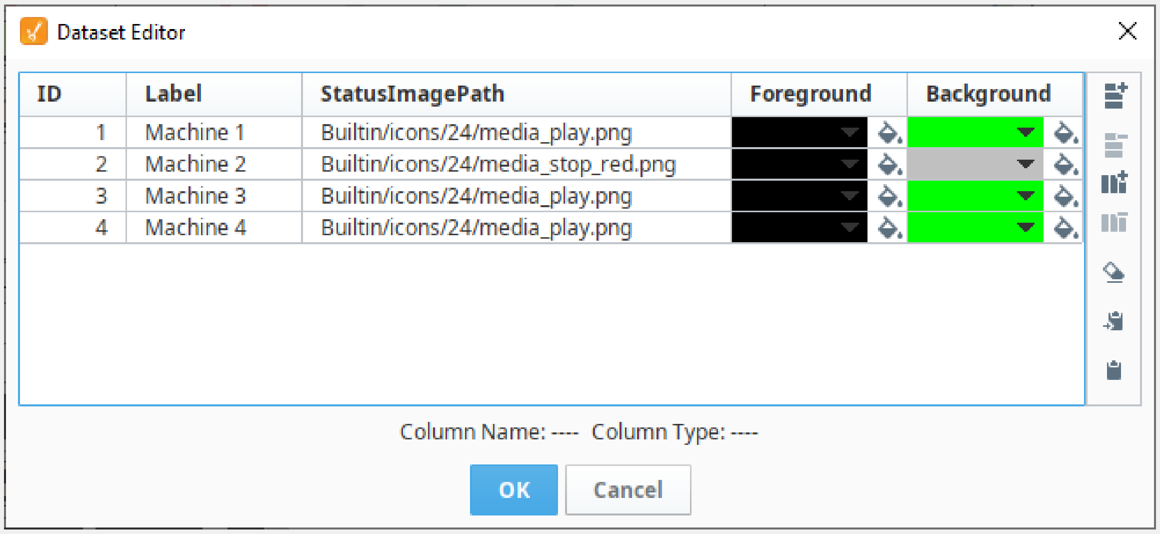
Examples

The Equipment Schedule contains a lot information about Machines 1-4 from May 18 through May 20 such as equipment status, the production schedule, production status, and schedule and unscheduled downtime. It provides a view into the status of equipment on the production floor in realtime and scheduled work planned for three days. It uses four datasets: Items, Scheduled Events, Downtime Events, and Break Events. Each dataset is shown below with it's associated raw data.

You'll notice each piece of equipment has a lead time or change-over time, a unique Order number for the run, background color and displays a progress bar. Equipment downtime entries are displayed as colored overlays on top of the events. Break events with a start and end time are displayed as colored underlays beneath the events.



Equipment Schedule - Items Dataset



Dataset Editor

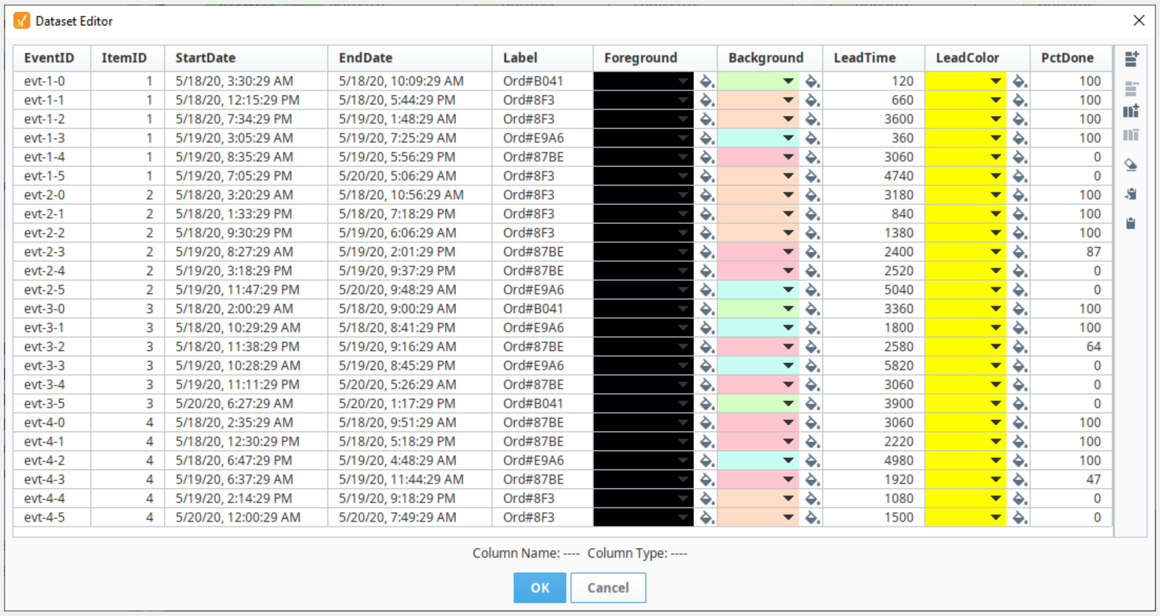
ID	Label	StatusImagePath	Foreground	Background
1	Machine 1	Builtin/icons/24/media_play.png	⬛	🟩
2	Machine 2	Builtin/icons/24/media_stop_red.png	⬛	⬜
3	Machine 3	Builtin/icons/24/media_play.png	⬛	🟩
4	Machine 4	Builtin/icons/24/media_play.png	⬛	🟩

Column Name: ---- Column Type: ----

Equipment Schedule - Items Raw Data

```
"#NAMES"
"ID","Label","StatusImagePath","Foreground","Background"
"#TYPES"
"I","str","str","clr","clr"
"#ROWS","4"
"1","Machine 1","Builtin/icons/24/media_play.png","color(0,0,0,255)","color(0,255,0,255)"
"2","Machine 2","Builtin/icons/24/media_stop_red.png","color(0,0,0,255)","color
(192,192,192,255)"
"3","Machine 3","Builtin/icons/24/media_play.png","color(0,0,0,255)","color(0,255,0,255)"
"4","Machine 4","Builtin/icons/24/media_play.png","color(0,0,0,255)","color(0,255,0,255)"
```

Equipment Schedule - Scheduled Events Dataset



Dataset Editor

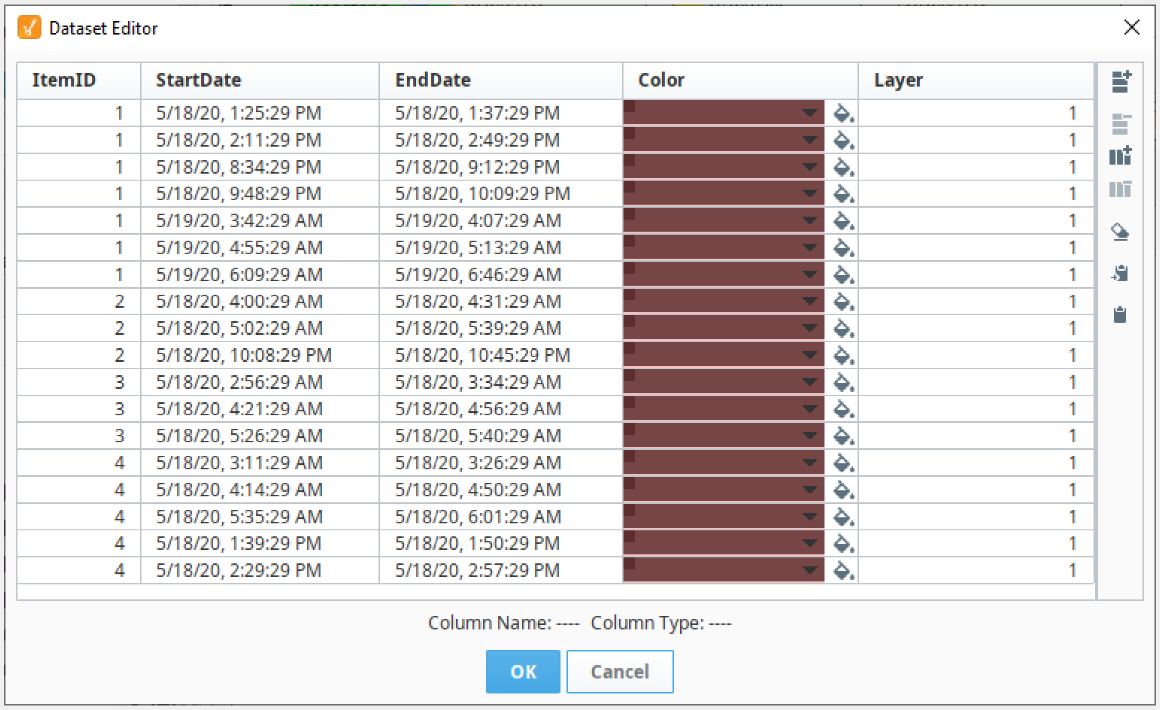
EventID	ItemID	StartDate	EndDate	Label	Foreground	Background	LeadTime	LeadColor	PctDone
evt-1-0	1	5/18/20, 3:30:29 AM	5/18/20, 10:09:29 AM	Ord#B041	⬛	🟩	120	🟨	100
evt-1-1	1	5/18/20, 12:15:29 PM	5/18/20, 5:44:29 PM	Ord#8F3	⬛	🟧	660	🟨	100
evt-1-2	1	5/18/20, 7:34:29 PM	5/19/20, 1:48:29 AM	Ord#8F3	⬛	🟧	3600	🟨	100
evt-1-3	1	5/19/20, 3:05:29 AM	5/19/20, 7:25:29 AM	Ord#E9A6	⬛	🟦	360	🟨	100
evt-1-4	1	5/19/20, 8:35:29 AM	5/19/20, 5:56:29 PM	Ord#87BE	⬛	🟥	3060	🟨	0
evt-1-5	1	5/19/20, 7:05:29 PM	5/20/20, 5:06:29 AM	Ord#8F3	⬛	🟧	4740	🟨	0
evt-2-0	2	5/18/20, 3:20:29 AM	5/18/20, 10:56:29 AM	Ord#8F3	⬛	🟧	3180	🟨	100
evt-2-1	2	5/18/20, 1:33:29 PM	5/18/20, 7:18:29 PM	Ord#8F3	⬛	🟧	840	🟨	100
evt-2-2	2	5/18/20, 9:30:29 PM	5/19/20, 6:06:29 AM	Ord#8F3	⬛	🟧	1380	🟨	100
evt-2-3	2	5/19/20, 8:27:29 AM	5/19/20, 2:01:29 PM	Ord#87BE	⬛	🟥	2400	🟨	87
evt-2-4	2	5/19/20, 3:18:29 PM	5/19/20, 9:37:29 PM	Ord#87BE	⬛	🟥	2520	🟨	0
evt-2-5	2	5/19/20, 11:47:29 PM	5/20/20, 9:48:29 AM	Ord#E9A6	⬛	🟦	5040	🟨	0
evt-3-0	3	5/18/20, 2:00:29 AM	5/18/20, 9:00:29 AM	Ord#B041	⬛	🟩	3360	🟨	100
evt-3-1	3	5/18/20, 10:29:29 AM	5/18/20, 8:41:29 PM	Ord#E9A6	⬛	🟦	1800	🟨	100
evt-3-2	3	5/18/20, 11:38:29 PM	5/19/20, 9:16:29 AM	Ord#87BE	⬛	🟥	2580	🟨	64
evt-3-3	3	5/19/20, 10:28:29 AM	5/19/20, 8:45:29 PM	Ord#E9A6	⬛	🟦	5820	🟨	0
evt-3-4	3	5/19/20, 11:11:29 PM	5/20/20, 5:26:29 AM	Ord#87BE	⬛	🟥	3060	🟨	0
evt-3-5	3	5/20/20, 6:27:29 AM	5/20/20, 1:17:29 PM	Ord#B041	⬛	🟩	3900	🟨	0
evt-4-0	4	5/18/20, 2:35:29 AM	5/18/20, 9:51:29 AM	Ord#87BE	⬛	🟥	3060	🟨	100
evt-4-1	4	5/18/20, 12:30:29 PM	5/18/20, 5:18:29 PM	Ord#87BE	⬛	🟥	2220	🟨	100
evt-4-2	4	5/18/20, 6:47:29 PM	5/19/20, 4:48:29 AM	Ord#E9A6	⬛	🟦	4980	🟨	100
evt-4-3	4	5/19/20, 6:37:29 AM	5/19/20, 11:44:29 AM	Ord#87BE	⬛	🟥	1920	🟨	47
evt-4-4	4	5/19/20, 2:14:29 PM	5/19/20, 9:18:29 PM	Ord#8F3	⬛	🟧	1080	🟨	0
evt-4-5	4	5/20/20, 12:00:29 AM	5/20/20, 7:49:29 AM	Ord#8F3	⬛	🟧	1500	🟨	0

Column Name: ---- Column Type: ----

Equipment Schedule - Scheduled Events Raw Data

```
"#NAMES"
"EventID","ItemID","StartDate","EndDate","Label","Foreground","Background","LeadTime","
LeadColor","PctDone"
"#TYPES"
"str","I","date","date","str","clr","clr","I","clr","D"
"#ROWS","24"
"evt-1-0","1","2020-05-18 03:30:29.002","2020-05-18 10:09:29.002","Ord#B041","color
(0,0,0,255)","color(214,255,198,255)","120","color(255,255,0,255)","100.0"
"evt-1-1","1","2020-05-18 12:15:29.002","2020-05-18 17:44:29.002","Ord#8F3","color
(0,0,0,255)","color(255,220,198,255)","660","color(255,255,0,255)","100.0"
"evt-1-2","1","2020-05-18 19:34:29.002","2020-05-19 01:48:29.002","Ord#8F3","color
(0,0,0,255)","color(255,220,198,255)","3600","color(255,255,0,255)","100.0"
"evt-1-3","1","2020-05-19 03:05:29.002","2020-05-19 07:25:29.002","Ord#E9A6","color
(0,0,0,255)","color(198,255,242,255)","360","color(255,255,0,255)","100.0"
"evt-1-4","1","2020-05-19 08:35:29.002","2020-05-19 17:56:29.002","Ord#87BE","color
(0,0,0,255)","color(255,198,207,255)","3060","color(255,255,0,255)","0.0"
"evt-1-5","1","2020-05-19 19:05:29.002","2020-05-20 05:06:29.002","Ord#8F3","color
(0,0,0,255)","color(255,220,198,255)","4740","color(255,255,0,255)","0.0"
"evt-2-0","2","2020-05-18 03:20:29.002","2020-05-18 10:56:29.002","Ord#8F3","color
(0,0,0,255)","color(255,220,198,255)","3180","color(255,255,0,255)","100.0"
"evt-2-1","2","2020-05-18 13:33:29.002","2020-05-18 19:18:29.002","Ord#8F3","color
(0,0,0,255)","color(255,220,198,255)","840","color(255,255,0,255)","100.0"
"evt-2-2","2","2020-05-18 21:30:29.002","2020-05-19 06:06:29.002","Ord#8F3","color
(0,0,0,255)","color(255,220,198,255)","1380","color(255,255,0,255)","100.0"
"evt-2-3","2","2020-05-19 08:27:29.002","2020-05-19 14:01:29.002","Ord#87BE","color
(0,0,0,255)","color(255,198,207,255)","2400","color(255,255,0,255)","87.0"
"evt-2-4","2","2020-05-19 15:18:29.002","2020-05-19 21:37:29.002","Ord#87BE","color
(0,0,0,255)","color(255,198,207,255)","2520","color(255,255,0,255)","0.0"
"evt-2-5","2","2020-05-19 23:47:29.002","2020-05-20 09:48:29.002","Ord#E9A6","color
(0,0,0,255)","color(198,255,242,255)","5040","color(255,255,0,255)","0.0"
"evt-3-0","3","2020-05-18 02:00:29.002","2020-05-18 09:00:29.002","Ord#B041","color
(0,0,0,255)","color(214,255,198,255)","3360","color(255,255,0,255)","100.0"
"evt-3-1","3","2020-05-18 10:29:29.002","2020-05-18 20:41:29.002","Ord#E9A6","color
(0,0,0,255)","color(198,255,242,255)","1800","color(255,255,0,255)","100.0"
"evt-3-2","3","2020-05-18 23:38:29.002","2020-05-19 09:16:29.002","Ord#87BE","color
(0,0,0,255)","color(255,198,207,255)","2580","color(255,255,0,255)","64.0"
"evt-3-3","3","2020-05-19 10:28:29.002","2020-05-19 20:45:29.002","Ord#E9A6","color
(0,0,0,255)","color(198,255,242,255)","5820","color(255,255,0,255)","0.0"
"evt-3-4","3","2020-05-19 23:11:29.002","2020-05-20 05:26:29.002","Ord#87BE","color
(0,0,0,255)","color(255,198,207,255)","3060","color(255,255,0,255)","0.0"
"evt-3-5","3","2020-05-20 06:27:29.002","2020-05-20 13:17:29.002","Ord#B041","color
(0,0,0,255)","color(214,255,198,255)","3900","color(255,255,0,255)","0.0"
"evt-4-0","4","2020-05-18 02:35:29.002","2020-05-18 09:51:29.002","Ord#87BE","color
(0,0,0,255)","color(255,198,207,255)","3060","color(255,255,0,255)","100.0"
"evt-4-1","4","2020-05-18 12:30:29.002","2020-05-18 17:18:29.002","Ord#87BE","color
(0,0,0,255)","color(255,198,207,255)","2220","color(255,255,0,255)","100.0"
"evt-4-2","4","2020-05-18 18:47:29.002","2020-05-19 04:48:29.002","Ord#E9A6","color
(0,0,0,255)","color(198,255,242,255)","4980","color(255,255,0,255)","100.0"
"evt-4-3","4","2020-05-19 06:37:29.002","2020-05-19 11:44:29.002","Ord#87BE","color
(0,0,0,255)","color(255,198,207,255)","1920","color(255,255,0,255)","47.0"
"evt-4-4","4","2020-05-19 14:14:29.002","2020-05-19 21:18:29.002","Ord#8F3","color
(0,0,0,255)","color(255,220,198,255)","1080","color(255,255,0,255)","0.0"
"evt-4-5","4","2020-05-20 00:00:29.002","2020-05-20 07:49:29.002","Ord#8F3","color
(0,0,0,255)","color(255,220,198,255)","1500","color(255,255,0,255)","0.0"
```

Equipment Schedule - Downtime Events Dataset



Dataset Editor

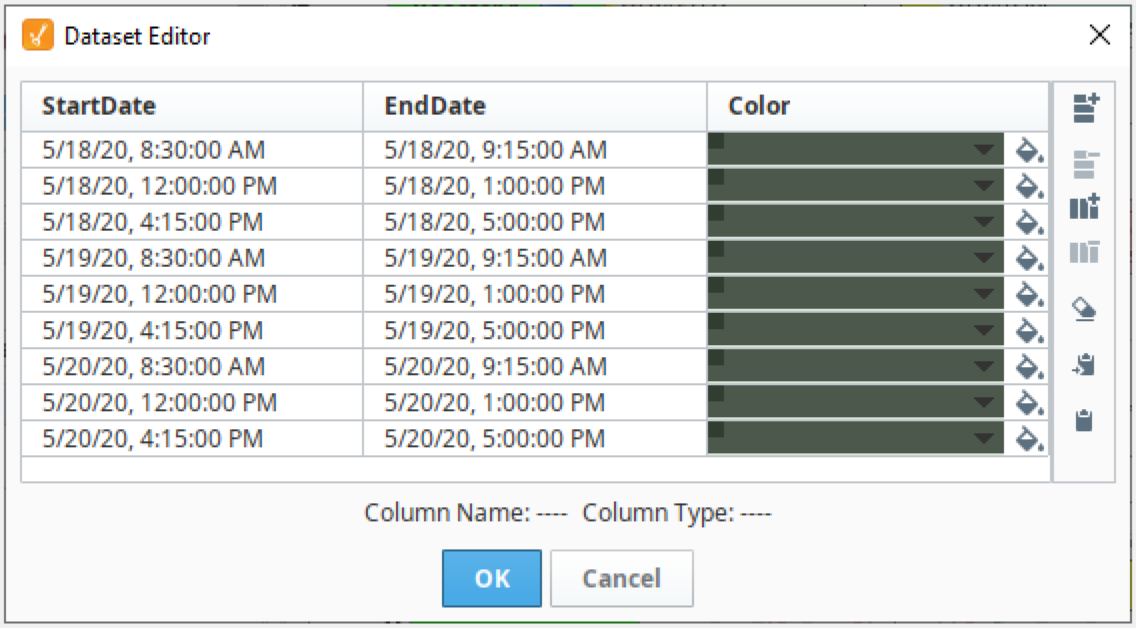
ItemID	StartDate	EndDate	Color	Layer
1	5/18/20, 1:25:29 PM	5/18/20, 1:37:29 PM	🟥	1
1	5/18/20, 2:11:29 PM	5/18/20, 2:49:29 PM	🟥	1
1	5/18/20, 8:34:29 PM	5/18/20, 9:12:29 PM	🟥	1
1	5/18/20, 9:48:29 PM	5/18/20, 10:09:29 PM	🟥	1
1	5/19/20, 3:42:29 AM	5/19/20, 4:07:29 AM	🟥	1
1	5/19/20, 4:55:29 AM	5/19/20, 5:13:29 AM	🟥	1
1	5/19/20, 6:09:29 AM	5/19/20, 6:46:29 AM	🟥	1
2	5/18/20, 4:00:29 AM	5/18/20, 4:31:29 AM	🟥	1
2	5/18/20, 5:02:29 AM	5/18/20, 5:39:29 AM	🟥	1
2	5/18/20, 10:08:29 PM	5/18/20, 10:45:29 PM	🟥	1
3	5/18/20, 2:56:29 AM	5/18/20, 3:34:29 AM	🟥	1
3	5/18/20, 4:21:29 AM	5/18/20, 4:56:29 AM	🟥	1
3	5/18/20, 5:26:29 AM	5/18/20, 5:40:29 AM	🟥	1
4	5/18/20, 3:11:29 AM	5/18/20, 3:26:29 AM	🟥	1
4	5/18/20, 4:14:29 AM	5/18/20, 4:50:29 AM	🟥	1
4	5/18/20, 5:35:29 AM	5/18/20, 6:01:29 AM	🟥	1
4	5/18/20, 1:39:29 PM	5/18/20, 1:50:29 PM	🟥	1
4	5/18/20, 2:29:29 PM	5/18/20, 2:57:29 PM	🟥	1

Column Name: ---- Column Type: ----

Equipment Schedule - Downtime Events Raw Data

```
"#NAMES"
"ItemID","StartDate","EndDate","Color","Layer"
"#TYPES"
"I","date","date","clr","I"
"#ROWS","18"
"1","2020-05-18 13:25:29.002","2020-05-18 13:37:29.002","color(212,49,49,75)","1"
"1","2020-05-18 14:11:29.002","2020-05-18 14:49:29.002","color(212,49,49,75)","1"
"1","2020-05-18 20:34:29.002","2020-05-18 21:12:29.002","color(212,49,49,75)","1"
"1","2020-05-18 21:48:29.002","2020-05-18 22:09:29.002","color(212,49,49,75)","1"
"1","2020-05-19 03:42:29.002","2020-05-19 04:07:29.002","color(212,49,49,75)","1"
"1","2020-05-19 04:55:29.002","2020-05-19 05:13:29.002","color(212,49,49,75)","1"
"1","2020-05-19 06:09:29.002","2020-05-19 06:46:29.002","color(212,49,49,75)","1"
"2","2020-05-18 04:00:29.002","2020-05-18 04:31:29.002","color(212,49,49,75)","1"
"2","2020-05-18 05:02:29.002","2020-05-18 05:39:29.002","color(212,49,49,75)","1"
"2","2020-05-18 22:08:29.002","2020-05-18 22:45:29.002","color(212,49,49,75)","1"
"3","2020-05-18 02:56:29.002","2020-05-18 03:34:29.002","color(212,49,49,75)","1"
"3","2020-05-18 04:21:29.002","2020-05-18 04:56:29.002","color(212,49,49,75)","1"
"3","2020-05-18 05:26:29.002","2020-05-18 05:40:29.002","color(212,49,49,75)","1"
"4","2020-05-18 03:11:29.002","2020-05-18 03:26:29.002","color(212,49,49,75)","1"
"4","2020-05-18 04:14:29.002","2020-05-18 04:50:29.002","color(212,49,49,75)","1"
"4","2020-05-18 05:35:29.002","2020-05-18 06:01:29.002","color(212,49,49,75)","1"
"4","2020-05-18 13:39:29.002","2020-05-18 13:50:29.002","color(212,49,49,75)","1"
"4","2020-05-18 14:29:29.002","2020-05-18 14:57:29.002","color(212,49,49,75)","1"
```

Equipment Schedule - Break Events



Dataset Editor

StartDate	EndDate	Color
5/18/20, 8:30:00 AM	5/18/20, 9:15:00 AM	■
5/18/20, 12:00:00 PM	5/18/20, 1:00:00 PM	■
5/18/20, 4:15:00 PM	5/18/20, 5:00:00 PM	■
5/19/20, 8:30:00 AM	5/19/20, 9:15:00 AM	■
5/19/20, 12:00:00 PM	5/19/20, 1:00:00 PM	■
5/19/20, 4:15:00 PM	5/19/20, 5:00:00 PM	■
5/20/20, 8:30:00 AM	5/20/20, 9:15:00 AM	■
5/20/20, 12:00:00 PM	5/20/20, 1:00:00 PM	■
5/20/20, 4:15:00 PM	5/20/20, 5:00:00 PM	■

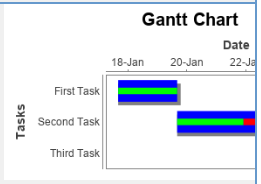
Column Name: ---- Column Type: ----

Equipment Schedule - Break Events Raw Data

```
"#NAMES"
"StartDate","EndDate","Color"
"#TYPES"
"date","date","clr"
"#ROWS","9"
"2020-05-18 08:30:00.002","2020-05-18 09:15:00.002","color(55,120,55,50)"
"2020-05-18 12:00:00.002","2020-05-18 13:00:00.002","color(55,120,55,50)"
"2020-05-18 16:15:00.002","2020-05-18 17:00:00.002","color(55,120,55,50)"
"2020-05-19 08:30:00.002","2020-05-19 09:15:00.002","color(55,120,55,50)"
"2020-05-19 12:00:00.002","2020-05-19 13:00:00.002","color(55,120,55,50)"
"2020-05-19 16:15:00.002","2020-05-19 17:00:00.002","color(55,120,55,50)"
"2020-05-20 08:30:00.002","2020-05-20 09:15:00.002","color(55,120,55,50)"
"2020-05-20 12:00:00.002","2020-05-20 13:00:00.002","color(55,120,55,50)"
"2020-05-20 16:15:00.002","2020-05-20 17:00:00.002","color(55,120,55,50)"
```

Vision - Gantt Chart

General



Component Palette Icon:

Description

A Gantt chart is used for task scheduling. It shows a list of named tasks, each of which have a start date, an end date, and a percentage complete. This allows an easy way to visualize tasks, workflows, and scheduling.

The Gantt chart is configured by populating its Data property. Each row of the dataset represents a task. There should be four columns: the task label, the start date, the end date, and the percentage (0-100) complete.

Properties

Name	Description	Property Type	Scripting	Category
Axis Font	The font for axis labels.	Font	.axisLabelFont	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Chart Title	An optional title that will appear at the top of the chart.	String	.title	Appearance
Complete Color	The color to draw the amount completed in. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.completeColor	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Data	The data driving the chart.	Dataset	.data	Data
Date Axis Title	A date label to display on the axis title.	String	.dateAxisTitle	Appearance
Incomplete Color	The color to draw the amount remaining to do in. See Color Selector.	Color	.incompleteColor	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Plot Background	The background color for the plot. See Color Selector.	Color	.plotBackground	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Task Axis Title	A task label to display on the Axis Title.	String	.taskAxisTitle	Appearance
Task Color	The main color to draw tasks. See Color Selector.	Color	.taskColor	Appearance
Tick Font	The font for tick labels.	Font	.axisTickLabelFont	Appearance
Title Font	The font for the optional chart title.	Font	.titleFont	Appearance
Tooltips?	Show tooltips on tasks?	boolean	.tooltips	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

The following feature is new in Ignition version **8.0.16**
Click here to check out the other new features

- Description

  Provides an opportunity to perform further chart configuration via scripting.

- Parameters

  Component self- A reference to the component that is invoking this function.

  JFreeChart chart- A JFreeChart object. Refer to the JFreeChart documentation for API details.

- Return

  Nothing

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

This event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

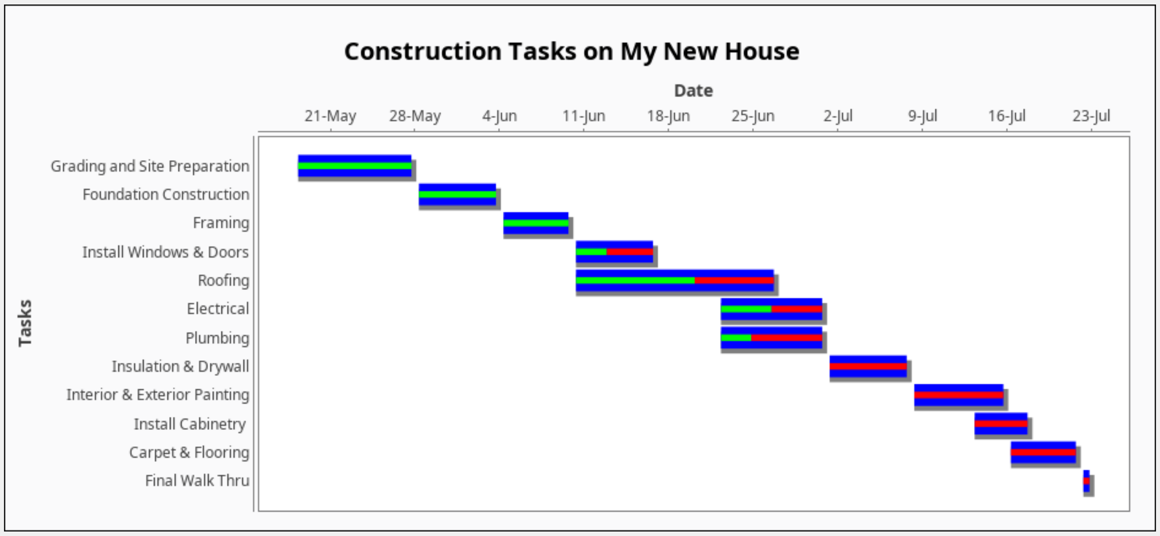
.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

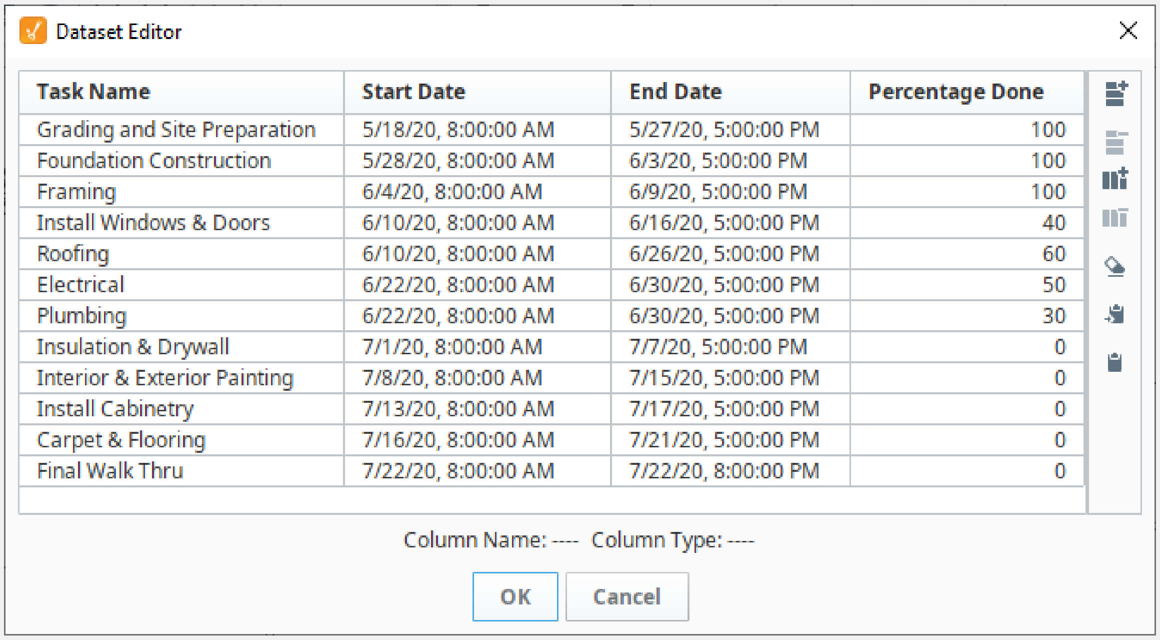
This component does not have any custom properties.

Examples

This example shows the tasks associated with a construction project on a new house. It is configured by populating the Data Property. Each row of the dataset includes the start date, end date and a percentage complete for each task. It is a good tool for task scheduling and a easy way to visualize tasks, workflow and scheduling.



Gantt Chart - Dataset Editor



Task Name	Start Date	End Date	Percentage Done
Grading and Site Preparation	5/18/20, 8:00:00 AM	5/27/20, 5:00:00 PM	100
Foundation Construction	5/28/20, 8:00:00 AM	6/3/20, 5:00:00 PM	100
Framing	6/4/20, 8:00:00 AM	6/9/20, 5:00:00 PM	100
Install Windows & Doors	6/10/20, 8:00:00 AM	6/16/20, 5:00:00 PM	40
Roofing	6/10/20, 8:00:00 AM	6/26/20, 5:00:00 PM	60
Electrical	6/22/20, 8:00:00 AM	6/30/20, 5:00:00 PM	50
Plumbing	6/22/20, 8:00:00 AM	6/30/20, 5:00:00 PM	30
Insulation & Drywall	7/1/20, 8:00:00 AM	7/7/20, 5:00:00 PM	0
Interior & Exterior Painting	7/8/20, 8:00:00 AM	7/15/20, 5:00:00 PM	0
Install Cabinetry	7/13/20, 8:00:00 AM	7/17/20, 5:00:00 PM	0
Carpet & Flooring	7/16/20, 8:00:00 AM	7/21/20, 5:00:00 PM	0
Final Walk Thru	7/22/20, 8:00:00 AM	7/22/20, 8:00:00 PM	0

Column Name: ---- Column Type: ----

Gantt Chart - Raw Data

```
"#NAMES"
"Task Name","Start Date","End Date","Percentage Done"
"#TYPES"
"str","date","date","I"
"#ROWS","12"
"Grading and Site Preparation","2020-05-18 08:00:00.000","2020-05-27 17:00:00.000","100"
"Foundation Construction","2020-05-28 08:00:00.000","2020-06-03 17:00:00.000","100"
"Framing","2020-06-04 08:00:00.000","2020-06-09 17:00:00.000","100"
"Install Windows & Doors","2020-06-10 08:00:00.000","2020-06-16 17:00:00.000","40"
"Roofing","2020-06-10 08:00:00.000","2020-06-26 17:00:00.000","60"
"Electrical","2020-06-22 08:00:00.000","2020-06-30 17:00:00.000","50"
"Plumbing","2020-06-22 08:00:00.000","2020-06-30 17:00:00.000","30"
"Insulation & Drywall","2020-07-01 08:00:00.000","2020-07-07 17:00:00.000","0"
"Interior & Exterior Painting","2020-07-08 08:00:00.000","2020-07-15 17:00:00.000","0"
"Install Cabinetry ","2020-07-13 08:00:00.000","2020-07-17 17:00:00.000","0"
"Carpet & Flooring","2020-07-16 08:00:00.000","2020-07-21 17:00:00.000","0"
"Final Walk Thru","2020-07-22 08:00:00.000","2020-07-22 20:00:00.000","0"
```

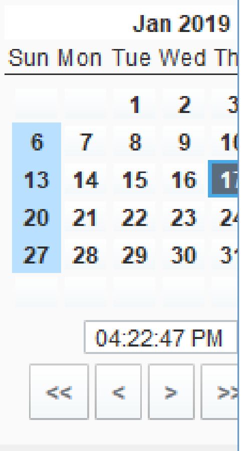
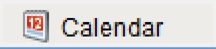
Vision - Calendar Palette

Calendar Components

The following components give you options for displaying and selecting dates and times.

In This Section ...

Vision - Calendar

General

Component Palette Icon:


Description
Displays a calendar and time input directly embedded in your window. Most commonly used by including one of the two date properties (immediate or latched) from the calendar in dynamic SQL Query Binding in Vision.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Date (immediate)	The date as it is selected right now.	Date	.date	Data
Date (latched)	The date the last time "OK" was pressed.	Date	.latchedDate	Data
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common

Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See Color Selector.	Color	.foreground	Appearance
Format String	The date formatting pattern used to format the string versions of the dates.	String	.format	Behavior
Formatted Date	The date property, as formatted by the format string.	String	.formattedDate	Data
Formatted Latched Date	The latched date property, as formatted by the format string.	String	.formattedLatchedDate	Data
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Opaque	If false, backgrounds are not drawn. If true, backgrounds are drawn.	boolean	.opaque	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Selected Border	The border for the selected day indicator.	Border	.selectedBorder	Appearance
Show OK Button	Turn this off if you don't want to show the OK button. The latched date and the immediate date will be equivalent.	boolean	.showOkButton	Behavior
Show Time	Turn this off if you don't want to show the time panel.	boolean	.showTime	Behavior
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Time Display Format	The format for displaying time in the panel.	int	.timeDisplayFormat	Behavior
Time Style	Select how this calendar should treat the time portion of the date.	int	.timeStyle	Behavior
Title Background	The background of the title bar. See Color Selector.	Color	.titleBackground	Appearance
Today Background	Background color for the today indicator. See Color Selector.	Color	.todayBackground	Appearance
Today Foreground	Foreground color for the today indicator. See Color Selector.	Color	.todayForeground	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Weekend Background	Background color for the weekend indicators. See Color Selector.	Color	.weekendBackground	Appearance
Weekend Foreground	Foreground color for the weekend indicators. See Color Selector.	Color	.weekendForeground	Appearance
Deprecated Properties				

Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

⚠ This event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
. property Name	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

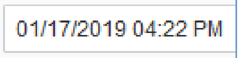
- Vision Component Customizers
- Style Customizer

Examples

There are no examples associated with this component.

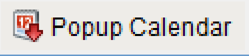
Vision - Popup Calendar

General



01/17/2019 04:22 PM

Component Palette Icon:



Description

The popup calendar is a popular way to provide date/time choosing controls on a window. Similar to the Calendar component, but takes up much less screen real estate. Most commonly used by including this component's Date property in dynamic SQL Query Binding.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Calendar Background	The background color for the popup calendar. See Color Selector.	Color	.calendarBackground	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Date	The date that this component represents.	Date	.date	Data
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See Color Selector.	Color	.foreground	Appearance
Format String	The date formatting pattern used to display this date.	String	.format	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common

Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Selected Border	The border for the selected day indicator.	Border	.selectedBorder	Appearance
Show Navigation	Turn this off if you don't want to show the year and month navigation buttons.	boolean	.showNavigation	Appearance
Show OK Button	Turn this off if you don't want to show the OK button. The latched date and the immediate date will be equivalent.	boolean	.showOkButton	Behavior
Show Time	Turn this off if you don't want to show the time panel.	boolean	.showTime	Behavior
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Text	The displayed text of the date (depends on the format string).	String	.text	Data
Time Display Format	The format for displaying time in the panel.	int	.timeDisplayFormat	Behavior
Time Style	Select how this calendar should treat the time portion of the date.	int	.timeStyle	Behavior
Title Background	The background of the title bar.	Color	.titleBackground	Appearance
Today Background	Background color for the today indicator. See Color Selector.	Color	.todayBackground	Appearance
Today Foreground	Foreground color for the today indicator. See Color Selector.	Color	.todayForeground	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Weekend Background	Background color for the weekend indicators. See Color Selector.	Color	.weekendBackground	Appearance
Weekend Foreground	Foreground color for the weekend indicators. See Color Selector.	Color	.weekendForeground	Appearance
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

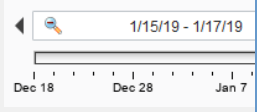
- Vision Component Customizers
- Style Customizer

Examples

There are no examples associated with this component.

Vision - Date Range

General



Component Palette Icon:

Date Range

Description

The date range component provides an intuitive, drag-and-drop way to select a contiguous range of time. The user is shown a timeline and can drag or stretch the selection box around on the timeline. The selected range is always a whole number of units, where the unit is determined by the current zoom level.

Note: The **Start/End** dates and **Outer Start/End** dates will be ignored when the window opens unless the Startup Mode property is set to "None."

Data Density Histogram

As an advanced optional feature, the date range can display a data density histogram inside the timeline. This is useful for historical data with gaps in it, so that the end user isn't hunting for data. (Tip: This is also great for demos, to make it easy to find historical data in a database that isn't being continuously updated).

To use this feature, bind the Data Density dataset to a query that returns just the timestamps of the target table. These timestamps will be used to fill in the histogram behind the timeline. You can use the Outer Range Start Date and Outer Range End Date properties in your query to limit the overall return size for the query.

⚠ Timestamps must be ordered by date (ascending) to display correctly.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Box Fill	The fill color for the selection box.	Color	.boxFill	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common

Data Density	A dataset that is used to calculate a histogram of data density.	Dataset	.densityData	Data
Date Style	The style to display dates in. For international support.	int	.dateStyle	Appearance
Editor Background	The background color of the textual date range editor portion of this component.	Color	.editorBackground	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
End Date	The ending date of the currently selected range.	Date	.endDate	Data
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance
High Density Color	The color used to indicate high data density. See Color Selector.	Color	.highDensityColor	Appearance
Max Selection	The maximum size of the selected date range.	String	.maxSelectionSize	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Opaque	If false, backgrounds are not drawn. If true, backgrounds are drawn.	boolean	.opaque	Common
Outer Range End	The ending date of the available outer range.	Date	.outerRangeEndDate	Data
Outer Range Start	The starting date of the available outer range.	Date	.outerRangeStartDate	Data
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Selection Highlight	The focus highlight color for the selection box. See Color Selector.	Color	.selectionHighlight	Appearance
Start Date	The starting date of the currently selected range.	Date	.startDate	Data
Startup Mode	Controls whether or not this date range automatically assigns itself a starting range based on the current time	int	.startupMode	Behavior
Startup Range	If startup mode is Automatic, this will be the starting range of time available for selection.	String	.startupRange	Behavior
Startup Selection	If startup mode is Automatic, this will be the starting selected range.	String	.startupSelection	Behavior

Styles	Contains the component's styles.	Dataset	.styles	Appearance
Tick Density	This is multiplied by the width to determine the current ideal tick unit.	float	.tickDensity	Behavior
Time Style	The style to display times of day. For international support.	int	.timeStyle	Appearance
Today Color	The color of the "Today Arrow" indicator. See Color Selector.	Color	.todayIndicatorColor	Appearance
Track Margin	The amount of room on either side of the slider track. May need adjusting of default font is changed.	int	.trackMargin	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

- Since 7.8.1
- Description

  Sets the selected range. The outer range will move if needed. Note: The start and end times are determined based on the zoom level and may not move (or may move farther than intended) if the component is zoomed out too far for the amount of change attempted. For example, when days are showing, moving the start time 5 minutes forward will not effect the start, and moving the end time 5 minutes forward will add one day.

- Parameters

  Date start - The starting date for the new selection.

  Date end - The ending date for the new selection.

- Return

  Nothing

Code Snippet

```
# This example moves the existing Start Date and End Date
# of a Date Range component ahead 8 hours
from java.util import Calendar

# Get the current start and end
dateRangeComponent = event.source.parent.getComponent('Date Range')
startDate = dateRangeComponent.startDate
endDate = dateRangeComponent.endDate

# Calculate the new start and end dates
cal = Calendar.getInstance();
cal.setTime(startDate);
cal.add(Calendar.HOUR, -8);
newStart = cal.getTime();

cal.setTime(endDate);
cal.add(Calendar.HOUR, -8);
newEnd = cal.getTime();

# Set the new range for the component. The outer range will
# automatically expand if needed.
dateRangeComponent.setRange(newStart, newEnd)
```

- Since 7.8.1
- Description

  Sets the outer range. The selected range will move if needed. Note: The start and end times are determined based on the zoom level and may not move (or may move farther than intended) if the component is zoomed out too far for the amount of change attempted. For example, when days are showing, moving the start time 5 minutes forward will not effect the start, and moving the end time 5 minutes forward will add one day.
- Parameters

  Date start - The starting date for the new outer range.

  Date end - The ending date for the new outer range.
- Return

  Nothing

Code Snippet

```
# This example moves the existing Outer Date Range
# of a Date Range component back two days
from java.util import Calendar

# Get the current start and end of the outer range
dateRangeComponent = event.source.parent.getComponent('Date Range')
startDate = dateRangeComponent.outerRangeStartDate
endDate = dateRangeComponent.outerRangeEndDate

# Calculate the new start and end dates for the outer range
cal = Calendar.getInstance();
cal.setTime(startDate);
cal.add(Calendar.DAY_OF_MONTH, 2);
newStart = cal.getTime();

cal.setTime(endDate);
cal.add(Calendar.DAY_OF_MONTH, 2);
newEnd = cal.getTime();

# Set the new outer range for the component.
dateRangeComponent.setOuterRange(newStart, newEnd)
```

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

⚠ This event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers
- Style Customizer

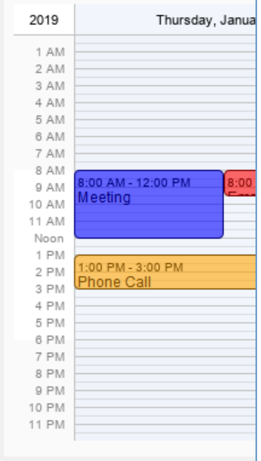
Examples

Code Snippet

```
//A Query binding on another component on the same window might look like this:

SELECT Column1, Column2, Column3
FROM MyTable WHERE
  t_stamp >= "{Root Container.Date Range.startDate}" AND
  t_stamp <= "{Root Container.Date Range.endDate}"
```

Vision - Day View

General



Component Palette Icon:

Day View

Description

This component displays a timeline for a single day, similar to what you might find in a personal planner/organizer. By filling in the Calendar Events dataset property, the component will display events that occur on this day. Each event can have custom text and a custom display color associated with it.

Properties

Name	Description	Property Type	Scripting	Category
24 Hour Format	Whether or not to show 24 hour or 12 hour format.	boolean	.twentyFourHour	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Calendar Background Color	The color of the calendar's background. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.calendarBackground	Appearance
Calendar events	Contains the calendar events.	Dataset	.events	Data
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Day	Set the calendar's day.	int	.day	Data

Day Outline Color	The color of the day's outline. See Color Selector.	Color	.boxOutline	Appearance
Event Font	The font for all calendar events.	Font	.eventFont	Appearance
Grid marks	Set the amount of grid lines.	int	.gridMarks	Appearance
Hour Font	The font for the hour of the day.	Font	.hourFont	Appearance
Hour Foreground Color	The foreground color for hours in a day. See Color Selector.	Color	.hourForeground	Appearance
Hover Background Color	The background color of the hovered time. See Color Selector.	Color	.hoverBackground	Appearance
Hovered Event	The calendar's hovered event.	int	.hoveredEvent	Data
Hovered Time	The calendar's hovered time.	String	.hoveredTime	Data
Month	Set the calendar's month.	int	.month	Data
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Non-Working Hours Background Color	The background color for the non-working hours of the day. See Color Selector.	Color	.nonWorkingHourBackground	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Selected Event	The calendar's selected event.	int	.selectedEvent	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Today's Background Color	The color of the today's background. See Color Selector.	Color	.todayBackground	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Week Day Background Color	The color of the week day's background. See Color Selector.	Color	.weekDaysBackground	Appearance
Week Day Font	The font of the week day's text.	Font	.weekdayFont	Appearance
Week Day Foreground Color	The color of the week day's text. See Color Selector.	Color	.weekDaysForeground	Appearance
Working End Hour	The end hour of a working day.	int	.workingEndHour	Appearance

Working Start Hour	The start hour of a working day.	int	.workingStartHour	Appearance
Year	Set the calendar's year.	int	.year	Data
Zoom	Zooms into the specified zoom time-range.	boolean	.autoZoom	Appearance
Zoomed End Hour	The end hour zoomed in.	int	.autoZoomEndHour	Appearance
Zoomed Start Hour	The start hour zoomed in.	int	.autoZoomStartHour	Appearance
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

This event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

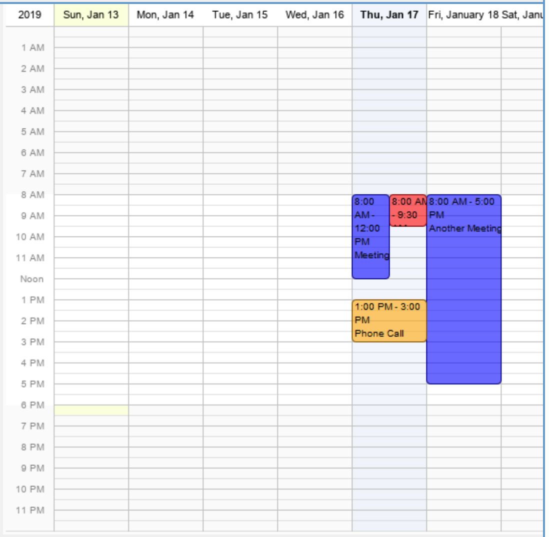
- Vision Component Customizers
- Style Customizer

Examples

There are no examples associated with this component.

Vision - Week View

General



Component Palette Icon:

Week View

Description

Displays a full week's worth of events on a calendar. Configuration is achieved by populating the Calendar Events dataset. See the Vision - Day View for details.

Properties

Name	Description	Property Type	Scripting	Category
24 Hour Format	Whether or not to show 24 hour or 12 hour format.	boolean	.twentyFourHour	Appearance
Border	The border surrounding this component. NOptions are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Calendar Background Color	The color of the calendar's background. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.calendarBackground	Appearance
Calendar events	Contains the calendar events.	Dataset	.events	Data
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Day	Set the calendar's day.	int	.day	Data

Day Outline Color	The color of the day's outline. See Color Selector.	Color	.boxOutline	Appearance
Event Font	The font for all calendar events.	Font	.eventFont	Appearance
Grid marks	Set the amount of grid lines.	int	.gridMarks	Appearance
Hour Font	The font for the hour of the day.	Font	.hourFont	Appearance
Hour Foreground Color	The foreground color for hours in a day. See Color Selector.	Color	.hourForeground	Appearance
Hover Background Color	The background color of the hovered day and time. See Color Selector.	Color	.hoverBackground	Appearance
Hovered Day	The calendar's hovered day.	String	.hoveredDay	Data
Hovered Event	The calendar's hovered event.	int	.hoveredEvent	Data
Hovered Time	The calendar's hovered time.	String	.hoveredTime	Data
Month	Set the calendar's month.	int	.month	Data
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Non-Working Hours Background Color	The background color for the non-working hours of the day. See Color Selector.	Color	.nonWorkingHourBackground	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Selected Background Color	The color of the selected day's background. See Color Selector.	Color	.selectedBackground	Appearance
Selected Day	The calendar's selected day.	String	.selectedDay	Data
Selected Event	The calendar's selected event.	int	.selectedEvent	Data
Show Event Time?	Whether or not to show the event time.	boolean	.showEventTime	Appearance
Show Weekend?	Whether or not to show Saturday and Sunday.	boolean	.showWeekend	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Today's Background Color	The color of the today's background. See Color Selector.	Color	.todayBackground	Appearance

Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Week Day Background Color	The color of the week day's background. See Color Selector.	Color	.weekDaysBackground	Appearance
Week Day Font	The font of the week day's text.	Font	.weekdayFont	Appearance
Week Day Foreground Color	The color of the week day's text. See Color Selector.	Color	.weekDaysForeground	Appearance
Working End Hour	The end hour of a working day.	int	.workingEndHour	Appearance
Working Start Hour	The start hour of a working day.	int	.workingStartHour	Appearance
Year	Set the calendar's year.	int	.year	Data
Zoom	Zooms into the specified zoom time range.	boolean	.autoZoom	Appearance
Zoomed End Hour	The end hour zoomed in.	int	.autoZoomEndHour	Appearance
Zoomed Start Hour	The start hour zoomed in.	int	.autoZoomStartHour	Appearance
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

⚠ This event fires *after* the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

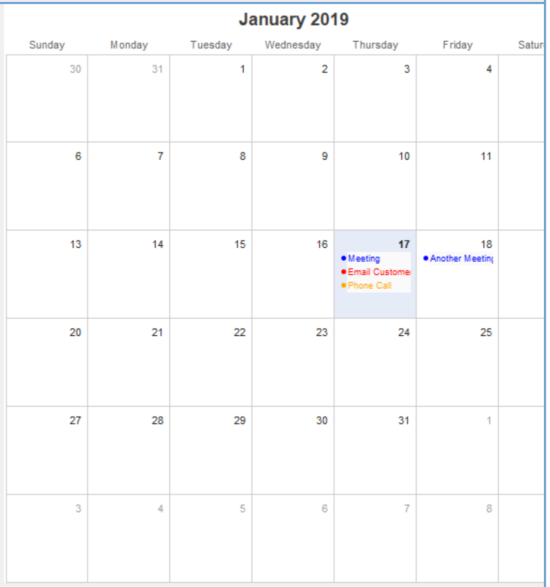
- Vision Component Customizers
- Style Customizer

Examples

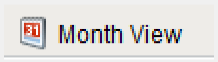
There are no examples associated with this component.

Vision - Month View

General



Component Palette Icon:



Description

This component displays events for an entire month. By filling in the Calendar Events dataset property, the component will display events that occur for each day of the month. Each event can have custom text and a custom display color associated with it.

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Calendar Background Color	The color of the calendar's background. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.calendarBackground	Appearance
Calendar events	Contains the calendar events.	Dataset	.events	Data
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Day Font	The font for the number representing the day of the month.	Font	.dayFont	Appearance

Day Foreground Color	The foreground color for days in this month. See Color Selector.	Color	.dayOfMonthForeground	Appearance
Day Other Foreground Color	The foreground color for days not in this month. See Color Selector.	Color	.dayOfMonthOtherForeground	Appearance
Day Outline Color	The color of the day's outline. See Color Selector.	Color	.boxOutline	Appearance
Event Background Color	The background color of the selected event. See Color Selector.	Color	.itemSelBackground	Appearance
Event Display Mode	Affects how events are displayed. \\ \\Standard Mode: Displays each event \\Highlight Mode: Highlights each day that contains events using the event highlight background color.	int	.displayMode	Appearance
Event Font	The font for all calendar events.	Font	.eventFont	Appearance
Event Highlight Background	The background color of a day with events. Used only in highlight mode.	Color	.highlightBackground	Appearance
Header Background Color	The color of the header's background. See Color Selector.	Color	.monthHeaderBackground	Appearance
Header Font	The font of the header's text.	Font	.headerFont	Appearance
Header Foreground Color	The color of the header's text. See Color Selector.	Color	.monthHeaderForeground	Appearance
Hover Background Color	The background color of the hovered day. See Color Selector.	Color	.hoverBackground	Appearance
Hovered Day	The calendar's hovered day.	String	.hoveredDay	Data
Month	Set the calendar's month.	int	.month	Data
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Selected Background Color	The color of the selected day's background. See Color Selector.	Color	.selectedBackground	Appearance
Selected Day	The calendar's selected day.	String	.selectedDay	Data
Selected Event	The calendar's selected event.	int	.selectedEvent	Data

Styles	Contains the component's styles.	Dataset	.styles	Appearance
Today's Background Color	The color of the today's background. See Color Selector.	Color	.todayBackground	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Week Day Background Color	The color of the week day's background. See Color Selector.	Color	.weekDaysBackground	Appearance
Week Day Font	The font of the week day's text.	Font	.weekdayFont	Appearance
Week Day Foreground Color	The color of the week day's text. See Color Selector.	Color	.weekDaysForeground	Appearance
Year	Set the calendar's year.	int	.year	Data
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

⚠ This event fires *after* the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers
- Style Customizer

Examples

There are no examples associated with this component.

Vision - Admin Palette

Admin Components

The following components give you administrative access to various gateway systems.

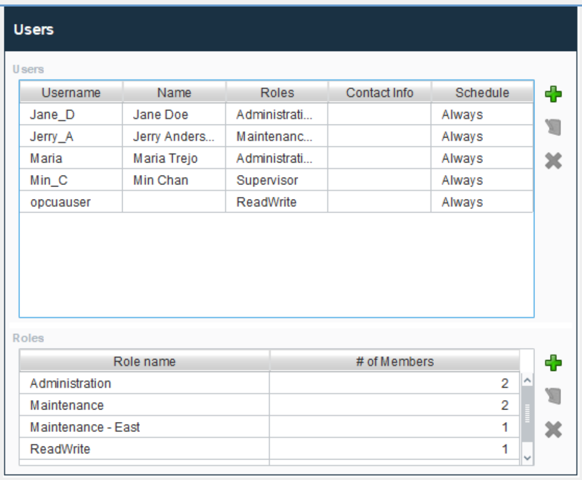
In This Section ...

Vision - User Management

INDUCTIVE UNIVERSITY

User, Schedule and Roster Management

Watch the Video

General

Component Palette Icon:
User Management

Users

Users

Username	Name	Roles	Contact Info	Schedule
Jane_D	Jane Doe	Administrati...		Always
Jerry_A	Jerry Anders...	Maintenanc...		Always
Maria	Maria Trejo	Administrati...		Always
Min_C	Min Chan	Supervisor		Always
opcuauser		ReadWrite		Always

Roles

Role name	# of Members
Administration	2
Maintenance	2
Maintenance - East	1
ReadWrite	1

Description

The user management panel provides a built-in way to edit User Source users and roles from a Vision Client.

To make changes to the Gateway's system user source from the Designer or Client, **Allow User Admin** must be checked in Gateway Settings in the Gateway Config page.

This component can be run in one of three modes:

Manage Users Mode: In this mode, the component manages all of the users contained in the user source. Users and roles may be added, removed, and edited.

Edit Single Mode: In this mode, the component only edits a single user. Which user is being edited is controlled via the "User Source" and "Username" properties.

Edit Current Mode: In this mode, the user who is currently logged into the project can edit themselves. Obviously, the ability to assign roles is not available in this mode. This can be useful to allow users to alter their own password, adjust their contact information, and update their schedules.

Warning: Be careful to only expose this component to users who should have the privileges to alter other users. Access to this component in "Manage Users" mode will allow users to edit other users' passwords and roles.

Properties

Name	Description	Property Type	Scripting	Category

Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Contact Info Editing Enabled	If true, a user's contact info will be editable.	boolean	.allowContactInfoEditing	Behavior
Editing Schedule Available Color	Changes the color of the available times in the schedule. See Color Selector.	Color	.schedulePreviewAvailableColor	Appearance
Editing Schedule Available Text Color	Changes the text color of events on the schedule preview. See Color Selector.	Color	.eventForeground	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of the text on this component.	Font	.font	Appearance
Mode	Affects what mode the user management component runs in. **Value** | **Description** | **intValue** Manage Users | Allows edits to all Users and Roles in a single source determined by the User Source property. Default | 0 Edit Current | Allows edits to the currently logged in user details. | 1 Edit Single | Allows edits to a specific user determined by the User Source and Username properties. | 2	int	.mode	Behavior
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Role Assigning Enabled	If true, a user's roles will be editable.	boolean	.allowRoleAssigning	Behavior
Role Management Enabled	If true, role management is available.	boolean	.allowRoleManagement	Behavior
Row Height	Alter the size of the rows in the component's tables.	int	.rowHeight	Appearance
Schedule Adjustments Enabled	If true, a user's schedule adjustments will be editable.	boolean	.allowScheduleModifications	Behavior
Show Contact Info Column	Controls whether the user table shows the contact info column or not.	boolean	.columnContactInfo	Appearance

Show Name Column	Controls whether the user table shows the name column or not.	boolean	.columnName	Appearance
Show Roles Column	Controls whether the user table shows the roles column or not.	boolean	.columnRoles	Appearance
Show Schedule Column	Controls whether the user table shows the schedule column or not.	boolean	.columnSchedule	Appearance
Show Username Column	Controls whether the user table shows the username column or not.	boolean	.columnUsername	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Table Color	Changes the background color of the tables, User Roles and Role Member lists. Note: When a row is selected it will revert to highlighted.	Color	.tableBackground	Appearance
Table Header Color	Changes the background color of the table headers. See Color Selector.	Color	.tableHeaderBackground	Appearance
Table Header Text Color	Changes the text color of the table headers. See Color Selector.	Color	.tableHeaderTextColor	Appearance
Table Text Color	Changes the text color of the tables. Note: When a row is selected, it will revert to black. See Color Selector.	Color	.tableForeground	Appearance
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
User Source	The user source to manage users in. If blank, uses the project's default user source.	String	.userProfile	Behavior
Username	The name of the user being edited. Read-only except when mode is **Edit Single**, in which case it defines the user to be edited.	String	.username	Behavior
Username Editing Enabled	If true, usernames will be editable.	boolean	.allowUsernameEditing	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Window Color	Changes the window background color. See Color Selector.	Color	.windowBackground	Appearance
Window Header Color	Changes the window header background color. See Color Selector.	Color	.windowHeaderBackground	Appearance
Window Header Save Button Background Color	Changes the window header save button background color. See Color Selector.	Color	.windowHeaderSaveButtonBackground	Appearance
Window Header Save Button Text Color	Changes the window header save button text color. See Color Selector.	Color	.windowHeaderSaveButtonForeground	Appearance

Window Header Text Color	Changes the window header text color. See Color Selector.	Color	.windowHeaderForeground	Appearance
Window Text Color	Changes the text color of the window. See Color Selector.	Color	.windowForeground	Appearance
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

- Description

  Called for each user loaded into the management table. Return false to hide this user from the management table. This code is executed in a background thread.

- Parameters

  Component self - A reference to the component that is invoking this function.

  User Object user - The user object itself. Call user.get('propertyName') to inpsect. Common properties: 'username',' schedule', 'language', user.getRoles() for a list of rolenames.

- Return

  Boolean

- Description

  Called for each role loaded into the management table. Return false to hide this role from the management table. This code is executed in a background thread.

- Parameters

  Component self - A reference to the component that is invoking this function.

  String role - The role name.

- Return

  Boolean

- Description

  Called for each schedule loaded into the schedule dropdown in the edit user panel. Return false to hide this schedule from the dropdown. This code is executed in a background thread.

- Parameters

  Component self - A reference to the component that is invoking this function.

  String schedule - The schedule name.

- Return

  Boolean

- Description

  Called when the add button is pressed in the users table

- Parameters

  Component self - A reference to the component that is invoking this function.

  Object saveContext - An object that can be used to reject the add by calling saveContext.rejectSave ('reason')

- Return

  Nothing

- Description

  Called when the delete button is pressed in the users table. This code is executed in the background thread and is called once for each user selected.

- Parameters

  Component self - A reference to the component that is invoking this function.

  Object saveContext - An object that can be used to reject the edit by calling saveContext.rejectSave ('reason'). If more than one user is rejected, reasons will be concatenated.

  Object user - The user that is trying to be deleted. Call user.get('propertyName') to inspect. Common properties: 'username', 'schedule', 'language'. Call user.getRoles() for a list of rolenames.

- Return

  Nothing

- Description

  Called when the save button is pressed when adding or editing a user. This code is executed in a background thread.

- Parameters

  Component self - A reference to the component that is invoking this function.

  Object saveContext - An object that can be used to reject the edit by calling saveContext.rejectSave ('reason').

  User Object user - The user that is trying to be saved. Call user.get('propertyName') to inspect. Common properties: 'username', 'schedule','language'. Call user.getRoles() for a list of rolenames.

- Return

  Nothing

- Description

  Called when the add button is pressed in the roles table.

- Parameters

  Component self - A reference to the component that is invoking this function.

  Object saveContext - An object that can be used to reject the add by calling saveContext.rejectSave ('reason')

- Return

  Nothing

- Description

  Called when the save button is pressed when adding or editing a role. This code is executed in a background thread.

- Parameters

  Component self - A reference to the component that is invoking this function.

  Object saveContext - An object that can be used to reject the edit by calling saveContext.rejectSave ('reason'). If more than one role is rejected, reasons will be concatenated.

  String name - The role name that is being deleted.

- Return

  Nothing

- Description

  Called when the save button is pressed when adding or editing a role. This code is executed in a background thread.

- Parameters

  Component self - A reference to the component that is invoking this function.

  Object saveContext - An object that can be used to reject the edit by calling saveContect.rejectSave ('reason').

  String oldName - The role name before editing. Will be None for a role being added.

  String newName - The new name of the edited role.

- Return

  Nothing

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

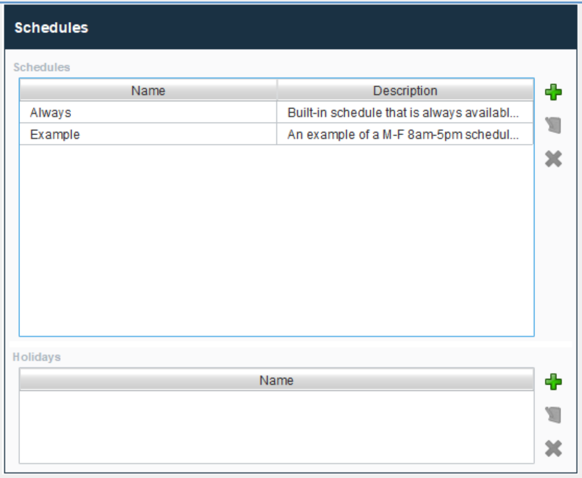
Customizers

- Vision Component Customizers

Examples

There are no examples associated with this component.

Vision - Schedule Management

General
 Schedules Schedules | Name | Description | |---|---| | Always | Built-in schedule that is always availabl... | | Example | An example of a M-F 8am-5pm schedul... | Holidays | Name | |---| **Component Palette Icon:** Schedule Management

User, Schedule and Roster Management

Watch the Video

Description
This component allows for management of schedules. Schedules can be defined by specifying which days of the week and which times of day they are active on. The times of day are defined using a string of time ranges, where the times are specified in 24-hr format with dashes between the beginning and the end. Multiple ranges can be specified by separating them with commas. Examples:

8:00-17:00	Valid from 8am to 5pm
6:00-12:00, 12:45-14:00	Valid from 6am to noon, and then again from 12:45pm to 2pm
0:00-24:00	Always valid.

See Color Selector.Schedules that alternate weekly or daily can be specified by using the repetition settings. All repeating schedules need a starting day. For example, you could have a schedule that repeats on a weekly basis, with 1-week on and 1-week off. This schedule would be active for seven days starting on the starting day, and then inactive for the next seven days, then active for seven days, and so on. Note that the days of the week and time settings are evaluated in addition to the repetition settings. This means that both settings must be true for the schedule to be active. Also note that if you set "Repeat / Alternate" to a setting other than "Off" and you do not specify a starting day, the schedule will never be active.

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. The border is unaffected by rotation.	Border	.border	Common
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common

Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Schedule Available Color	Changes the color of the available times in the schedule. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSLvalue. See Color Selector.	Color	.schedulePreviewAvailableColor	Appearance
Schedule Available Text Color	Changes the text color of events on the schedule preview. See Color Selector.	Color	.eventForeground	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Table Color	Changes the background color of the tables, User Roles and Role Member lists. See Color Selector. Note: When a row is selected it will revert to highlighted.	Color	.tableBackground	Appearance
Table Header Color	Changes the background color of the table headers. See Color Selector.	Color	.tableHeaderBackground	Appearance
Table Header Text Color	Changes the text color of the table headers. See Color Selector.	Color	.tableHeaderTextColor	Appearance
Table Text Color	Changes the text color of the tables. Note: When a row is selected, it will revert to black. See Color Selector.	Color	.tableForeground	Appearance
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Window Color	Changes the window background color. See Color Selector.	Color	.windowBackground	Appearance
Window Header Color	Changes the window header background color. See Color Selector.	Color	.windowHeaderBackground	Appearance
Window Header Save Button Background Color	Changes the window header save button background color. See Color Selector.	Color	.windowHeaderSaveButtonBackground	Appearance
Window Header Save Button Text Color	Changes the window header save button text color. See Color Selector.	Color	.windowHeaderSaveButtonForeground	Appearance
Window Header Text Color	Changes the window header text color. See Color Selector.	Color	.windowHeaderForeground	Appearance
Window Text Color	Changes the text color of the window. See Color Selector.	Color	.windowForeground	Appearance
Deprecated Properties				

Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

- Description

  Called for each schedule loaded into the management table. Return false to hide this schedule from the management table. This code is executed in a background thread.

- Parameters

  Component self - A reference to the component that is invoking this function.

  String schedule - The schedule name

- Return

  Boolean

- Description

  Called for each holiday loaded into the management table. Return false to hide this holiday from the management table. This code is executed in a background thread.

- Parameters

  Component self - A reference to the component that is invoking this function.

  String holiday - The holiday name.

- Return

  Boolean

- Description

  Called when the add button is pressed when adding a schedule. This code is executed in a background thread.

- Parameters

  Component self - A reference to the component that is invoking this function.

  Object saveContext - An object that can be used to reject the add by calling saveContect.rejectSave ('reason').

- Return

  Nothing

- Description

  Called when the delete button is pressed for one or more schedules. This code is executed in a background thread, once for each schedule to be deleted.

- Parameters

  Component self - A reference to the component that is invoking this function.

  Object saveContext - An object that can be used to reject the deletion by calling saveContect.rejectSave('reason').

  String name - The name of the schedule to be deleted.

- Return

  Nothing

- Description

  Called when the save button is pressed when adding or editing a schedule. This code is executed in a background thread.

- Parameters

  Component self - A reference to the component that is invoking this function.

  Object saveContext - An object that can be used to reject the edit by calling saveContect.rejectSave('reason').

  String oldName - The schedule name before editing. Will be None for a schedule being added.

  String newName - The new name of the edited schedule.

- Return

  Nothing

- Description

  Called when the add button is pressed when to add a holiday. This code is executed in a background thread.

- Parameters

  Component self - A reference to the component that is invoking this function.

  Object saveContext - An object that can be used to reject the add by calling saveContect.rejectSave('reason').

- Return

  Nothing

- Description

  Called when the delete button is pressed for one or more holidays. This code is executed in a background thread, once for each holiday to be deleted.

- Parameters

  Component self - A reference to the component that is invoking this function.

  Object saveContext - An object that can be used to reject the edit by calling saveContect.rejectSave('reason').

  String name - The name of the holiday to be deleted.

- Return

  Nothing

- Description

  Called when the save button is pressed when adding or editing a holiday. This code is executed in a background thread.

- Parameters

  Component self - A reference to the component that is invoking this function.

  Object saveContext - An object that can be used to reject the edit be calling saveContext.rejectSave('reason')

  String oldName - The holiday name before editing. Will be None for a holiday being added.

  String newName - The new name of the edited holiday.

- Return

  Nothing

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

⚠ This event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

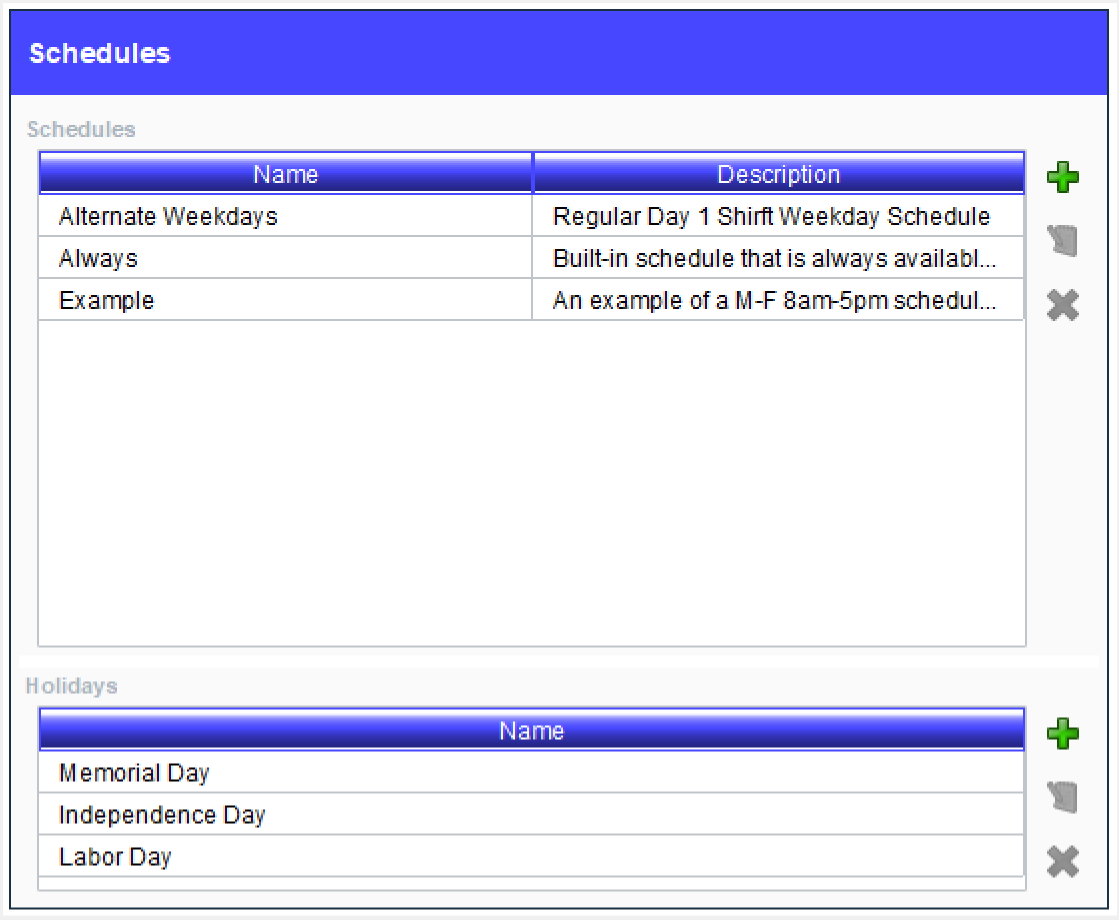
Customizers

- Vision Component Customizers

Examples

Here is an example of the schedule management component and its property table.

Property Name	Value
Name	Schedules
Enabled	True
Visible	True
Touchscreen Mode	Single-Click
Table Header Color	71,71,255
Table Header Text Color	255,255,255
Window Header Color	71,71,255



Schedules

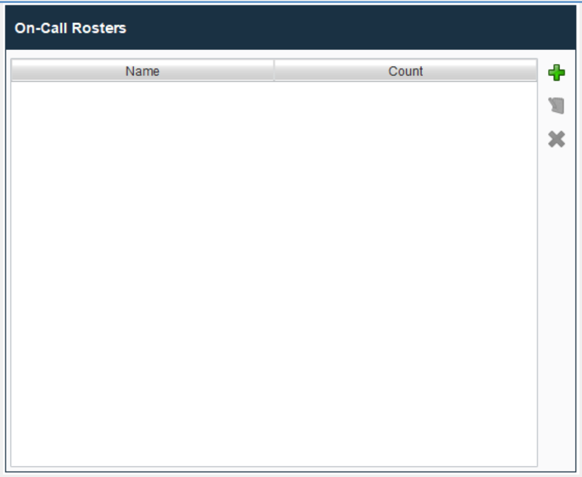
Schedules

Name	Description
Alternate Weekdays	Regular Day 1 Shirft Weekday Schedule
Always	Built-in schedule that is always availabl...
Example	An example of a M-F 8am-5pm schedul...

Holidays

Name
Memorial Day
Independence Day
Labor Day

Vision - Roster Management

General
 **Component Palette Icon:** Roster Management

User, Schedule and Roster Management

Watch the Video

Description
The user management panel provides a built-in way to edit rosters from a client.

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
User Source	The user source to manage users in. If blank, uses the project's default user source.	String	.addFromUserSource	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

- Description

  Called for each roster loaded into the management table. Return false to hide this roster from the management table. This code is executed in a background thread.

- Parameters

  Component self- A reference to the component that is invoking this function.

  String roster - The name of the roster.

- Return

  Boolean

- Description

  Called for each user in a user source to be shown as an available user for the roster currently being edited. Return false to hide this user so that it cannot be added to the roster. This code is executed in a background thread.

- Parameters

  Component self- A reference to the component that is invoking this function.

  String roster - The name of the roster being edited.

  String userSource - The name of the user source being used to populate the list of available users.

  User Object user - The user object itself. Call user.get('propertyName') to inspect. Common properties: 'username','schedule','language'. Call user.getRoles() for a list of rolenames.

- Return

  Boolean

- Description

  Called when the save button is pressed when editing a roster. This code is executed in a background thread.

- Parameters

  Component self - A reference to the component that is invoking this function.

  Object saveContext - An object that can be used to reject the edit by calling saveContext.rejectSave('reason')

  String rosterName - The name of the roster being edited.

- Return

  Nothing

- Description

  Called when the add button is pressed. This code is executed in a background thread.

- Parameters

  Component self - A reference to the component that is invoking this function.

  Object createContext - An object that can be used to reject the edit by calling createContext.rejectCreate('reason')

  String rosterName - The name of the roster being created.

- Return

  Nothing

- Description

  Called when the delete button is pressed. This code is executed in a background thread.

- Parameters

  Component self - A reference to the component that is invoking this function.

  Object deleteContext - An object that can be used to reject the edit by calling deleteContext.rejectDelete('reason')

  String rosterNames - A list of the roster names being deleted.

- Return

  Nothing

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

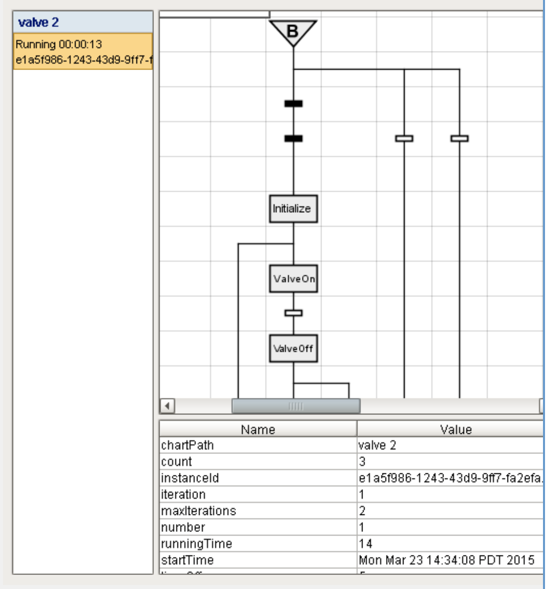
- Vision Component Customizers

Examples

There are no examples associated with this component.

Vision - SFC Monitor

General



valve 2

Running 00:00:13
e1a5f986-1243-43d9-9ff7-f

Name	Value
chartPath	valve 2
count	3
instanceId	e1a5f986-1243-43d9-9ff7-fa2efa.
iteration	1
maxIterations	2
number	1
runningTime	14
startTime	Mon Mar 23 14:34:08 PDT 2015

Component Palette Icon:

SFC Monitor

Description

A component to monitor Sequential Function Chart performance. In addition the component allows for the operator to control the chart instance through the charts instance 'id' property. The chart scoped variables are available through the scope dataset property.

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Instance ID	The UUID of the sequential function chart to monitor.	String	.instanceId	Data
Instance List Visible	Shows or hides the list of SFC instances on the left.	boolean	.instanceListVisible	Appearance
Legend Visible	Shows or hides the step and transition state legend.	boolean	.legendVisible	Appearance
Name	The name of this component.	String	.name	Common
Scope Dataset	Dataset containing the variables in chart scope.	Dataset	.scopeDataset	Data
Scope Table Visible	Shows or hides the chart scope inspection table.	boolean	.scopeTableVisible	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Zoom	The zoom multiplier to display the chart's status at.	float	.zoom	Appearance

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

This event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
. property Name	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers

Examples

There are no examples associated with this component.

Vision - Alarming Palette

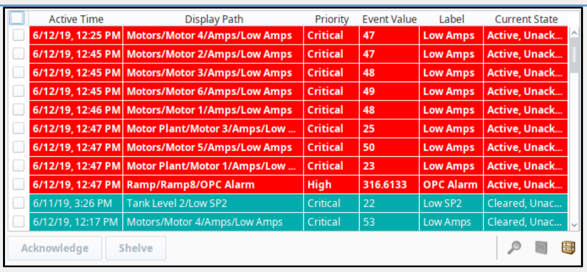
Alarming Components

The following components give you options for displaying Alarm information.

In This Section ...

Vision - Alarm Status Table

General



Active Time	Display Path	Priority	Event Value	Label	Current State
6/12/19, 12:25 PM	Motors/Motor 4/Amps/Low Amps	Critical	47	Low Amps	Active, Unack...
6/12/19, 12:45 PM	Motors/Motor 2/Amps/Low Amps	Critical	47	Low Amps	Active, Unack...
6/12/19, 12:45 PM	Motors/Motor 3/Amps/Low Amps	Critical	48	Low Amps	Active, Unack...
6/12/19, 12:45 PM	Motors/Motor 6/Amps/Low Amps	Critical	49	Low Amps	Active, Unack...
6/12/19, 12:46 PM	Motors/Motor 1/Amps/Low Amps	Critical	48	Low Amps	Active, Unack...
6/12/19, 12:47 PM	Motor Plant/Motor 3/Amps/Low ...	Critical	25	Low Amps	Active, Unack...
6/12/19, 12:47 PM	Motors/Motor 5/Amps/Low Amps	Critical	50	Low Amps	Active, Unack...
6/12/19, 12:47 PM	Motor Plant/Motor 1/Amps/Low ...	Critical	23	Low Amps	Active, Unack...
6/12/19, 12:47 PM	Ramp/Ramp8/OPC Alarm	High	316.6133	OPC Alarm	Active, Unack...
6/11/19, 3:26 PM	Tank Level 2/Low SP2	Critical	22	Low SP2	Cleared, Unac...
6/12/19, 12:17 PM	Motors/Motor 4/Amps/Low Amps	Critical	53	Low Amps	Cleared, Unac...

Component Palette Icon:

Alarm Status Table

INDUCTIVE UNIVERSITY

Alarm Status Table

Watch the Video

Description

The alarm status table displays the current state of the alarm system. It can be configured to show active, unacknowledged, cleared, and acknowledged alarms. By default it shows all non-cleared/non-ack'ed alarms.

Acknowledgement is handled by selecting (checking) alarms and pressing the "Acknowledge" button. If any of the selected alarms require acknowledge notes, then a small text area will be presented in which the operator must add notes to the acknowledgement.

Selecting/Checking Alarms

The Alarm Status Table component allows you to select an individual alarm, multiple alarms, or the Select All checkbox in the header bar. You can also use the Shift+Click multi select feature to select a range of alarms for acknowledging and shelving. Check one alarm and Shift+Click another alarm several rows down. All of the alarms between them, including the one you shift clicked, will be selected.

Shelving is supported by pressing the "Shelve" button when an alarm is selected. This will temporarily remove the alarm from the entire alarm system (not just the local client). When the time is up, if the alarm is still active, it will pop back into the alarm system. The times shown to the user are customizable by editing the values inside the "Shelving Times" dataset property. The alarms that have been shelved can be un-shelved by pushing the shelf management button in the lower right-hand side of the component.

If a more simplified alarm status table is needed, many of the features of the status table can be removed, for example, the header, footer, and multi-selection checkboxes. If a very short alarm status table is needed, turn on the "Marquee Mode" option, which will automatically scroll through any alarms if there is not enough vertical space to show all of them at once.

To change the columns that are displayed, the order of the columns, and/or the column width, put the Designer into preview mode. Then right-click on the table header to show/hide columns. Click and drag to re-order columns, and drag the margins of the columns to resize their width. No further action is necessary - the column configuration will remain in place after the window is saved.

For alarms that originate from Tags that have Tag history turned on, users can see an automatic ad-hoc chart for the value of the source Tag by pressing the chart button.

For information on how to configure the Alarm Status Table refer to Alarming in Vision.

Properties

Name	Description	Property Type
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. The border is unaffected by rotation.	Border
Chart Resolution	The resolution for the ad-hoc tag historian chart.	int
Date Format	A date format pattern used to format dates in the table. If blank, the default format for the locale is used.	String
Display Path Filter	Filter alarms by alarm display path, falling back to the source path if a custom display path isn't set. Specify multiple paths by separating them with commas. Supports the wildcard "*".	String
Duration Format	The following feature is new in Ignition version **8.0.3** Click here to check out the other new features Formats styles for fields like Active and Ack durations: Long, Short, Compact, and Abbreviated. Duration Format property, allows users to format the time units on the Active Duration column.	int
Enabled	If disabled, a component cannot be used.	boolean
Flash Interval	The time interval to use for flashing row styles.	int
Journal Name	The name of the alarm journal to query for the chart's annotations. Leave this blank to automatically pick the journal if there is only one.	String
Marquee Mode	Turn the table into a scrolling marquee	boolean
Min Priority	The minimum priority alarm to be displayed by this table.	int
Multi Select	Allow multi select. Will show/hide the checkbox column.	boolean
Name	The name of this component.	String
Notes Area Border	The border surrounding the notes area.	Border
Notes Area Font	The font for the notes area.	Font
Notes Area Location	The location of the notes display area.	int
Notes Area Size	The size of the notes area, in pixels.	int

Number Format	A number format string to control the format of the value column.	String	. r c
Provider Filter	Filter alarms by Tag provider. Specify multiple providers by separating them with commas. A value of "." denotes the default Tag provider.	String	. p i
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.
Refresh Rate	The rate at which this table will poll changes to the alarm status, in milliseconds.	long	. r a
Row Height	The height, in pixels, for each row of the table.	int	. r
Row Styles	A dataset containing the different styles configured for different alarm states.	Dataset	. r
Scroll Delay	The time (in mSec) to wait between performing each step in a scroll	int	. s a
Selected Alarms	A dataset containing each selected alarm. (Read-only)	Dataset	. s A
Selection Color	The color of the selection border. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	. s C
Selection Thickness	The size of the selection border.	int	. s T
Shelving Times	This dataset holds the times that are suggested when shelving an alarm. Allowable units are "second", "minute", or "hour".	Dataset	. s i
Show Ack Button	Show the acknowledge button on the footer panel.	boolean	.
Show Active and Acked	Show alarms that are active and acknowledged.	boolean	. a c
Show Active and Unacked	Show alarms that are active and unacknowledged.	boolean	. a c
Show Chart Button	Show the chart button on the footer panel.	boolean	. s r
Show Clear and Acked	Show alarms that are cleared and acknowledged.	boolean	. c A
Show Clear and Unacked	Show alarms that are cleared and unacknowledged.	boolean	. c U

Show Details Button	Show the view details button on the footer panel.	boolean	. s a
Show Footer	Show a footer with acknowledge and shelf functions below the alarms.	boolean	. s e
Show Header Popup	Toggles the table header's built-in column selection popup menu.	boolean	. s e P
Show Manage Shelf Button	Show the manage shelf button on the footer panel.	boolean	. s a
Show Shelve Button	Show the shelve button on the footer panel.	boolean	. s v
Show Table Header	Toggles visibility of the table's header.	boolean	. s e
Sort Oldest First	Sort times by oldest first.	boolean	. s t
Sort Order	The default sort order for alarms in the status table.	int	. s
Source Filter	Filter alarms by alarm source path. Specify multiple paths by separating them with commas. Supports the wildcard "*".	String	. s e
Stay Delay	The time (in mSec) to wait between scrolls	int	. s
Table Background	The background of the alarm table. See Color Selector.	Color	. t k
Table Header Font	The font for the table header.	Font	. t c
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	. t e
Visible	If disabled, the component will be hidden.	boolean	.
Deprecated Properties			
Data Quality	The data quality code for any Tag bindings on this component.	int	. c t

Scripting

Scripting Functions

- Description

  This specialized print function will paginate the table onto multiple pages. This function accepts keyword-style invocation.

- Keyword Args

  boolean fitWidth - If true, the table's width will be stretched to fit across one page's width. Rows will still paginate normally. If false, thetable will paginate columns onto extra pages. (default = true) [optional]

  string headerFormat - A string to use as the table's page header. The substring "{0}" will be replaced with the current page number. (default = None) [optional]

  string footerFormat - A string to use as the table's page footer. The substring "{0}" will be replaced with the current page number. (default = "Page {0}") [optional]

  boolean showDialog - Whether or not the print dialog should be shown to the user. Default is true. [optional]

  boolean landscape - Used to specify portrait (0) or landscape (1) mode. Default is portrait (0). [optional]

- Return

  Boolean- True if the print job was successful.

- Description

  Returns a dataset of the alarms currently displayed in the Alarm Status Table component. The columns will be: EventId, Source, DisplayPath, EventTime, State, and Priority.

- Keyword Args

  None

- Return

  Dataset - A dataset of alarms.

Extension Functions

- Description

  Returns a popup menu that will be displayed when the user triggers a popup menu (right click) in the table. Use *system.gui.PopupMenu()* to create the popup menu.

- Parameters

  Component self- A reference to the component that is invoking this function.

  List selectedAlarmEvents - The alarm events selected on the Alarm Status Table. For an individual alarm Event, call *alarmEvent.get('propertyName')* to inspect. Common properties: 'name', 'source', 'priority'.

- Return

  Object - the popup menu.

- Description

  Called for each event loaded into the alarm status table. Return false to hide this event from the table. This code is executed in a background thread.

- Parameters

  Component self- A reference to the component that is invoking this function.

  Alarm Event alarmEvent - The alarm event itself. Call alarmEvent.get('propertyName') to inspect. Common properties: 'name', 'source','priority'.

- Return

  Boolean- Returns true or false for every alarm event in the table. True will show the alarm. False will not show the alarm.

- Description

  Returns a boolean that represents whether the selected alarm can be acknowledged

- Parameters

  Component self- A reference to the component that is invoking this function.

  List selectedAlarmEvents - The alarm events selected on the Alarm Status Table. For an individual alarmEvent, call alarmEvent.get('propertyName') to inspect. Common properties: 'name','source','priority'.

- Return

  Boolean- Returns true or false for every alarm event in the table.

- Description

  Returns a boolean that represents whether the selected alarm can be shelved.

- Parameters

  Component self- A reference to the component that is invoking this function.

  List selectedAlarmEvents - The alarm events selected on the Alarm Status Table. For an individual alarmEvent, call alarmEvent.get('propertyName') to inspect. Common properties: 'name', 'source', 'priority'.

- Return

  Boolean- Returns true or false for every alarm event in the table.

- Description

  Called when an alarm is double-clicked on to provide custom functionality.

- Parameters

  Component self- A reference to the component that is invoking this function.

  Alarm Event alarmEvent - The alarm event that was double clicked. For an individual alarmEvent, call alarmEvent.get('propertyName') to inspect. Common properties: 'name', 'source', 'priority'.

- Return

  Nothing

- Description

  Called when the Acknowledge button is pressed; the script runs before the ack happens. Return False to abort the acknowledgement, return True to continue as normal.

- Parameters

  Component self- A reference to the component that is invoking this function.

  List alarms - A list of the alarms to be acknowledged.

- Return

  Boolean- Returns true or false for every alarm event that is selected.

Event Handlers

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
. property Name	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

Alarm Row Styles

The Alarm Status Table has a customizer.

- Vision Component Customizers

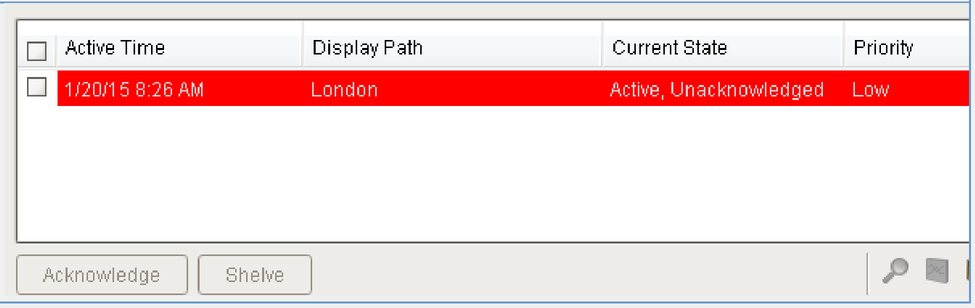
Examples

Code Snippet

```
#The following code is an example of the filter alarm expression function.
#The function results in advanced filtering for the alarm table.
#In this example the alarm table will only show alarms with a name that matches the value of
the "AreaName" property located on the container the Alarm Status Table resides in.

name = self.parent.AreaName
if name == alarmEvent.get('name'):
        return True
else:
        return False
```

Gallery

Alarm Status Table with a Single Alarm

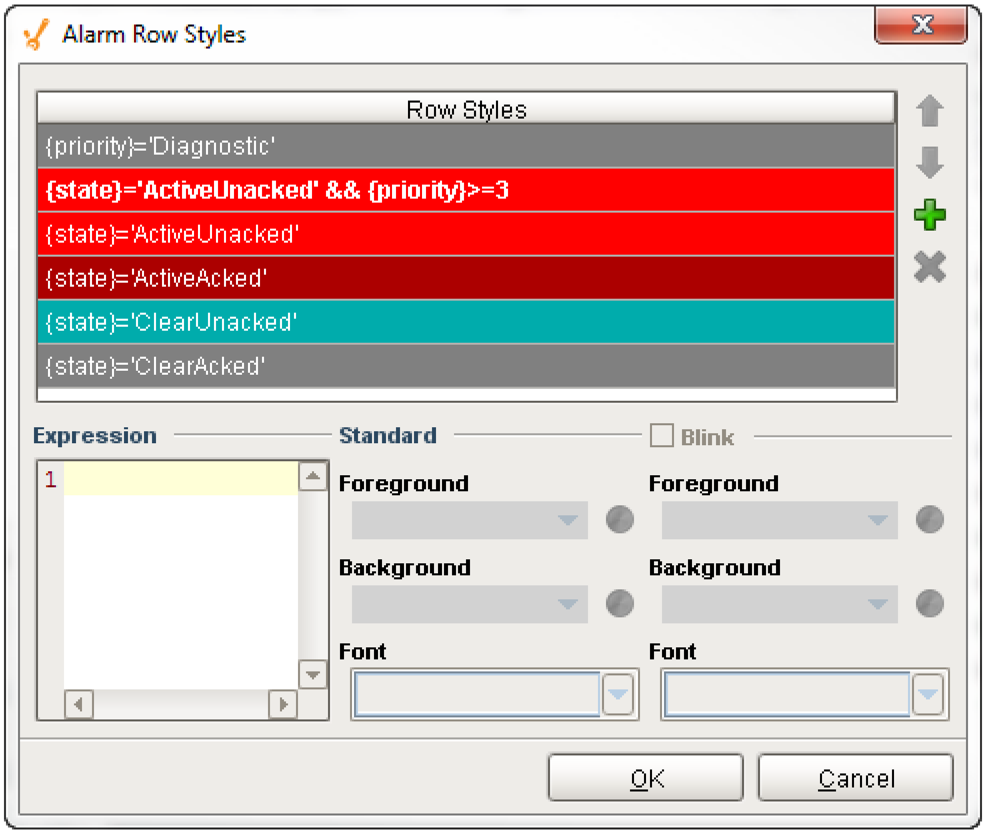


☐	Active Time	Display Path	Current State	Priority
☐	1/20/15 8:26 AM	London	Active, Unacknowledged	Low

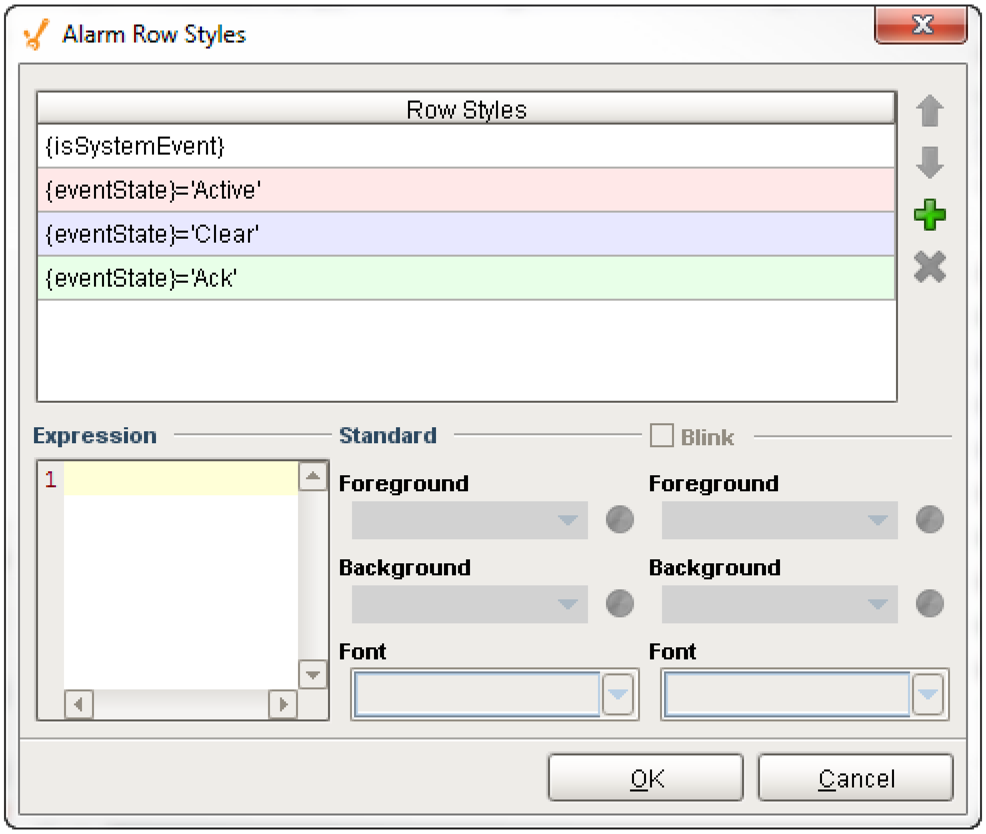
Vision - Alarm Row Style Customizer

Alarm Row Styles - Alarm Status Table Table

Alarm Row Styles - Alarm Journal



{priority}='Diagnostic'
{state}='ActiveUnacked' && {priority}>=3
{state}='ActiveUnacked'
{state}='ActiveAcked'
{state}='ClearUnacked'
{state}='ClearAcked'



{isSystemEvent}
{eventState}='Active'
{eventState}='Clear'
{eventState}='Ack'

Description

The Alarm Row Styles Customizer manages the way the Alarm Status Table and the Alarm Journal Table render each alarm. The Alarm Row Styles Customizer allows you to change the styles of the alarms and the logic that governs each style. Both the Alarm Status Table and the Alarm Journal Table evaluate each alarm and applies the logic of the expression block to decide to implement a style. If the expression returns a logical "True" then the Alarm Row Styles Customizer applies the color formatting options defined in the area to the right of the Expression block. If the expression returns a logical "False" then the Alarm Row Styles Customizer evaluates the next expression associated with the next row style. The process continues until an expression returns a logical "True." There can be many rows with different logic and styles. You can add and remove rows by selecting the "plus" button or "delete" button.

Customizers

The Alarm Row Styles Customizer is used by both the Alarm Status Table and the Alarm Journal Table components. Each table comes with their own predefined set of colors. The Alarm Row Styles Customizer is where you can modify an existing style, add more styles, delete a style, and change the order. Each row style has an expression, a color, and the option to make it blink. The Alarm Row Styles Customizer already has some preset states and predefined styles to help you get started. It works by changing colors on each of the individual rows styles based on the state of the alarm.

Alarm Rows Styles Customizer - Property Descriptions

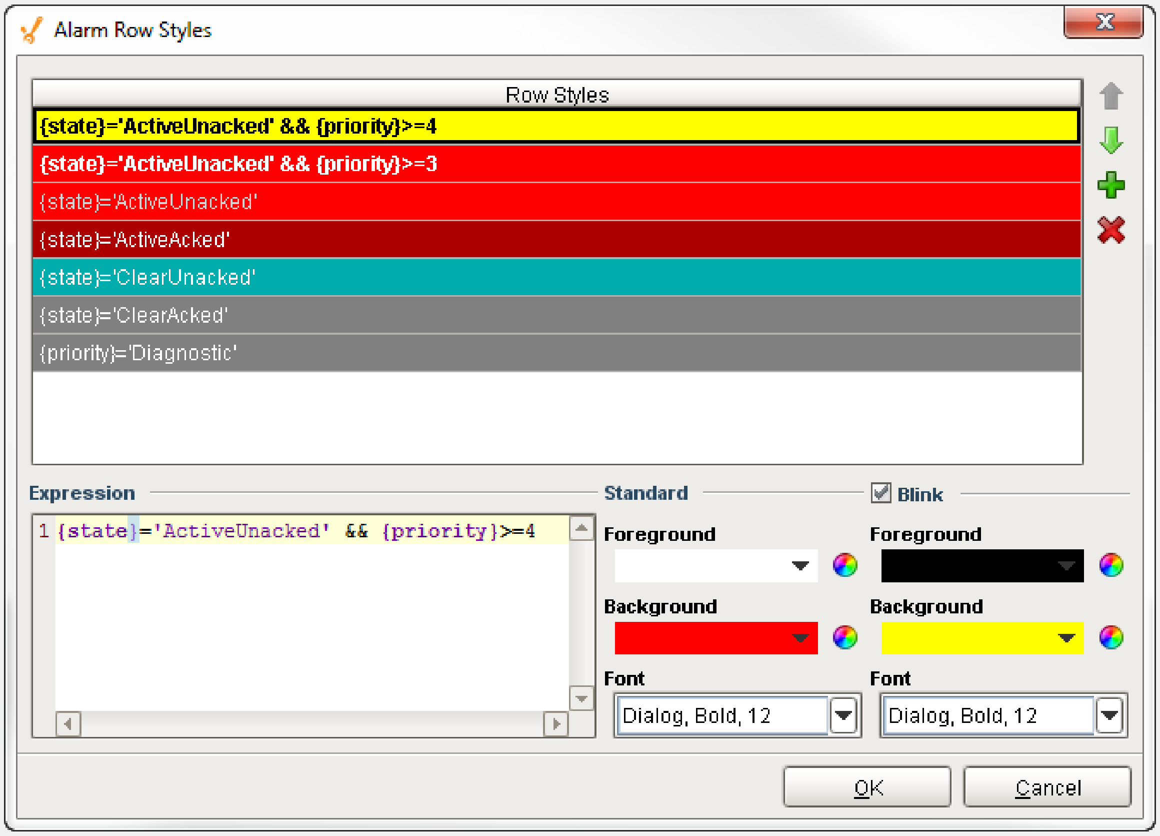
Property	Description
Row Styles	Each row has a unique style associated with each of the alarm states. You can add and delete row styles, and change the order of the rows with the up or down arrow buttons.
Expression	Each style has an expression. The expression allows you to do any evaluation you want using any parts of the alarm: Priority, State, Display Path, Active Time, and Clear Time.
Standard	One solid color on a row style.
Blink	Two colors alternately flashing on a row style used to draw attention. Commonly used for critical alarms to draw the operator's attention.
Foreground	Specifies the color of the text.
Background	Specifies the color of the row.
Font	Specifies the font type, font size, and style.

- Alarm Status - Row Styles
- Alarm Journal - Row Styles

Examples

In these examples, the Alarm Row Styles was modified for the Alarm Status Table and the Alarm Journal Table to add another row style for Active, Unacknowledged alarms with a priority 4, or Critical alarms.

Alarm Status Table - Alarm Row Styles



{state}='ActiveUnacked' && {priority}>=4
{state}='ActiveUnacked' && {priority}>=3
{state}='ActiveUnacked'
{state}='ActiveAcked'
{state}='ClearUnacked'
{state}='ClearAcked'
{priority}='Diagnostic'

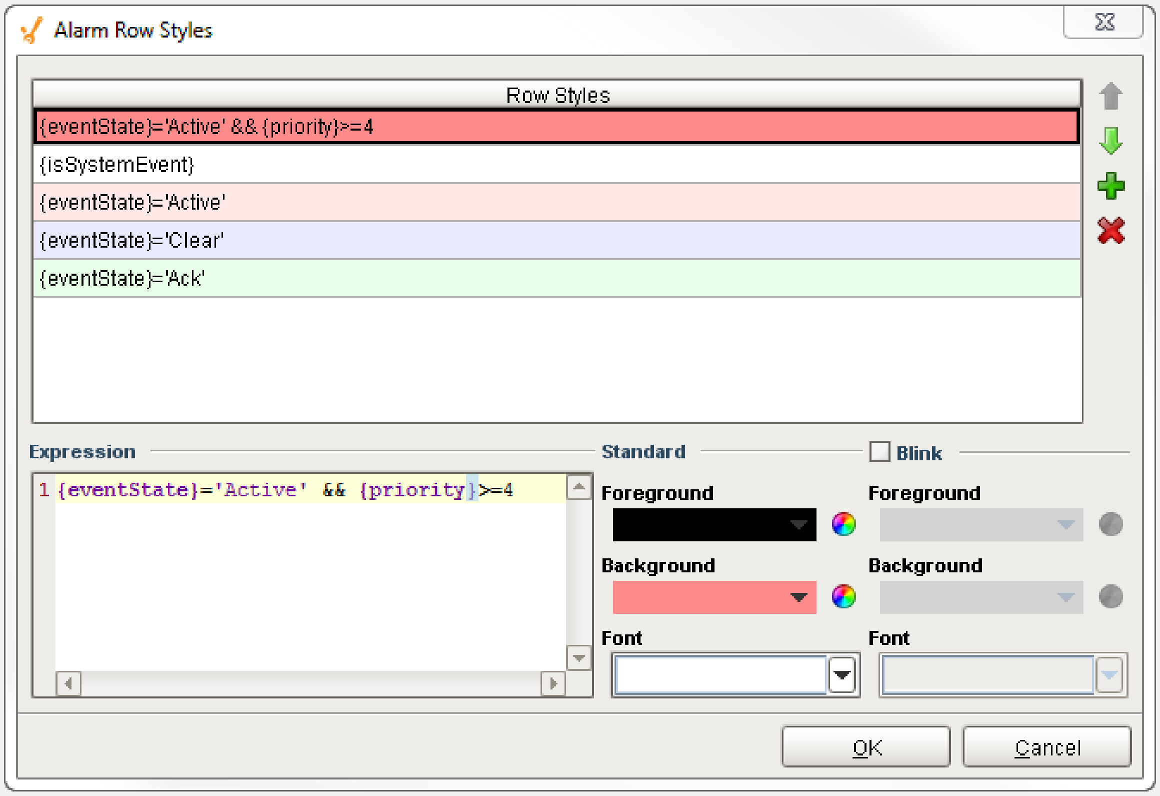
Expression

```
{state}='ActiveUnacked' && {priority}>=4
```

Standard: Foreground; Background; Font: Dialog, Bold, 12

Blink [✓]: Foreground; Background; Font: Dialog, Bold, 12

Alarm Journal Table - Alarm Row Styles



{eventState}='Active' && {priority}>=4
{isSystemEvent}
{eventState}='Active'
{eventState}='Clear'
{eventState}='Ack'

Expression

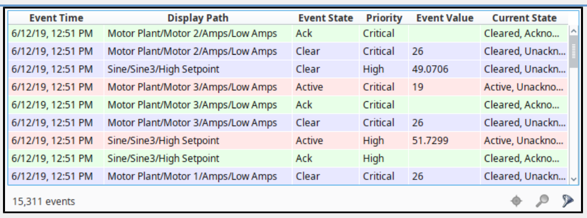
```
{eventState}='Active' && {priority}>=4
```

Standard: Foreground; Background; Font

Blink: Foreground; Background; Font

Vision - Alarm Journal Table

General



Event Time	Display Path	Event State	Priority	Event Value	Current State
6/12/19, 12:51 PM	Motor Plant/Motor 2/Amps/Low Amps	Ack	Critical		Cleared, Ackno...
6/12/19, 12:51 PM	Motor Plant/Motor 2/Amps/Low Amps	Clear	Critical	26	Cleared, Unackn...
6/12/19, 12:51 PM	Sine/Sine3/High Setpoint	Clear	High	49.0706	Cleared, Unackn...
6/12/19, 12:51 PM	Motor Plant/Motor 3/Amps/Low Amps	Active	Critical	19	Active, Unackno...
6/12/19, 12:51 PM	Motor Plant/Motor 3/Amps/Low Amps	Ack	Critical		Cleared, Ackno...
6/12/19, 12:51 PM	Motor Plant/Motor 3/Amps/Low Amps	Clear	Critical	26	Cleared, Unackn...
6/12/19, 12:51 PM	Sine/Sine3/High Setpoint	Active	High	51.7299	Active, Unackno...
6/12/19, 12:51 PM	Sine/Sine3/High Setpoint	Ack	High		Cleared, Ackno...
6/12/19, 12:51 PM	Motor Plant/Motor 1/Amps/Low Amps	Clear	Critical	26	Cleared, Unackn...

15,311 events

Component Palette Icon:

Alarm Journal Table

INDUCTIVE UNIVERSITY

Alarm Journal Table

Watch the Video

Description

The alarm journal table provides a built-in view to explore alarm history that has been stored in an alarm journal. If you only have one alarm journal specified on your Gateway, then you do not need to specify the journal name. If you have more than one specified, then you need to provide the name of the journal you'd like to query.

The journal table shows the alarm history that is found between the Start Date and End Date properties. When you first put an alarm journal table on a window, these properties will be set to show the most recent few hours of journal history. Note that without further configuration, the journal table will always show the few hours before it was created. To properly configure an alarm journal table, bind its start and end date properties to something what will update, such as the Date Range component or expressions involving the time now(). This way, you can configure it so that operators can choose the time to display, or have dates will be update automatically to have it poll.

To change the columns that are displayed, the order of the columns, and/or the column width, put the Designer into preview mode. Then right-click on the table header to show/hide columns. Click and drag to re-order columns, and drag the margins of the columns to resize their width. No further action is necessary - the column configuration will remain in place after the window is saved.

Additional examples of configuring the Alarm Journal Table can be found on the Alarm Journal Table Component page.

Properties

Name	Description	Property Type	Scripting	Category
Acked Events	Show acked events.	boolean	.includeAckedEvents	Filters
Active Events	Show active events.	boolean	.includeActiveEvents	Filters
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. The border is unaffected by rotation.	Border	.border	Common
Cleared Events	Show cleared events.	boolean	.includeClearedEvents	Filters

Date Format	A date format pattern used to format dates in the table. If blank, the default format for the locale is used.	String	.dateFormat	Appearance
Display Path Filter	Filter alarms by alarm display path, falling back to the source path if display path isn't set. Specify multiple paths by separating them with commas. Supports the wildcard "*".	String	.displayPathFilter	Filters
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
End Date	The ending date for the displayed history range. If left blank, will default to the current time when the component was loaded.	Date	.endDate	Behavior
Is Filtered	True if the results are filtered. (Read-only)	boolean	.isFiltered	Behavior
Journal Name	The name of the alarm journal to query.	String	.journalName	Behavior
Max Priority	The maximum priority to display.	int	.maximumPriority	Filters
Min Priority	The minimum priority to display.	int	.minimumPriority	Filters
Name	The name of this component.	String	.name	Common
Notes Area Border	The border surrounding the notes area.	Border	.notesAreaBorder	Appearance
Notes Area Font	The font for the notes area.	Font	.notesAreaFont	Appearance
Notes Area Location	The location of the notes display area.	int	.notesAreaLocation	Appearance
Notes Area Size	The size of the notes area, in pixels.	int	.notesAreaSize	Appearance
Number Format	A number format string to control the format of the value column.	String	.numberFormat	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Read Timeout	The timeout, in milliseconds, for running the alarm history query.	int	.readTimeout	Behavior
Row Height	The height, in pixels, for each row of the table.	int	.rowHeight	Appearance
Row Styles	A dataset containing the different styles configured for different alarm states.	Dataset	.rowStyles	Appearance
Search String	Filter alarms by searching for a string in both source path and display path.	String	.searchString	Filters
Selected Alarms	A dataset containing each selected alarm. (Read-only)	Dataset	.selectedAlarms	Data
Selection Color	The color of the selection border. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.selectionColor	Appearance

Selection Thickness	The size of the selection border.	int	.selectionThickness	Appearance
Show Table Header	Toggles visibility of the table's header.	boolean	.showTableHeader	Appearance
Source Filter	Filter alarms by alarm source path. Specify multiple paths by separating them with commas. Supports the wildcard "*".	String	.sourceFilter	Filters
Start Date	The starting date for the displayed history range. If left blank, will default to 8 hours prior to when the component was loaded.	Date	.startDate	Behavior
System Events	Show system events such as startup and shutdown.	boolean	.includeSystemEvents	Filters
Table Background	The background of the alarm table. See Color Selector.	Color	.tableBackground	Appearance
Table Font	The font for the Alarm Journal's rows.	Font	.font	Appearance
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

- Description

  This specialized print function will paginate the table onto multiple pages.This function accepts keyword-style invocation.

- Keyword Args

  boolean fitWidth - If true, the table's width will be stretched to fit across one page's width. Rows will still paginate normally. If false, the table will paginate columns onto extra pages. (default = true) [optional]

  String headerFormat - A string to use as the table's page header. The substring "{0}" will be replaced with the current page number. (default = None) [optional]

  String footerFormat - A string to use as the table's page footer. The substring "{0}" will be replaced with the current page number. (default = "Page {0}") [optional]

  boolean showDialog - Whether or not the print dialog should be shown to the user. Default is true. [optional]

  boolean landscape - Used to specify portrait (0) or landscape (1) mode. Default is portrait (0). [optional]

- Return

  boolean - True if the print job was successful.

- Description

  Returns a dataset of the alarms currently displayed in the Alarm Journal Table component. The columns will be: EventId, Source, DisplayPath, EventTime, State, Priority and IsSystemEvent

- Keyword Args

  None

- Return

  Dataset - A dataset of alarms.

Extension Functions

- Description

  Returns a popup menu that will be displayed when the user triggers a popup menu (right click) in the table. Use system.gui.createPopupMenu() to create the popup menu.

- Parameters

  Component self - A reference to the component that is invoking this function.

  List selectedAlarmEvents - The alarm events selected on the Alarm Status Table. For an individual alarmEvent, call alarmEvent.get('propertyName') to inspect. Common properties: 'name', 'source', 'priority'.

- Return

  JPopupMenu - A popup menu that was created with system.gui.createPopupMenu()

- Description

  Called for each event loaded into the alarm status table. Return false to hide this event from the table. This code is executed in a background thread.

- Parameters

  Component self - A reference to the component that is invoking this function.

  Alarm Event alarmEvent - The alarm event itself. Call alarmEvent.get('propertyName') to inspect. Common properties: 'name', 'source', 'priority'.

- Return

  Boolean

- Description

  Called when an alarm is double-clicked on to provide custom functionality. Does not return a value.

- Parameters

  Component self - A reference to the component that is invoking this function.

  Alarm Event alarmEvent - The alarm event itself. Call alarmEvent.get('propertyName') to inspect. Common properties: 'name', 'source', 'priority'.

- Return

  Nothing

Event Handlers

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event.
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when a component that had the input focus lost it to another component.

.source	The component that fired this event
.oppositeComponent	The other component involved in this focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

An integer that indicates whether the state was changed to "Selected" (on) or "Deselected" (off). Compare this to the event object's constants to determine what the new state is.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is released and the source component has the input focus. Works for all characters, including non-printable ones, such as SHIFT and F3.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is pressed and then released when source component has the input focus. Only works for characters that can be printed on the screen.

.source	The component that fired this event.
.keyCode	The key code for this event. Used with the `keyPressed` and `keyReleased` events. See below for the key code constants.
.keyChar	The character that was typed. Used with the `keyTyped` event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. the left and right shift keys. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the documentation, the keyTyped event always has a location of KEY_LOCATION_UNKNOWN.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires *aft er* the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
. clickCo unt	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
. popupT rigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
. altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
. control Down	True (1) if the Control key was held down during this event, false (0) otherwise.
. shiftDo wn	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
. clickCo unt	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
. popupT rigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
. altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
. control Down	True (1) if the Control key was held down during this event, false (0) otherwise.
. shiftDo wn	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a *bindable* property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

The Alarm Row Styles Customizer manages the way the Alarm Journal renders each alarm.

- Vision Component Customizers

Examples

There are no examples associated with this component.

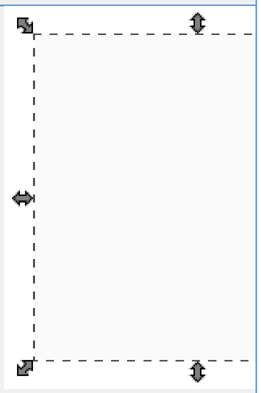
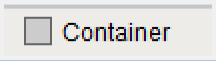
Vision - Containers Palette

Container Components

The following components give you the ability to group and display components.

In This Section ...

Vision - Container

General

Component Palette Icon:


Description
The container is a very important component. All components are always inside of a container, except for the special "Root Container" of each window (see Window Properties). A container is different than normal components in that it can contain other components, including other containers. Uses for containers include: • **Organization** - Containers can be used to group components together. These components can then easily be moved, copied, or deleted as a group. Furthermore, they will all be organized inside of their parent container in the project navigation tree, which makes them easier to find. • **Re-usability** - Containers allow a unique opportunity to create a complex component that is made up of multiple other components. The Container's ability to have dynamic properties aids this greatly. For instance, if you wanted to make your own custom HOA control, you can put three buttons inside of a container and configure them to all use a 'status' property that you add to their parent Container. Now you have built an HOA control that can be re-used and treated like its own component. The possibilities here are endless. Create a date range control that generates an SQL WHERE clause that can be used to control Charts and Tables. Create a label/button control that can be used to display datapoints, and pop up a parameterized window that displays meta-data (engineering units, physical location, notes, etc.) about that datapoint. Creating re-usable controls with Containers containing multiple components is the key to rapid application development. • **Layout** - Containers are a great way to improve window aesthetics through borders and layout options. To move a container around on a window, you need to hold the alt key while clicking and dragging.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Combine Repaints	Set this to true for containers with many sub-components that need to redraw frequently (flashing, rotating, animating).	boolean	.combineRepaints	Behavior
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Font	Font of text on this component.	Font	.font	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Opaque	If false, backgrounds are not drawn. If true, backgrounds are drawn.	boolean	.opaque	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Texture	Background texture image for this container.	String	.texturePath	Appearance
Tile Optimized	If true, this container's children should never overlap, and you'll get better painting performance.	boolean	.optimizedDrawingEnabled	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

This event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

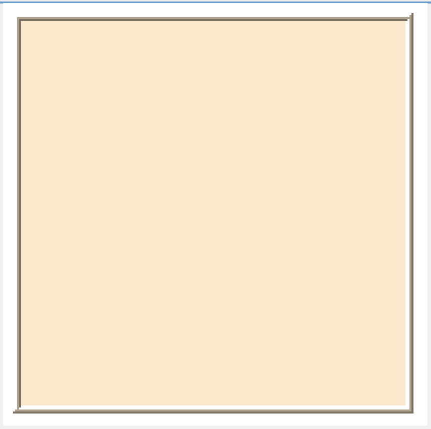
.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers
- Style Customizer

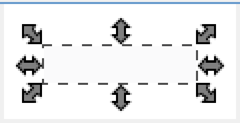
Examples

Customized Container with Border



Property Name	Value
Border	Bevel (Double)
Background Color	255,232,204

Vision - Template Repeater

General

Component Palette Icon:

INDUCTIVE UNIVERSITY

Template Repeater

Watch the Video

Description

The Template Repeater repeats instances of templates any number of times. It can arrange them vertically, horizontally, or in a "flow" layout, which can either be top-to-bottom or left-to-right. If there are too many to fit, a scrollbar will be shown. This makes it easy to quickly create screens that represent many similar pieces of equipment. It also can be used to create screens that are dynamic, and automatically configure themselves based on configuration stored in a database or tag structure. When first dropped on a window, the template repeater will look like any other empty container. To select the template to repeat, configure the repeater's Template Path property. There are two ways to set how many times the template should repeat:

- Count - The template will be repeated X times, where X is the value of "Repeat Count". The repeat count starts at zero and increments X amount of times. Each value for X will be inserted into the custom property of the template that will be repeated. Template repeater inserts the value of X into the custom property on the template with the same name as the template repeater's "Index Property Name." For example, if the template has a custom property of "index" and the template repeater's Index Property Name is also "index," then the template will be repeated X many time with the value of X being inserted into the template's custom property called "index."

- Dataset - The template will be repeated once for each row in the "Template Parameters" dataset. The template's custom properties with the same names as the dataset's column names will assume the values of each row of the dataset.

An Example of configuring the Template Repeater can be found on the Using the Template Repeater page.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. The border is unaffected by rotation.	Border	.border	Common
Flow Alignment	Alignment for "Flow" layout style.	int	.flowAlignment	Appearance
Flow Direction	When the layout style is flow, this property controls if the components in the container flow horizontally or vertically.	int	.flowDirection	Appearance

Horizontal Gap	The gap size to use for horizontal gaps.	int	.horizontalGap	Appearance
Index Parameter Name	A name of an integer parameter on the template that will be set to an index number.	String	.indexParamName	Behavior
Layout Style	Controls how the repeated template instances are laid out inside the repeater.	int	.layoutStyle	Appearance
Marquee Mode	Turn the repeater into a scrolling marquee.	boolean	.marqueeMode	Behavior
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Repeat Behavior	"Count" will repeat the template a number of times, assigning each template an index number. "Dataset" will repeat the template once per row in the template parameter's dataset.	int	.repeatBehavior	Behavior
Repeat Count	The template will be repeated this many times, if the repeat behavior is set to "Count."	int	.repeatCount	Behavior
Scroll Delay	The time (in milliseconds) to wait between performing each step in a scroll.	int	.scrollDelay	Behavior
Stay Delay	The time (in milliseconds) to wait between scrolls.	int	.stayDelay	Behavior
Template Parameters	This dataset will be used to control the number of templates and the parameters set on the templates if the repeat behavior is set to "Dataset."	Dataset	.templateParams	Behavior
Template Path	The path to the template that this container will repeat.	String	.templatePath	Behavior
Vertical Gap	The gap size to use for vertical gaps.	int	.verticalGap	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

- Description

  Returns a list of templates loaded into the Template Repeater. Properties on the components within each instance can be references by calling getComponent().

- Parameters

  None

- Return

  List of Templates

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

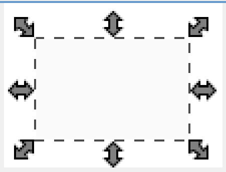
- Vision Component Customizers

Examples

Code Snippet: getLoadedTemplates()

```
#This script will call getLoadedTemplates() on a Template Repeater, and
#then print the text property of a Label component in each instance

#Store a reference to the Template Repeater component in a variable
repeater = event.source.parent.getComponent('Template Repeater')

#Store the list of templates in another variable
templateList = repeater.getLoadedTemplates()

#Iterate through the list
for template in templateList:
        #find a component named "Label" in the instance,
        #and print the value of the text property
        print template.getComponent('Label').text
```

Vision - Template Canvas

General

Component Palette Icon:
Template Canvas

INDUCTIVE UNIVERSITY

Template Canvas

Watch the Video

Description
The template canvas is similar to the template repeater but allows for more control of the templates than the template repeater. The Templates property on the template canvas is a dataset. Each row in this dataset represents a manifestation of a template. It can be the same template or a different template on each row. This dataset allows for control over the size, position and layout of the template. There are two methods of controlling the layout of each template inside the template canvas: • **Absolute Positioning** - The location of the template is explicitly managed through the "X" and "Y" columns of the Templates property's dataset. Consequently the columns labeled Width and Height control the size of the template. • **Layout Positioning** - The template canvas uses "MiGLayout" to manage the location of the template. MigLayout is a common albeit complicated layout methodology. It supports layouts that wrap the templates automatically as well as docking the template to one side of the template canvas. You can learn more about MiG Layout at http://www.miglayout.com In addition, control over data inside each template can be achieved by adding a column with the name Parameters to the dataset and populating this column with dictionary style key words and definitions. Additional templates can be added to the template canvas by inserting an additional row to the Templates property's dataset. The same applies to removing the templates but with removing the rows from the dataset.

Properties

Name	Description	Property Type	S
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	. b r
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. The border is unaffected by rotation.	Border	.
Layout Constraints	The overall layout constraints for the canvas.	String	. l r
Name	The name of this component.	String	.
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.
Scroll Behavior	Controls which direction(s) the canvas will scroll in.	int	. s a
Show Loading	The following feature is new in Ignition version **8.0.6** Click here to check out the other new features If false, the loading indicator will never be shown.	boolean	. s c
Templates	A dataset containing a row per template to instantiate.	Dataset	. t
Visible	If disabled, the component will be hidden.	boolean	.
Deprecated Properties			
Data Quality	The data quality code for any Tag bindings on this component.	int	. c t

Scripting

Scripting Functions

- Description

  Returns a list of the templates that comprise the template canvas.
- Parameters

  Nothing
- Return

  List - A list of VisionTemplate definitions. Each instance in the canvas will return its definition's name. The names of each instance can be accessed with getInstanceName(). Individual components in each instance can accessed with getComponent().
- Description

  Obtains the designated template object from the template canvas.
- Parameters

  String name- The name of the template as defined by the "name" column of the dataset populating the template canvas.
- Return

  VisionTemplate - Returns the template instance. Properties on the instance can be access by calling .propertyName

Extension Functions

- Description

  This will be called once per template that is loaded. This is a good chance to do any custom initialization or setting parameters on the template.
- Parameters

  Component self- A reference to the component that is invoking this function.

  Vision Template template - The template. The name of the template in the dataset will be available as template.instanceName
- Return

  Nothing

Event Handlers

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

This component has its own customizer called the Template Canvas Customizer. The Template Canvas Customizer allows you to create multiple instances of a template. Here is where you can configure some of the properties of the template instance that are inside the Template Canvas. To edit a template instance, select it from the Instances list. To cancel your edit and add a new instance instead, click the Cancel button in the bottom left.

Templates Property

The "Templates" property, in the Property Editor, stores all the data that is entered into the customizer. New template instances can be created directly on the "Templates" property as well. To edit or view the dataset, click the Dataset Viewer next to the "Templates" property.

Template Canvas Customizer - Property Description

Property	Description
Instances	A list of the templates currently in the Template Canvas.
Add/Edit Instances	Section of the Template Canvas Customizer where you add new instances and edit existing instances. Select an instance from above to edit that instance.
Name	Name of the selected template instance.
Z-Index	The index position along the Z axis that should be used for the instance. If left empty, then Z order will be determined by the row index position of the instance as it sits in the Template Canvas' **Templates** property.
Template	The template path for the selected template instance.
Absolute Positioning	Sets the position and size of the components inside the template. In order from left to right, the four boxes are X, Y, Width, and Height.
Layout Positioning	Uses MiGLayout to manage template location. It allows you to easily determine the layout of components or templates within a container (i.e., "span,wrap"). To learn more, go to http://www.miglayout.com
Parameters	Shows a list of all the parameters that are defined in the selected template. Specify the values for each template parameter. To make this dynamic, you must bind the **Templates** property of the Template Canvas.

More information on the Template Canvas Customizer can be found on the Component Customizers page.

Data Types and the Parameters Field

The "Parameters" field in the customizer accepts string values, but attempts to convert the value if the underlying template parameter is set to a non-string type. In some cases this may require special formatting on the supplied string. The table below provides some examples.

Data Type	Expected Format	Format Examples

Color	Colors may be entered in as either a name, or an RBG string	`red`
		
		`0,0,255`

Date	Date objects may be entered as either a UNIX timestamp in milliseconds, or in the following notation. In all cases, quotation marks should not be added.
	
	`yyyy-MM-dd HH:mm:ss.SSS`
	`yyyy-MM-dd`
	`MM/dd/yyyy`
	`MM/dd/yyyy HH:mm:ss`
	`hh:mm:ss a`
	`hh:mm a`
	`MM/dd/yyyy hh:mm:ss a`
	`yyyy-MM-dd HH:mm:ss.SSS`
	`yyyy-MM-dd HH:mm:ss`
	`EEE MMM dd HH:mm:ss z yyyy`
		`1591374627000`
		
		`2020-03-28 06:38:00:000`

Examples

Code Snippet

```
#This example demonstrates how to pull value information from templates that are inside the
template canvas.
#This example assumes that each template has a custom property called ContentValue

#Get all the template instances of the canvas.
templates = event.source.parent.getComponent('Template Canvas').getAllTemplates()

#The templates are a list therefore you can iterate through them.
for template in templates:

        #You can access the properties of the template. This example prints the ContentValue
custom property to the console.
        print template.ContentValue
```

Code Snippet - Seach by Name

```
#This example demonstrates how to iterate through each template in a template canvas
#looking for a named instance. Once found, print the value of a property on a component in
#that instance.

#This assumes that the canvas contains a template instance named "timerTemplate" and
#a Timer component (named Timer) is inside the instance.

#Create a reference to the Template Canvas
canvas = event.source.parent.getComponent('Template Canvas')

#Retrieve all template instances in the canvas
tempInstance = canvas.getAllTemplates()

#Iterate through each template instance
for template in tempInstance:

        #Compare the name of each instance.
        if template.getInstanceName() == "timerTemplate":

                #Print the Value property on the Timer component inside the template
                print template.getComponent("Timer").value
```

Code Snippet - Read User Input Example

```
#This script will retrieve a list of all templates in a template canvas, and record user
input.

#The code is designed to work with the a User Input example,
#but can be easily modified to work with different templates.

#Reference the template canvas component, and call the getAllTemplates() method.
#This will return a list of every instance in the canvas
templateList = event.source.parent.getComponent('Template Canvas').getAllTemplates()

#Initialize a list. User input from each text field will be stored in this variable
userInput = []

#Iterate through each template instance inside the canvas
for template in templateList:

        #add the user inputted value to the userInput list. The values are originally
returned in Unicode.
        #the Python str() function is casting the Unicode values as string values.
        userInput.append(str(template.TextField_Text))

#Show the values in a messageBox. This could be replaced with an INSERT query, or some other
action.
#str() is used again to case the list as a string. This only required to work with the
messageBox function
#since the function requires a string argument be passed in
system.gui.messageBox(str(userInput))
```

Related Topics ...

- Vision Component Customizers

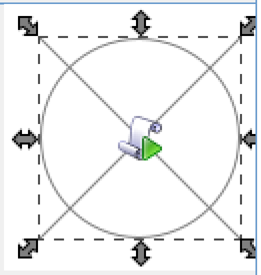
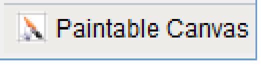
Vision - Misc Palette

Misc Components

The following components give you various ways to create or animate displays.

In This Section ...

Vision - Paintable Canvas

General

Component Palette Icon:
 Paintable Canvas

Description

The Paintable Canvas component is a component that can be custom "painted" using Jython scripting. By responding to the component's repaint event, a designer can use Java2D to draw anything within the component's bounds. Whenever any dynamic properties on the component change, the component is re-painted automatically, making it possible to create dynamic, vector-drawn components that can represent anything.

This component is an advanced component for those who are very comfortable using scripting. It is not user-friendly. The upside is that it is extraordinarily powerful, as your imagination is the only limit with what this component can be.

When you first drop a Paintable Canvas onto a window, you'll notice that it looks like a placeholder. If you switch the Designer into preview mode, you'll see an icon of a pump displayed. The pump is an example that comes pre-loaded into the Paintable C anvas. By editing the component's event scripts, you can dissect how the pump was drawn. You will notice that the script uses Java2D. You can read more about Java2D here. You will notice that as you resize the pump, it scales beautifully in preview mode. Java2D is a vector drawing library, enabling you to create components that scale very gracefully.

Tips:

- Don't forget that you can add dynamic properties to this component, and use the styles feature. Use the values of dynamic properties in your repaint code to create a dynamic component. The component will repaint automatically when these values change.
- You can create an interactive component by responding to mouse and keyboard events
- You can store your custom components on a custom palette and use them like standard components.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Focusable	If the component is focusable, it will receive keyboard input and can detect if it is the focus owner.	boolean	.focusable	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See Color Selector.	Color	.foreground	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event occurs when a component that can receive input, such as a text box, receives the input focus. This usually occurs when a user clicks on the component or tabs over to it.

.source	The component that fired this event
.oppositeComponent	The other component involved in thie focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

This event occurs when a component that had the input focus lost it to another component.

.source	The component that fired this event
.oppositeComponent	The other component involved in thie focus change. That is, the component that lost focus in order for this one to gain it, or vise versa.

An integer that indicates whether the state was changed to "Selected" (on) or "Deselected" (off). Compare this to the event object's constants to determine what the new state is.

.source	The component that fired this event
.keyCode	The key code for this event. Used with the keyPressed and keyReleased events.
.keyChar	The character that was typed. Used with the keyTyped event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a key board, e.g. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when a key is released and the source component has the input focus. Works for all characters, including non-printable ones, such as SHIFT and F3.

.source	The component that fired this event
.keyCode	The key code for this event. Used with the keyPressed and keyReleased events.
.keyChar	The character that was typed. Used with the keyTyped event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a keyboard, e.g. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.keyCode	The key code for this event. Used with the keyPressed and keyReleased events.
.keyChar	The character that was typed. Used with the keyTyped event.
.keyLocation	Returns the location of the key that originated this key event. Some keys occur more than once on a key board, e.g. Additionally, some keys occur on the numeric keypad. This provides a way of distinguishing such keys. See the KEY_LOCATION constants in the
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that coused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that coused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that coused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that coused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that coused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that coused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.graphics	An instance of java.awt.Graphics2D that can be used to paint this component. The point (0,0) is located at the upper left of the component.
.width	The width of the paintable area of the component. This takes into account the component's border.
.height	The height of the paintable area of the component. This takes into account the component's border.

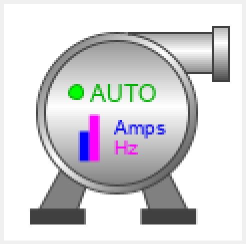
Customizers

- Vision Component Customizers
- Style Customizer

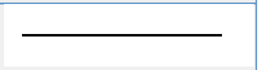
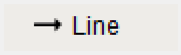
Examples

There are no examples associated with this component. However examples are available in the component itself.

The component comes prescripted to render the following pump:



Vision - Line

General

Component Palette Icon:


Description
The line component displays a straight line. It can run north-south, east-west, or diagonally. You can add arrows to either side. The line can be dashed using any pattern you want. You can even draw the line like a sinusoidal wave!

Reporting Line Component

If you are looking for the Line component used in Reporting, refer to Report - Line Shape.

Properties

Name	Description	Property Type	Scripting	Category
Color	Set the color of the line. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.foreground	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Dash Pattern	Enter a string of comma-delimited numbers which indicate the stroke pattern for a dashed line. For instance, "3,5" means three pixels on, five pixels off.	String	.strokePattern	Appearance
Left Arrow	Draw an arrow head on the left/top of the line?	boolean	.leftArrow	Appearance
Left Arrow Size	The size of the left arrow, if present.	int	.leftArrowSize	Appearance
Line Mode	The line mode determines where in the rectangle the line is drawn.	int	.lineMode	Appearance
Line Style	The line style determines how the shape of the line looks. Options are: Plane, Dashed, Sinusoidal, Sinusoidal-Dashed, Loop, and Loop-Dashed.	int	.lineStyle	Appearance
Line Width	Set the width of the line in pixels.	int	.lineWidth	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Right Arrow	Draw an arrow head on the right/bottom of the line?	boolean	.rightArrow	Appearance
Right Arrow Size	The size of the right arrow, if present.	int	.rightArrowSize	Appearance
Sine Height	Sets the amplitude of the sine wave to be drawn.	int	.sineHeight	Appearance
Sine Length	Sets the wavelength of the sine wave to be drawn.	int	.sineLength	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions
This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
. clickCo unt	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
. popupT rigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
. altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
. control Down	True (1) if the Control key was held down during this event, false (0) otherwise.
. shiftDo wn	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
. clickCo unt	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
. popupT rigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
. altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
. control Down	True (1) if the Control key was held down during this event, false (0) otherwise.
. shiftDo wn	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
. clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
. popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
. altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
. controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
. shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

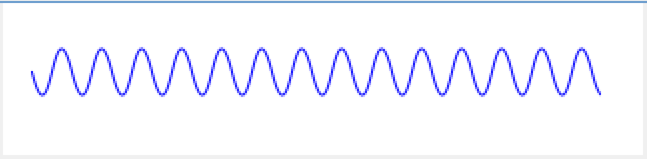
.source	The component that fired this event
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
. property Name	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers
- Style Customizer

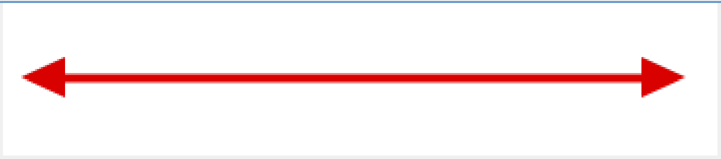
Examples

Line with Sinusoidal Pattern



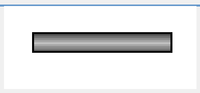
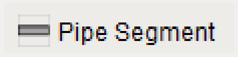
Property	Setting
Color	0,0,255
Line Style	Sinusoidal

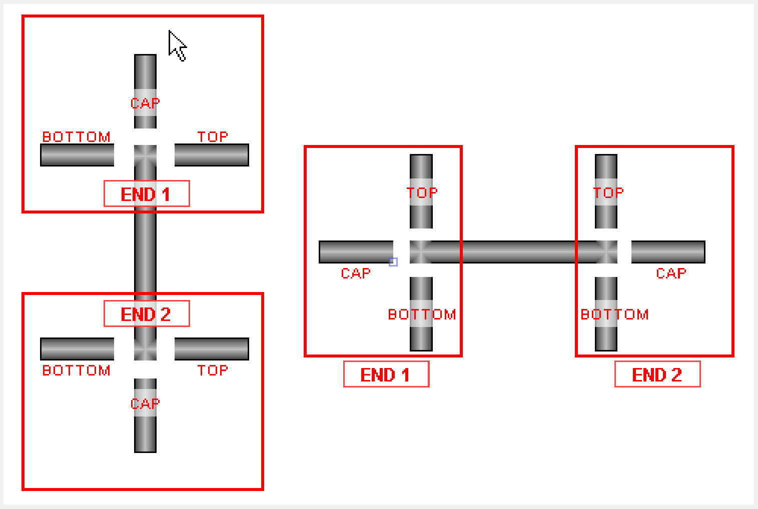
Line with Arrow Endpoints



Property	Setting
Color	217,0,0
Line Style	Plain
Left Arrow	True
Left Arrow Size	25
Line Width	4
Right Arrow	True
Right Arrow Size	25

Vision - Pipe Segment

General

Component Palette Icon:


Description
The pipe segment component displays a quasi-3D pipe. In its basic form it looks very much like a rectangle with a round gradient. The difference comes in its advanced rendering of its edges and endcaps. You can configure each pipe segment's end to mate perfectly with another pipe segment butted up against it perpendicularly. The result looks like a pipe welded together in a 90° corner. The control of the pipe's ends are done using 6 booleans - three per 'end'. End 1 is the top/left end, and End 2 is the bottom /right end. You turn off each boolean if there will be another pipe butted up against that side. The following diagram illustrates the naming conventions:  CAP BOTTOM TOP END 1 END 2 BOTTOM TOP CAP TOP CAP BOTTOM END 1 TOP CAP BOTTOM END 2

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Center Fill	The center of the fill gradient. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.mainColor	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Edge Fill	The edge of the fill gradient. See Color Selector.	Color	.secondaryColor	Appearance
End 1 Bottom?	Draw the border at end #1's bottom?	boolean	.end1Bottom	Appearance
End 1 Cap?	Draw the border at end #1's cap?	boolean	.end1Cap	Appearance
End 1 Top?	Draw the border at end #1's top?	boolean	.end1Top	Appearance
End 2 Bottom?	Draw the border at end #2's bottom?	boolean	.end2Bottom	Appearance
End 2 Cap?	Draw the border at end #2's cap?	boolean	.end2Cap	Appearance
End 2 Top?	Draw the border at end #2's top?	boolean	.end2Top	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Outline Color	The color of the outline border. See Color Selector.	Color	.outlineColor	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

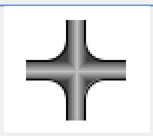
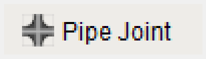
Customizers

- Vision Component Customizers
- Style Customizer

Examples

There are no examples associated with this component.

Vision - Pipe Joint

General

Component Palette Icon:


Description
The pipe joint displays a fancy joint component two join two pipe segments together. By turning off the cardinal directions, this will display a two-, three-, or four-pipe union. This component is optional, as pipes can butt up against each other and look joined.

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Bottom?	Indicates if the joint has an outlet at the bottom.	boolean	.bottom	Appearance
Center Fill	The center of the fill gradient. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.mainColor	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Edge Fill	The edge of the fill gradient. See Color Selector.	Color	.secondaryColor	Appearance
Left?	Indicates if the koint has an outlet at the left.	boolean	.left	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component. See Color Selector.	String	.name	Common
Outline Color	The color of the outline border. See Color Selector.	Color	.outlineColor	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Right?	Indicates if the joint has an outlet at the right.	boolean	.right	Appearance
Styles	Contains the component's styles	Dataset	.styles	Appearance
Top?	Indicated if that joint has an outlet at the top.	boolean	.top	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

⚠ This event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
. clickCo unt	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
. popupT rigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
. altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
. control Down	True (1) if the Control key was held down during this event, false (0) otherwise.
. shiftDo wn	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
. clickCo unt	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
. popupT rigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
. altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
. control Down	True (1) if the Control key was held down during this event, false (0) otherwise.
. shiftDo wn	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Always filter out these events for the property that you are looking for. Components often have many properties that change.

Customizers

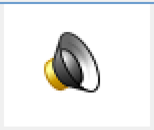
- Vision Component Customizers
- Style Customizer

Examples

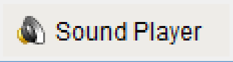
There are no examples associated with this component.

Vision - Sound Player

General



Component Palette Icon:



Description

The Sound Player component is an invisible component that facilitates audio playback in the client. Each Sound Player component has one sound clip associated with it, and will play that clip on demand. There is a built in triggering system, as well as facilities to loop the sound while the trigger is set. The sound clip needs to be a *.wav file. The clip becomes embedded within the window that the sound player is on. Clients do not need access to a shared *.wav file.

Properties

Name	Description	Property Type	Scripting	Category
Loop Count	If Loop Mode is "Loop N Times", this is the "N".	int	.loopCount	Behavior
Loop Mode	The Loop Mode determines how many times the sound is played when triggered.	int	.loopMode	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Mute	If true, the clip will be muted during playback.	boolean	.mute	Behavior
Name	The name of this component.	String	.name	Common
Play Mode	The Play Mode determines whether the sound is played automatically on trigger or manually.	int	.playMode	Behavior
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Sound Data	The clip that this component will play.	byte[]	.soundData	Data
Trigger	The clip will be played when the trigger is true, if Play Mode is "ON_TRIGGER"	boolean	.trigger	Data
Volume	The volume to use for playback (from 0.0 to 1.0).	double	.volume	Behavior
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component.

This event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers

Examples

There are no examples associated with this component.

Vision - Timer

General

Component Palette Icon:

Timer

Description

The timer button is an invisible button that can be used to create repeated events in a window. This is often used for animations or repetitive scripts within a window. When running, the timer's Value property is incremented by the Step By value, until the value tis the Bound, at which point it repeats. It is often useful to bind other values to a timer's Value property.

For instance, if you set the timer's Bound property to 360, and bind an object's rotation to the Value property, the object will spin in a circle when the timer is running.

How fast the timer counts is up to the Delay property, which is the time between counts in milliseconds.

Want to run a script every time the timer counts? First, make sure you don't actually want to write a project Timer Script, which will run on some interval whenever the application is running. In contrast, a script that works via a Timer component will only run while the window that contains the Timer is open, and the Timer is running. The way to do this is to attach an event script to the actionPerformed event.

Properties

Name	Description	Property Type	Scripting	Category
Bound	The value is always guaranteed to be less than this upper bound.	int	.max	Data
Delay (ms)	The delay in milliseconds between timer events.	int	.delay	Behavior
Initial Delay (ms)	The delay in milliseconds before the first event when running is set to true.	int	.initialDelay	Behavior
Name	The name of this component.	String	.name	Common
Running?	Determines whether or not the timer sends timer events.	boolean	.running	Behavior
Step by	The amount added to the value each time this timer fires for use as a counter. (should be positive)	int	.step	Data
Value	The current value of this timer, for use as a counter. At each iteration, this value will be set to ((value + step) MOD bound)	int	.value	Data

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

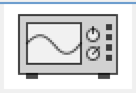
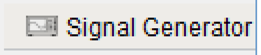
This component does not have any custom properties.

Examples

Expression Binding Example

```
//Suppose that you have images that make up frames of animation.
//Name your images: "Frame0.png", "Frame1.png", "Frame2.png". Set the timer's Bound to be 3,
then bind the image path of animate component to the following expression:
"Frame" + {Root Container.Timer.value} + ".png"
```

Vision - Signal Generator

General

Component Palette Icon:


Description
The signal generator is similar to the Timer component, but its value isn't simply a counter. Instead, you can choose from a variety of familiar signals. You configure the frequency by setting the Periodproperty, which is in milliseconds. You configure the resolution by setting the ValuesPerPeriod property. For example, if you choose a sine wave signal with a period of 2000 milliseconds and 10 valuesPerPeriod, your sine wave will have a frequency of 0.5 Hz, and its value will change 10 times every 2 seconds.

Properties

Name	Description	Property Type	Scripting	Category
Lower Bound	The lower bound of the signal value.	double	.lower	Data
Name	The name of this component.	String	.name	Common
Period	The period of the signal in milliseconds.	int	.period	Behavior
Running?	Determines whether or not the signal is being generated.	boolean	.running	Behavior
Signal Type	The signal type (shape) of the signal value.	int	.signalType	Behavior
Upper Bound	The upper bound of the signal value.	double	.upper	Data
Value	The current value of this signal generator.	double	.value	Data
Values/Period	The number of value changes per period.	int	.valuesPerPeriod	Behavior

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
. property Name	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

This component does not have any custom properties.

Examples

This component does not have any examples associated with it.

Vision - Reporting Palette

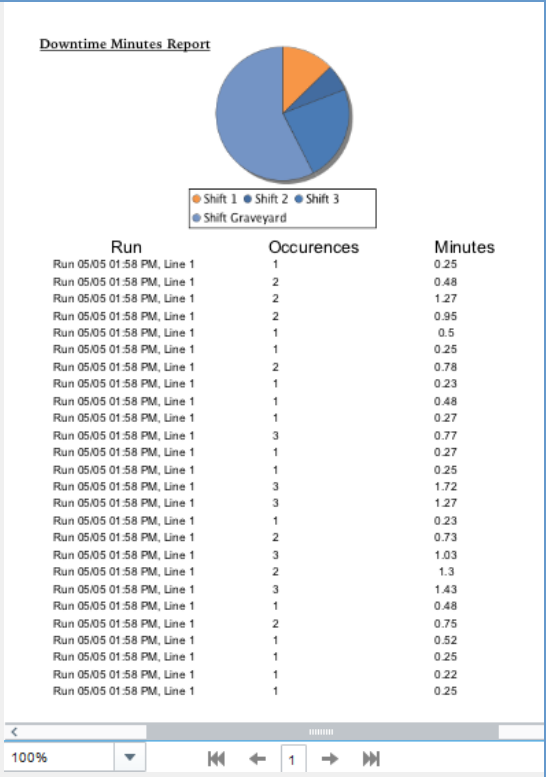
Reporting Components

The following components require the Report Module, and give you access to generated reports and various ways to filter and display data.

In This Section ...

Vision - Report Viewer

General



Run	Occurences	Minutes
Run 05/05 01:58 PM, Line 1	1	0.25
Run 05/05 01:58 PM, Line 1	2	0.48
Run 05/05 01:58 PM, Line 1	2	1.27
Run 05/05 01:58 PM, Line 1	2	0.95
Run 05/05 01:58 PM, Line 1	1	0.5
Run 05/05 01:58 PM, Line 1	1	0.25
Run 05/05 01:58 PM, Line 1	2	0.78
Run 05/05 01:58 PM, Line 1	1	0.23
Run 05/05 01:58 PM, Line 1	1	0.48
Run 05/05 01:58 PM, Line 1	1	0.27
Run 05/05 01:58 PM, Line 1	3	0.77
Run 05/05 01:58 PM, Line 1	1	0.27
Run 05/05 01:58 PM, Line 1	1	0.25
Run 05/05 01:58 PM, Line 1	3	1.72
Run 05/05 01:58 PM, Line 1	3	1.27
Run 05/05 01:58 PM, Line 1	1	0.23
Run 05/05 01:58 PM, Line 1	2	0.73
Run 05/05 01:58 PM, Line 1	3	1.03
Run 05/05 01:58 PM, Line 1	2	1.3
Run 05/05 01:58 PM, Line 1	3	1.43
Run 05/05 01:58 PM, Line 1	1	0.48
Run 05/05 01:58 PM, Line 1	2	0.75
Run 05/05 01:58 PM, Line 1	1	0.52
Run 05/05 01:58 PM, Line 1	1	0.25
Run 05/05 01:58 PM, Line 1	1	0.22
Run 05/05 01:58 PM, Line 1	1	0.25

Component Palette Icon:

Report Viewer

Description

The Report Viewer component provides a way to run and view Reports in Vision windows. Parameters added during Report creation are provided as Properties in the Viewer and can override any default values set in the Report Resource. Right clicking on the Report Viewer brings up a menu that allows you to easily print the report or save it in various formats. The Reporting Module comes with some additional reporting components that can be used with Vision components. To learn more, refer to the Vision Reporting Components section in the Reporting Module. To find documentation on the deprecated Report Viewer prior to Ignition 7.8, see the Legacy Report Viewer documentation.

Properties

Name	Description	Type	Scrip
Background Color	Color that lays underneath the report. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	. backg nd
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. The border is unaffected by rotation.	Border	.bord
Current Page	Current page in the report you would like to view.	Int	. curre ge
EndDate	End date for the report.	Date	.EndI
Fit Panel	The following feature is new in Ignition version **8.0.3** Click here to check out the other new features Ignore the zoom and fit the report to the component.	Boolean	.fitPa
Foreground Color	The foreground color the labels on the component. See Color Selector.	Color	. foreg
Name	The name of this component.	String	.nam
Page Count	Number of pages in the report.	Int	. page nt
Report Loading	Returns true while the report is loading, Note that this property does **NOT** appear in the Property Editor, but can easily be accessed from a Python script. Useful in scenarios where you wish to change the value of a parameter on the Report Viewer in a script and then do some additional work once the report has finished loading.	Boolean	repor ding
Report Path	Path in the Project to the Report you would like to view.	String	. repor
Show Controls	Show the bar with the page and the zoom controls.	Boolean	. show trols
Start Date	Start date for the report.	Date	. StartI
Suggested Filename	The filename that will come up by default when the user saves the report to disk.	String	. sugg dFile
Visible	If disabled, the component will be hidden.	Boolean	.visib
Zoom Factor	Zoom factor for this report.	Float	. zoom tor

Scripting

Scripting Functions

The following print method will only work if a report has finished loading on the Report Viewer component

- Description

  Uses the named printer and determine if the print dialog window should appear or not.

- Parameters

  String printerName - The name of the printer the report should be sent to. Will use the default printer if left blank. [optional]

  Boolean showDialog - True if the dialog window should appear, False if the dialog window should be skipped. Will be true if left blank. [optional]

- Return

  Nothing

- Description

  Return the bytes of the generated report in the Report Viewer using PDF format.

- Parameters

  None

- Return

  Byte Array - The bytes of the report in PDF format.

This function will return null if the trial has expired.

- Description

  Return the bytes of the generated report in the Report Viewer using PNG format.

- Parameters

  Nothing

- Return

  Byte Array - The bytes of the report in PNG format.

This function will return null if the trial has expired.

- Description

  Prompts the user to save a copy of the report as a PDF. Shows a file selection window with the extension set to PDF.

- Parameters

  String fileName - A suggested filename to save the report as

- Return

  Nothing

- Description

    Prompts the user to save a copy of the report as a PNG. Shows a file selection window with the extension set to PNG.

- Keyword Args

    String fileName - A suggested filename to save the report as.

- Return

    Nothing

- Description

    Prompts the user to save a copy of the report as an XLS file. Shows a file selection window with the extension set to XLS.

- Keyword Args

    String fileName - A suggested filename to save the report as.

- Return

    Nothing

Extension Functions

- Description

    Called when the Report generation process has been completed.

- Keyword Args

    Component self - A reference to the component invoking this method.

    Byte Array pdfBytes - The PDF formatted bytes generated by the Report.

- Return

    Nothing

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Examples

print()

```
#calls print on a Report Viewer component located in the same window

reportViewer = event.source.parent.getComponent('Report Viewer')
reportViewer.print()
```

print() with default printer, no dialog

```
#calls print on a Report Viewer component located in the same window
#bypasses the print dialog window and uses the default printer

reportViewer = event.source.parent.getComponent('Report Viewer')
reportViewer.print(None, False)
```

saveAsPDF()

```
#Saves the file as a PDF to a user selected location.
#The file selection window defaults to a name of "Daily Report"

reportViewer = event.source.parent.getComponent('Report Viewer')
reportViewer.saveAsPDF("Daily Report")
```

Utilizing reportLoading

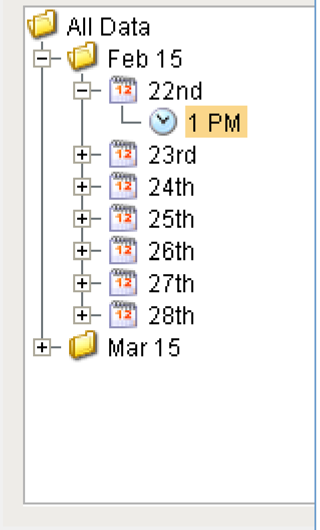
```
#This example will check if the report has finished loading. If so, print the
report.

#Reference the report viewer
reportViewer = event.source.parent.getComponent('Report Viewer')

if reportViewer.reportLoading:
        system.gui.warningBox("The report is still loading. Please wait")
else:
        reportViewer.print()
```

Customizers

This component does not have any custom properties.

Vision - Row Selector

General



Component Palette Icon:

Row Selector

INDUCTIVE UNIVERSITY

Row Selector

Watch the Video

Description

The row selector is a component that acts like a visual filter for datasets. It takes one dataset, chops it up into various ranges based on its configuration, and lets the user choose the splices. Then it creates a virtual dataset that only contains the rows that match the selected splices.

The most common way to splice the data is time. You could feed the row selector an input dataset that represents a large time range, and have it break it up by Month, Day, and then Shift, for example. Then you could power a report with the output dataset, and that would let the user dynamically create reports for any time range via an intuitive interface.

To configure the row selector, first set up the appropriate bindings for its input dataset. Then use its Customizer to alter the levels that it uses to break up the data. In the customizer, add various filters that act upon columns in the input dataset, sorting them by various criteria. For example, you could choose a date column, and have it break that up by quarter. Then below that, you could have it use a discrete filter on a product code. This would let the user choose quarterly results for each product. Each level of filter you create in the customizer becomes a level in the selection hierarchy. Note that the output data is completely unchanged other than the fact that rows that don't match the current user selection aren't present.

This component is very handy for driving the Report Viewer, Table, and Classic Chart components, among others.

Additional information on the Row Selector can be found on the Reporting in Vision page.

Properties

Name	Description	Property Type	Scripting	Category

All Data Node Text	Text for the All Data node, if it is displayed.	String	.allDataNodeText	Appearance
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	Cursor	.cursor	Common
Data In	The input of the row selection tree. The filter tree changes based on this Dataset.	Dataset	.dataIn	Data
Data Out	The output of the row selection tree. Changes based on user selection in the filter tree.	Dataset	.dataOut	Data
Expand All Data Node	If true, the 'All Data' (root) node will be expanded and selected when the user opens this window.	boolean	.expandAllDataNode	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See Color Selector.	Color	.foreground	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.tooltiptext	Common
Name	The name of this component.	String	.name	Common
Opaque	If false, backgrounds are not drawn. If true, backgrounds are drawn.	boolean	.opaque	Common
Selection Background	The background color of the selected node. See Color Selector.	Color	.selectionBackground	Appearance
Show All Data Node	Should the All Data (root) node be shown or hidden?	boolean	.showAllDataNode	Behavior
Show Node Size	If true, the number of rows in each node will be shown.	boolean	.showNodeSize	Behavior
Show Root Handles	Should root-level nodes have collapse handles?	boolean	.showRootHandles	Behavior
Unknown Node Icon	Icon for any Unknown nodes (nodes where the data didn't match the filter).	String	.unknownIconPath	Appearance
Unknown Node Text	Text for any Unknown nodes (nodes where the data didn't match the filter).	String	.unknownNodeText	Appearance
Visible	If disabled, the component will be hidden.	Boolean	.visible	Common

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

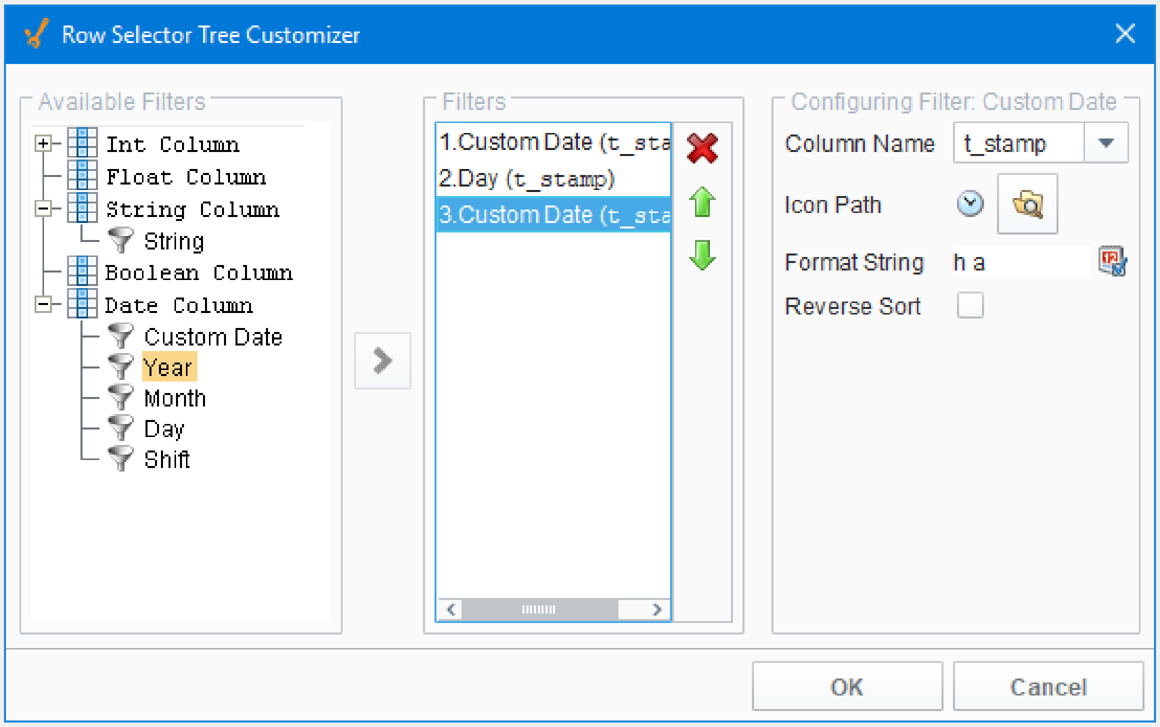
Customizers

The Row Selector has its own Row Selector Tree Customizer and allows users to customize the row filtering. The customizer provides some default filters which you can use, or customized based on the dataset.

The Row Selector Tree Customizer allows you to build and configure a tree of the data in the input dataset which can then be used to filter it. There are three main parts to the customizer. The left panel contains a list of available filters, the center panel contains a list of filters that will be used, and the right panel will contain configurable properties for the filter currently selected in the center panel.

In the Available Filters section on the left, a list of all of the columns of the dataset are shown. These can be expanded to show the filters available for that column type. Some columns might not have any filters, while others can have many, it just depends on the data type of column. These filters can then be dragged into the center panel, or highlighted and the Right Arrow icon pressed to push the filter into the center panel where it becomes an active filter.

The Filters panel in the center contains a list of filters that are being used with each filter being followed by the name of the column that it originated from, and is where you can decide on the order of the filters. The order is important because it is the order in which they will be used in the component. Using the image below as an example, The component will first show a list of years. You can select a particular year, and the output dataset will only contain rows from that year. Alternately, you can expand a year where you will then see a list of strings that are in rows with that year. Selecting one of the strings will display all rows with strings like the one that you selected, that are also in the same year.



1.Custom Date (t_sta
2.Day (t_stamp)
3.Custom Date (t_sta

Column Name: t_stamp
Icon Path
Format String: h a
Reverse Sort: ☐

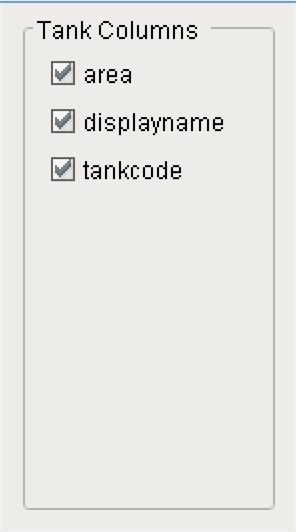
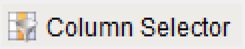
The Configure Filter panel on the right contains configurable settings that differ based on the type of filter selected. All filters at least contain an Icon Path property, which allows you to set what icon will be used with with that filter in the filter tree. Each filter type also has a reverse sort option, allowing you to have the filters displayed in reverse order in the filter tree. The unique properties are:

- Column Name
- Icon Path
- Format String (if applicable)
- Reverse Sort

Examples

There are no examples associated with this component. Refer to the examples in the Common Reporting Tasks.

Vision - Column Selector

General

Component Palette Icon:


INDUCTIVE UNIVERSITY

Column Selector

Watch the Video

Description
The column selector component is conceptually similar to the Row Selector, except that instead of filtering rows, it filters columns from its output dataset. Each column from the input dataset is shown as a checkbox. As the user checks and un-checks columns, the output dataset has those columns added or removed. This is very handy for driving the Table and Classic Chart components. In addition, this component can bring in multiple datasets and output just as many filtered datasets.   ⓘ Addition information on the Column Selector can be found on the Vision Reporting Components page.

Properties

Name	Description	Property Type	Scripting	Category
Alphabetize	If true, checkboxes will be ordered alphabetically by their text.	Boolean	.alphabetize	Behavior
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	Int	.cursorCode	Common
Data In	Input dataset. This is the default when first dropping the component on the window, the name may change based on configuration and there may be more of these input dataset properties.	Dataset	.Data_in	Custom Properties
Data Out	Output dataset. This is the default when first dropping the component on the window, the name may change based on configuration and there may be more of these output dataset properties.	Dataset	.Data_out	Custom Properties
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See Color Selector.	Color	.foreground	Appearance
Group By Dataset	If true, checkboxes will be grouped by their dataset. Otherwise, checkboxes will be arranged flat.	Boolean	.grouping	Behavior
Horizontal Gap	The horizontal gap between checkboxes or grouping panels.	Int	.hGap	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Normalize Widths	If true, all checkboxes will be assigned the same width, which causes them to line up in columns.	Boolean	.normalizeWidths	Appearance
Vertical Gap	The vertical gap between checkboxes and grouping panels.	Int	.vGap	Appearance
Visible	If disabled, the component will be hidden.	Boolean	.visible	Common

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

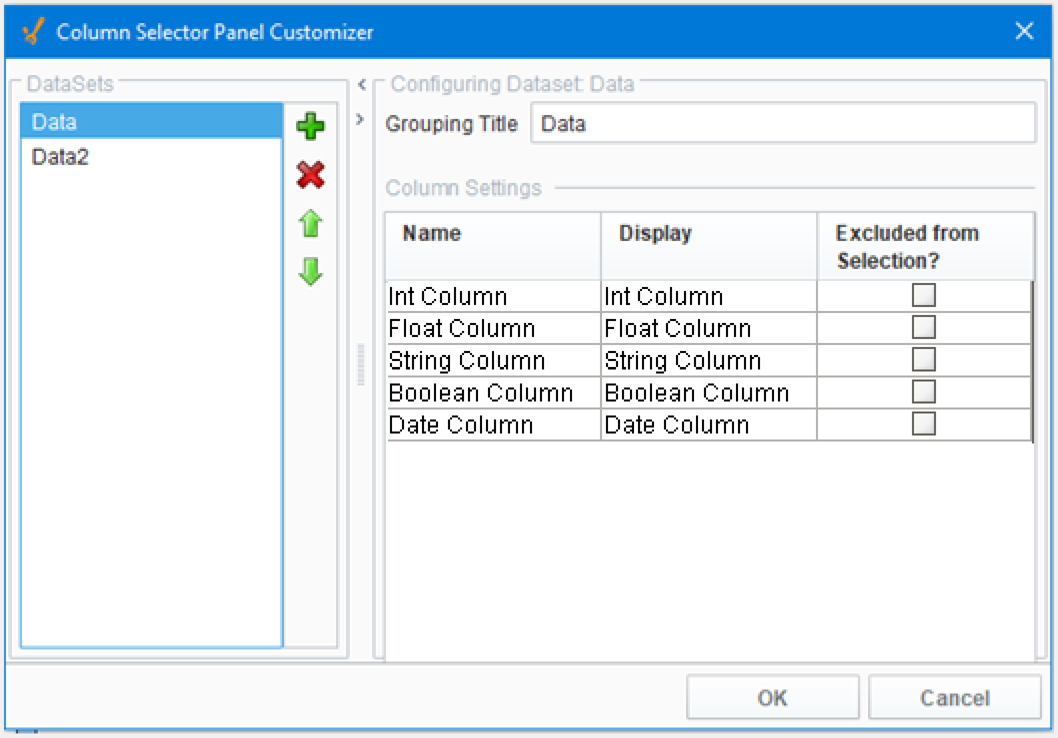
.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event.
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
. property Name	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers
The Column Selector component has its own Column Selector Panel Customizer that allows you to configure how the Column Selector filters columns.

The Column Selector Customizer contains two basic parts. The left side of the customizer allows you to configure how many datasets can be brought in for filtering. Each dataset added will add two additional custom properties to the Column Selector; an In dataset property and an Out filtered dataset property. Datasets can also be removed here, or moved up or down in the list. If there are multiple datasets, the columns from the first dataset in the list will be displayed at the top of the Column Selector, while the columns from the last will be at the bottom.

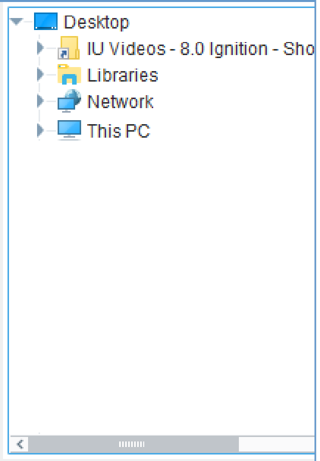
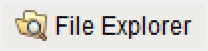


Name	Display	Excluded from Selection?
Int Column	Int Column	☐
Float Column	Float Column	☐
String Column	String Column	☐
Boolean Column	Boolean Column	☐
Date Column	Date Column	☐

The right side of the customizer allows you to configure the settings for each dataset. When a dataset is highlighted on the left, we can see some basic information about it on the right, such as the Grouping Title and a list of all of the columns in that dataset. The Grouping Title is only used if there is more than one dataset in the Column Selector. In the component, each dataset's columns will be contained in a border and will display the Grouping Title. This can be configured to be anything, so that it is easier for a user to distinguish what each set of columns is for. In the Column Settings table, we see each one of the columns in that dataset listed out. Here, the Display column allows us to alter what name that column will display on the component, again allowing you to create names that are more meaningful to the user. Finally, the Excluded from Selection column allows you to exclude certain columns from being filtered. Columns that have this enabled will not show up in the list of columns on the component. This will not filter them out in the output dataset, but instead forces them to be in the output dataset.

Examples
Refer to the example on the Vision Reporting Components page.

Vision - File Explorer

General
 Desktop IU Videos - 8.0 Ignition - Sho Libraries Network This PC
Component Palette Icon:


INDUCTIVE UNIVERSITY

File Explorer and PDF Viewer

Watch the Video

Description
The File Explorer component displays a filesystem tree to the user. It can be rooted at any folder, even network folders. It can also filter the types of files that are displayed by their file extension (i.e., "pdf"). The path to the file that the user selects in the tree is exposed in the bindable property Selected Path. The File Explorer component is typically used in conjunction with the PDF Viewer component in order to create a PDF viewing window. This is very useful for viewing manuals, documents, or reports from within your project. To use this component to drive a PDF Viewer component, refer to the section on File Explorer and PDF Viewer.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Enabled	If disabled, the component can't be used.	boolean	.componentEnabled	Common
File extension filter	Semi-colon seperated list of extensions to filter out files, such as pdf or txt. Example "pdf;html;txt" shows pdf, html, and text documents.	String	.fileFilter	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See Color Selector.	Color	.foreground	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Root Directory	A directory to act as the root of the file explorer.	String	.rootDir	Behavior
Selected Path	The selected file or folder's path.	String	.selectedPath	Data
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

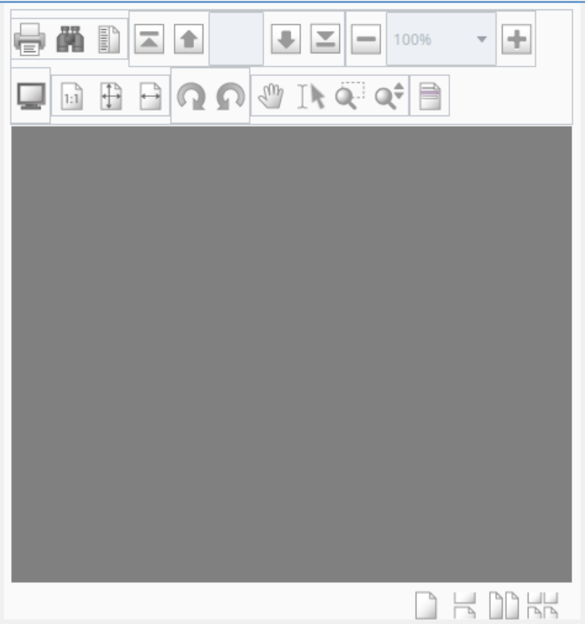
Customizers

The File Explorer component does not have a customizer.

Examples

Refer to the examples on the File Explorer and PDF Viewer pages.

Vision - PDF Viewer

File Explorer and PDF Viewer
 **Component Palette Icon:**

INDUCTIVE UNIVERSITY

File Explorer and PDF Viewer

Watch the Video

Description
The PDF Viewer component displays a PDF that exists as a file in some accessible file system, or as a URL. Note that this component is simply for viewing existing PDFs. To create dynamic reports, or view dynamically generated reports use the Reporting Module. This component is typically used in conjunction with the File Explorer component, in order to create a PDF viewing window. Simply bind the Selected Path property in the PDF Viewer to the File Explorer's *Selected Path* property. See the File Explorer's documentation, as well as the File Explorer and PDF Viewer pages for further instructions on how to put these two components together.

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. ⚠ The border is unaffected by rotation.	Border	.border	Common
File Path	Path to the .pdf file to be displayed.	String	.filePath	Data
Footer Visible	If false, the Footer is not displayed.	Boolean	.footerVisible	Appearance
Name	The name of this component.	String	.name	Common
Page Fit Mode	Mode to fit the document within the viewer. (1 = Disabled, 2 = Actual Size, 3 = Fit Height, 4 = Fit Width)	Integer	.pageFitMode	Appearance
Page View Mode	How to display PDF in Viewer (1 = One Page, 2 = One Column, 3 = Two Page Left, 4 = Two Col Left, 5 = Two Page Right, 6 = Two Col Right)	Integer	.pageViewMode	Appearance
Toolbar Visible	Sets the top PDF control toolbar to visible.	Boolean	.toolBarVisible	Appearance
Utility Visible	Sets the Utility Sidebar to visible.	Boolean	.utilityPaneVisible	Appearance
Visible	If disabled, the component will be hidden.	Boolean	.visible	Common

PDF Viewer Toolbar

Toolbar Buttons	Name	Function
💾	Save As	Will save the currently loaded pdf to the local computer. This feature was changed in Ignition version **8.0.11**: This button was removed, since the Save As functionality was never intended to be included on the component.
🖨	Print Document	Will print the currently loaded pdf from the local computer.
🔍	Search Document	Will open up a text field that can be used to search the currently loaded pdf for a specific word or phras *Note: This is located in the Utility Panel and can be accessed from there as well.

📄	Show /Hide Utility Panel	Will show/hide the Utility panel. The Utility Panel contains the following tabs: • Search - Will search the document for a specific word or phrase. • Bookmarks - Will display all of the bookmarks for the loaded pdf and allow you to quickly jump to them. • Thumbnails - Will display a thumbnail view of all of the pages of the loaded pdf. Clicking on one w jump to it. • Annotations - Will create a multitude of annotations on the currently loaded pdf. After adding an annotation, it can be selected and then configured in the Utility Panel. Annotations include highlights, strike through, underlines, text notes, and actions like navigating to a url. • Layers - Will display the layers of the currently loaded pdf, if any.
⤒	First Page	Will navigate back to the first page of the pdf.
⬆	Previous Page	Will navigate back one page of the pdf.
1 of 7	Current Page Number	Will show the current page number out of the total number of pages, also allowing a page number to b entered which will jump to that page immediately.
⬇	Next Page	Will navigate forward one page of the pdf.
⤓	Last Page	Will navigate forward to the last page of the pdf.
➖	Zoom Out	Will zoom out from the pdf.
100% ▾	Zoom	A drop down list that displays the current zoom, as well as giving the ability to switch between different preset zoom amounts.
➕	Zoom In	Will zoom in to the pdf.
1:1	Actual Size	Will revert back to a 100% zoom which is the natural size of the pdf.
✥	Fit In Window	Will fit the pdf to the pdf viewer window.
↔	Fit Width	Will fit the pdf to the width of the pdf viewer.
↻	Rotate Right	Will rotate the pdf right.
↺	Rotate Left	Will rotate the pdf left.
✋	Pan Tool	Will pan around a page of the pdf by clicking and dragging. Works better when zoomed in.
I	Text Select Tool	Can be used to select text in the pdf.
🔍	Zoom Marquee Tool	Will zoom into the pdf by clicking and dragging to select an area.

	Zoom Dynamic Tool	Will zoom in and out using the scroll wheel.
	Select Tool	Can be used to select objects on the pdf such as annotations.
	Highlight Annotation Tool	Can be used to highlight text in the pdf. Can also be done from the Utility Panel and can be configured there as well.
	Text Annotation Tool	Can be used to place a text comment on the pdf. Can be configured in the Utility Panel.
	Show/Hide Form Highlighting	Show or hide highlighting on the form.
	Single Page View Non-Continuous	View the pdf file one page at a time.
	Single Page View Continuous	View the pdf file one page wide with continuous scrolling.
	Facing Page View Non-Continuous	View the pdf file two pages at a time.
	Facing Page View Continuous	View the pdf file two pages wide with continuous scrolling.

Scripting

Scripting Functions

- Description

  This function will pass in the bytes of a PDF and load them into the PDF Viewer component. Please see Storing Files in a Database for more details

- Parameters

  string bytes - The bytes of the PDF to be displayed on the component

  string name - The name of the PDF

- Return

  Nothing

- Since 7.8.2

- Description

  This function will print the PDF.

- Parameters

  boolean showDialog- If true, shows the user a print dialog. Default is true [optional]

- Return

  Nothing

- Since 7.8.2

- Description

  This function will set the current zoom level of the PDF, adjusted to stay within the minimum / maximum zoom range. Will zoom in on center of page.

- Parameters

  float zoom- Zoom factor to use. 1.0 is no zoom.

- Return

  Nothing

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

This event signifies a mouse click on the source component. A mouse click the combination of a mouse press and a mouse release, both of which must have occurred over the source component. Note that this event fires after the pressed and released events have fired.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse enters the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when the mouse leaves the space over the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is pressed down on the source component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

This event fires when a mouse button is released, if that mouse button's press happened over this component.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component after a button has been pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires when the mouse moves over a component, but no buttons are pushed.

.source	The component that fired this event.
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.
.x	The x-coordinate (with respect to the source component) of this mouse event.
.y	The y-coordinate (with respect to the source component) of this mouse event.
.popupTrigger	Returns True (1) if this mouse event is a popup trigger. What constitutes a popup trigger is operating system dependent, which is why this abstraction exists.
.altDown	True (1) if the Alt key was held down during this event, false (0) otherwise.
.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event.
. newValue	The new value that this property changed to.
. oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
. property Name	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

The PDF Viewer component does not have a special customizer, however, it does use the Style Customizer and Custom Properties.

- Vision Component Customizers

Examples

Refer to the examples on the File Explorer and PDF Viewer pages.

Vision - Web Browser Palette

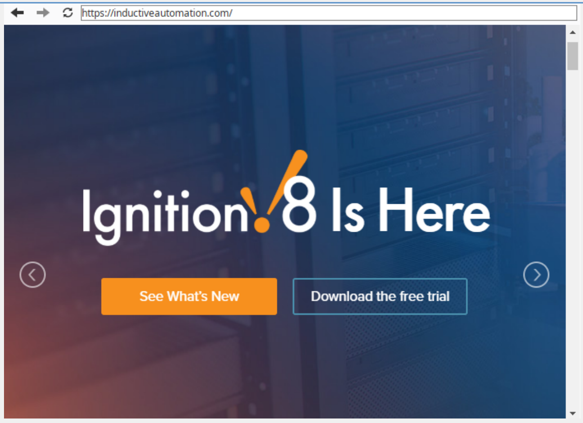
Web Browser Components

The following component gives you the ability to add a web browser to your client.

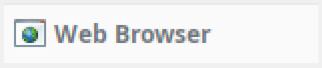
In This Section ...

Vision - Web Browser Component

General



Component Palette Icon:



INDUCTIVE UNIVERSITY

Web Browser

Watch the Video

Description

The **Web Browser** component in Designer allows you to embed a full web browser inside of an Ignition Client. This component becomes available in Designer after you download the Web Browser module from the Inductive Automation's website. The Web Browser module installs the same way as any other modules. Once this component is added onto a window, it will behave just like any other web browser when it is inside a Client.

Client machines need to meet the following minimum requirements to use this component. The component may not work properly if the requirements are not met.

Operating System Requirements

Windows

- Microsoft Windows XP (SP2), 7, 8, Vista, Server 2003 (SP1), Server 2008/2012, 32-bit and 64-bit.
    - Windows version 8 and 8.1 require Java 6 update 38 or greater
- Oracle (Sun) JRE 1.6.x and higher, 32-bit and 64-bit.

Linux

- Ubuntu 12.04+, Debian 7.7, RedHat Enterprise Linux 7, openSUSE 13.1, Fedora 20, 32-bit and 64-bit
- Oracle (Sun) JRE 1.6.x and higher, 32-bit and 64-bit.

Required Linux Libraries

Missing Libraries: 32-bit Ubuntu 12.04

Some 32-bit Linux distros are missing a needed library for running the Web Browser: libXss.so.
1

Steps that fixed it in Ubuntu 12.04:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install ia32-libs-multiarch
```

Missing Libraries: Ubuntu 17.04

Ubuntu 17.04 is missing a library that is required for the component to run. Running the following command can resolve the issue:

```
sudo apt-get install libgconf-2-4
```

Mac OS X

- Mac OS X 10.7.x - 10.10.x (Intel)
- Apple or Oracle (Sun) JRE 1.6.x and higher, 32-bit and 64-bit.

Windows

- Microsoft Windows 7, 8, 8.1, 10, Server 2008 R2, Server 2012, Server 2016, 32-bit and 64-bit.
    - Windows version 8 and 8.1 require Java 6 update 38 or greater
- Oracle (Sun) JRE 1.6.x and higher or IBM JRE 1.7.x and higher, 32-bit and 64-bit.

Linux

- Ubuntu 14.04+, 17.04 Desktop, Debian 8+, RedHat Enterprise Linux 7, openSUSE 13.3+, Fedora 24+, 64-bit only
- Oracle (Sun) JRE 1.6.x and higher or IBM JRE 1.7.x and higher, 32-bit and 64-bit.

Required Linux Libraries

Missing Libraries: Ubuntu 17.04

Ubuntu 17.04 is missing a library that is required for the component to run. Running the following command can resolve the issue:

```
sudo apt-get install libgconf-2-4
```

Mac OS X

- Mac OS X 10.9.x - 10.13.x (Intel)
- Apple or Oracle (Sun) JRE 1.6.x and higher, 32-bit and 64-bit.

The Web Browser Component will only support the following audio and video codecs: Opus, Theora, Vorbis, VP8, VP9, and WAV.

The underlying browser component is available in scripting through the getBrowser() method. Documentation on the browser component is available at the JxBrowser Programmer's Guide. The Inductive Automation support team is unable to provide detailed advice on scripting with this component. Furthermore, they are unable to provide troubleshooting beyond the basic functionality of the module.

Hardware Notes

ARM

Currently, the jxBrowser does not support the ARM architecture, thus the component will not work properly when used in conjunction with the ARM architecture.

Properties

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. ⚠ The border is unaffected by rotation.	Border	.border	Common
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
FTP Proxy Port	FTP Proxy Port sets the proxy port for FTP connections. This setting is only used when **Use Proxies** is checked.	int	.ftpProxyPort	Data
FTP Proxy Server	FTP Proxy Server sets the proxy server for FTP connections. This setting is only used when **Use Proxies** is checked. Can be empty	String	.ftpProxyServer	Data
HTTP Proxy Port	HTTP Proxy Port sets the proxy port for HTTP connections. This setting is only used when **Use Proxies** is checked.	int	.httpProxyPort	Data
HTTP Proxy Server	HTTP Proxy Server sets the proxy server for HTTP connections. This setting is only used when **Use Proxies** is checked. Can be empty	String	.httpProxyServer	Data
HTTPS Proxy Port	HTTPS Proxy Port sets the proxy port for HTTPS connections. This setting is only used when **Use Proxies** is checked.	int	.httpsProxyPort	Data
HTTPS Proxy Server	HTTPS Proxy Server sets the proxy server for HTTPS connections. This setting is only used when **Use Proxies** is checked. Can be empty	String	.httpsProxyServer	Data
Mode	Data source for browser. Mode controls whether **Starting URL** or **Starting HTML** will be used.	int	.mode	Data
Name	The name of this component.	String	.name	Common
Popups Allowed	This flag is used to allow popups in the web page displayed.	boolean	.popupsAllowed	Behavior
Proxy Exceptions	A comma delimited list of rules for websites that will bypass the proxy servers. An example sting would be "*foo.com,<local>, 127.0.1". This setting is only used when **Use Proxies** is checked.	String	.proxyExceptions	Data

Proxy Password	The password to use for proxy authentication. This setting is only used when **Use Proxies** and **Use Proxy Authentication** are checked.	String	.proxyPassword	Data
Proxy Username	The username to use for proxy authentication. This setting is only used when **Use Proxies** and **Use Proxy Authentication** are checked.	String	.proxyUsername	Data
SOCKS Proxy Port	The port number for SOCKS proxies.	int	.socksProxyPort	
SOCKS Proxy Server	The host name to use for SOCKS proxies. Can be empty.	String	.socksProxyServer	
Show Navigation Buttons	Show the navigation buttons at the top of the frame.	boolean	.showNavigation	Behavior
Starting HTML	The initial HTML displayed when the Mode is set to HTML. Starting HTML is `<html><body>&nbsp;</body></html>` by default, which gives a blank page.	String	.startingHtml	Data
Starting URL	The initial URL displayed when the Mode is set to URL. Starting URL is blank by default.	String	.startingUrl	Data
Touchscreen Mode	Controls when this input components responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
Use Proxies	If checked, the Web Browser will try to use the proxy settings.	boolean	.useProxies	Data
Use Proxy Authentication	If checked, the browser will use the username and password for proxy authentication. This setting is only used when Use Proxies is checked.	boolean	.useProxyAuthentication	Data
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Zoom Level	The zoom level the web page is displayed in. 0.0 is normal, positive numbers zoom in, negative numbers zoom out.	double	.zoomLevel	Behavior

Scripting

Scripting Functions

- Description

  This function will return the underlying browser object. See JxBrowser Programmer’s Guide for more information.

- Parameters

  none

- Return

  Object - The Browser Object

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. ⚠ Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

Customizers

- Vision Component Customizers

Examples

Setting Chromium Switches via JVM Arguments

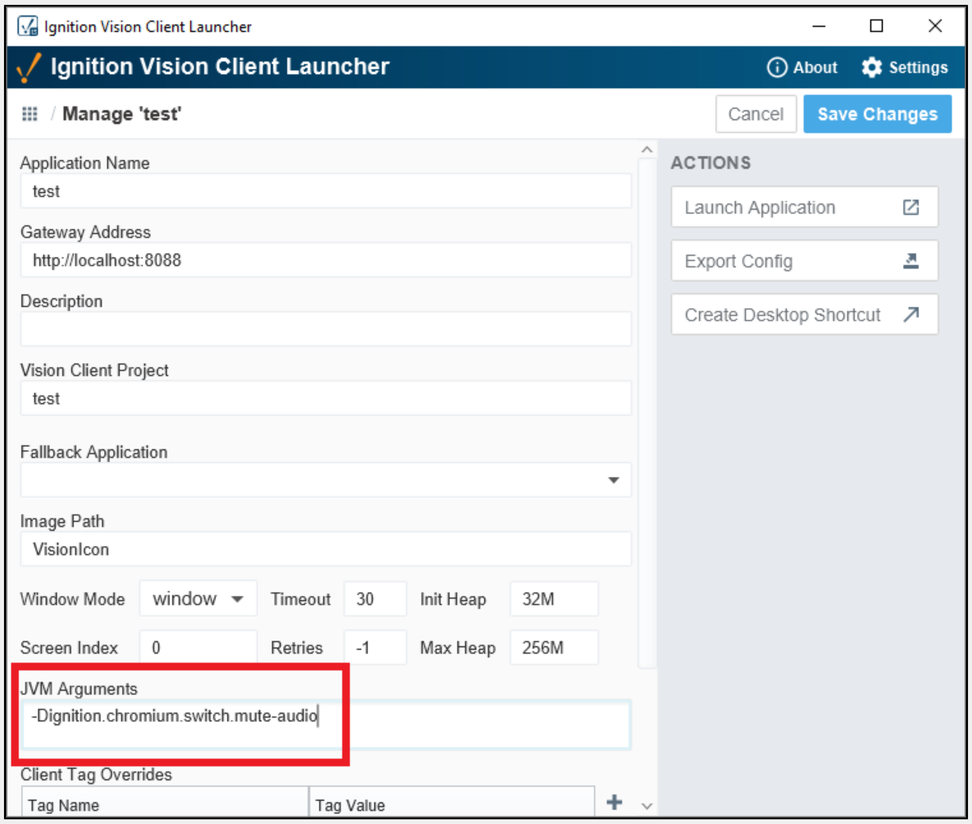
The Web Browser component is based off of the JxBrowser library, which in turn is based upon the Chromium engine. As a result, the Web Browser component can be further customized by manipulating Chromium Switches.

Caution: Implementing these switches is considered **unsupported** because they can drastically change the behavior of the Web Browser component. The exception to this case is when a member of our support team requests a switch be added to help troubleshoot an issue. For the sake of clarity, instructions on how to manipulate the switches via the Designer Launcher and Vision Client Launcher are listed below, but we generally do not recommend users implement these switches.

If you're going to make use of a switch, then you would do so on the Designer Launcher's/Vision Client Launcher application, under the JVM Arguments field. Below is an example on how to configure a switch for a client using the Vision Client Launcher. The same method applies for the Designer Launcher.

1. Open the Vision Client Launcher.
2. Once open, either create a new application or manage the settings on an existing application.
3. Once the Settings are open, add a new entry into the JVM Arguments text area. Arguments for Chromium Switches must have a prefix of "`-Dignition.chromium.switch.`" followed by the argument. Below is a example where we set the argument "`mute-audio`" :

```
-Dignition.chromium.switch.mute-audio
```

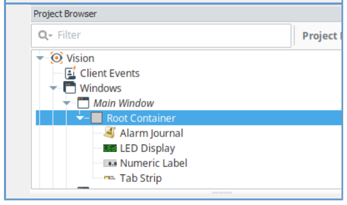


Application Name: test
Gateway Address: http://localhost:8088
Description:
Vision Client Project: test
Fallback Application:
Image Path: VisionIcon
Window Mode: window
Timeout: 30
Init Heap: 32M
Screen Index: 0
Retries: -1
Max Heap: 256M
JVM Arguments: -Dignition.chromium.switch.mute-audio

4. Following this change, audio from the Web Browser Component will be muted once the client is launched.

Vision - The Window Object

General



INDUCTIVE UNIVERSITY

Window Properties

Watch the Video

Description

Window

Windows are the top-level unit of design for Vision projects. A window is identified by its path, which is the name of all its parent folders plus its name, with forward slashes (/) in between. For example, the path to a window in the top level called MainWindow would simply be its name, whereas the path to a window named **UserOptions** under a folder called **OptionsWindows** would be: **OptionsWindows/UserOptions**.

A window may display a Titlebar and/or a Border. The titlebar allows the user to drag the window around in the client, and houses the window's close and maximize/restore buttons. The border of a window can be used to resize the window in the client when it is floating or docked. Whether on not the titlebar and border are displayed depends on the values of the window's titlebar and border display policy properties, and its current state. Commonly, a window will display both a titlebar and border when it is configured as a popup. It is often desirable to remove titlebars and borders on main windows so they join seamlessly with docked windows.

Note that the user manual describes different Window Types, technically there is only a single window object in the Vision module: different "types" of windows are simply instances of the window object configured in different ways. See Window Types for more information about changing types.

Root Container

Inside a window is always the Root Container. The Root Container is where you will place all of your components in the window. This is exactly the same as a normal container component except that it cannot be deleted. When in the designer, "resizing" the window from the main Vision workspace is really changing the size of the Root Container.

Window Opening Event Order

Window objects have several event handlers that trigger when the window opens. However, each event handler occurs at a separate time. Because of this, it is important to understand the order that these events occur:

Opening a window - When opening a window for the first time in a designer, the following event handlers are called in order:

1. visionWindowOpened - Important to note the description on this event: it occurs before any bindings on the window are evaluated.
2. internalFrameOpened- Again, the description notes that if the window has been cached, this will not fire on sequential opens.
3. internalFrameActivated - The last event, but also repeatable while the window is opened, since this event will trigger again if the window loses and then regains focus without being closed in between.

Closing a window - When closing a window, the following event handlers are called in order:

1. internalFrameClosing - This event would be ideal to "clean up" in the window, since the window is still technically open at this point.
2. visionWindowClosed - Triggers when the window is closed. Functionally, this is similar to internalFrameClosed, but happens slightly earlier.
3. internalFrameDeactivated - This triggers when the window is closed, or when the window loses focus, so you may want to avoid this event if your script should only trigger when the window is closed.
4. internalFrameClosed - Similar to visionWindowClosed. Triggers when the Java windowing system has finished closing the window.

Properties

Name	Description	Property Type	Scripting	Category
Border Display Policy	Determines if the window's border is shown in various window states. **Integer** | **Property** 0 | Always 1 | Never 2 | When Not Maximized	int	.borderDisplayPolicy	Behavior
Cache Policy	By default this property is set to **`Auto`**, which keeps a window in a memory cache for a while after it is closed, so that if it is opened again it will be quick. The window isn't "active" while it is closed: all of its bindings and scripts are shut down. Setting this property to **`Never`** causes a fresh copy of the window to be deserialized every time it is opened. This is a performance hit, but it also is a convenient way to "clear out" the values of the window from the last time it was opened, which can be helpful in data-entry screens. Setting the property to **`Always`** will trade memory for higher performance, causing the window to always remain cached after the first time it is opened. This means the window will open very fast, but your Client will need lots of memory if you do this to a large amount of windows. **Integer** | **Property** 0 | Auto 1 | Never 2 | Always	int	.cachePolicy	Behavior
Closeable	Determines whether or not to draw the close (X) button in the upper right corner.	boolean	.closable	Behavior
Dock Index	Determines the order of docked windows if multiple windows are open on the same edge. Lower numbers are on the outside (closest to the edge the window is docked to), and higher numbers are closer to the center.	int	.dockIndex	Layout
Dock Position	Determines the position this window is docked to, or if it is floating. **Integer** | **Property** 0 | Floating 3 | West 4 | South 2 | East 1 | North	int	.dockPosition	Layout
Layer	Sets the layer that this window is in. Default layer is 0, which is the bottom layer. Windows in higher layers will always be shown on top of windows in layers beneath them. A common strategy for using the layer property is to set Main Windows and Docked windows to 0, Popups to 1 and very important popups to 2.	int	.layer	Layout
Location	The starting location that this window will open up at. Only applicable to floating windows that are not set to start maximized. This value will be overridden when an open window script specifies where to open.	Point	.startingLocation	Layout

Maximizable	Determines whether or not to draw the maximize button in the upper right corner.	boolean	.maximizable	Behavior
Maximum Size	The maximum size that this window will allow itself to be resized to.	Dimension	.maximumSize	Layout
Minimum Size	The minimum size that this window will allow itself to be resized to.	Dimension	.minimumSize	Layout
Resizable	Determines whether or not to let the user resize the window.	boolean	.resizable	Behavior
Size	The dimensions of the window. This can be manipulated by selecting the window and dragging the resize handles along the windows right and bottom edges.	Dimension	.size	Layout
Start Maximized	When set to true, the window will become maximized when it is opened.	boolean	.startMaximzied	Behavior
Title	The title to be displayed in this window's titlebar. The title is also used in the Client's Windows menu.	String	.title	Appearance
Titlebar Display Policy	Determines if window's titlebar is shown in various window states. **Integer** **Property** 0 Always 1 Never 2 When Not Maximized	int	.titlebarDisplayPolicy	Appearance
Titlebar Font	The font of the window title in the titlebar.	Font	.titlebarFont	Appearance
Titlebar Height	The height of the window's titlebar.	int	.titlebarHeight	Appearance

Scripting

Scripting Functions
<ul><li>Description Returns a reference to the Root Container in the window.</li><li>Parameters none</li><li>Return Object - a reference to the Root Container, which is functionally just a Vision - Container.</li><li>Description Returns a reference to a component. The path paremeter allows you to specify the full path to the component as a string.</li><li>Parameters String path - The path to the component, using a period as a delimiter, such as "Root Container.Group.Label".</li><li>Return Object - to the component specified, or None if there is a typo in the path.</li></ul>

Extension Functions
This component does not have any extension functions.

Event Handlers

An "internalFrame" refers to the underlying object Java windowing system that windows in the Vision module use.

Fires whenever the window is shown or focused. If you want a script to fire every time a window is opened, use this event.

.source	The window that fired this event. Use source.rootContainer to get the root container.

Fires when a window is closed.

.source	The window that fired this event. Use source.rootContainer to get the root container.

Fires right before a window is closed.

.source	The window that fired this event. Use source.rootContainer to get the root container.

Fires when a window loses focus.

.source	The window that fired this event. Use source.rootContainer to get the root container.

Fires the first time a window is opened. Note that when windows are closed, they may be cached. If a window in a client is cached, subsequent attempts to open the window will not trigger this event. If you disable caching (by setting the **Cache Policy** property to **Never**) then this event will trigger every time the window is opened. Alternatively, you could use **internalFrameActivated** instead to consistently trigger a script when a window is opened.

.source	The window that fired this event. Use source.rootContainer to get the root container.

Fires whenever a bindable property of the source component changes. This works for standard and custom (dynamic) properties.

.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.
.propertyName	The name of the property that changed. Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.

This event is fired each time the window is opened and before any bindings are evaluated.

.source	The window that fired this event. Use source.rootContainer to get the root container.

This event is fired each time the window is closed.

.source	The window that fired this event. Use source.rootContainer to get the root container.

Examples

For examples of windows, please see the Vision Windows section.