

Ignition 7.9 User Manual — Component Reference

Generated 2026-10-12 from
<https://www.docs.inductiveautomation.com/docs/7.9/>

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Appendix

The appendix is your quick reference to all the Components, Expression Functions, and Scripting Functions in Ignition. Each item has a definition, property references, and examples of how to use it.

 **Components**

 **Expression Functions**

 **Scripting Functions**

 **Reference Pages**

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Components

Components are what fill up your windows with useful content. Anyone familiar with computers should already understand the basic concept of a component - they are the widgets that you deal with every day - buttons, text areas, dropdowns, charts, and so on. The Vision module comes with a host of useful components out of the box, many of which are specialized for industrial controls use. Other modules, like the Reporting module, add more components for specialty purposes.

Configuring components will likely be the bulk of a designer's work when designing a Vision project. The basic workflow is to take a component from the palette and drop it into a container on a window. From there, you can use the mouse to drag and resize the component into the correct position. 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 components too. Each shape may be individually selected, named, and has its own properties. Shapes have some additional capabilities that other 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 the component 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.

 **Reporting Components**

 **Admin**

 **Alarming**

 **Buttons**

 **Calendars**

 **Charts**

 **Containers**

 **Display**

 **Input**

 **Misc**

 **Tables**

 **Web Browser**

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Reporting Components

The following components require the Report Module, and give you access to generated reports and various ways to filter and display data.

Report Viewer

Row Selector

The row selector is a component that acts like a visual filter for datasets.

Column Selector

The column selector component is conceptually similar to the Row Selector, except that instead of filtering rows, ...

File Explorer

The File Explorer component displays a filesystem tree to the user. It can be rooted at any folder, even network fo...

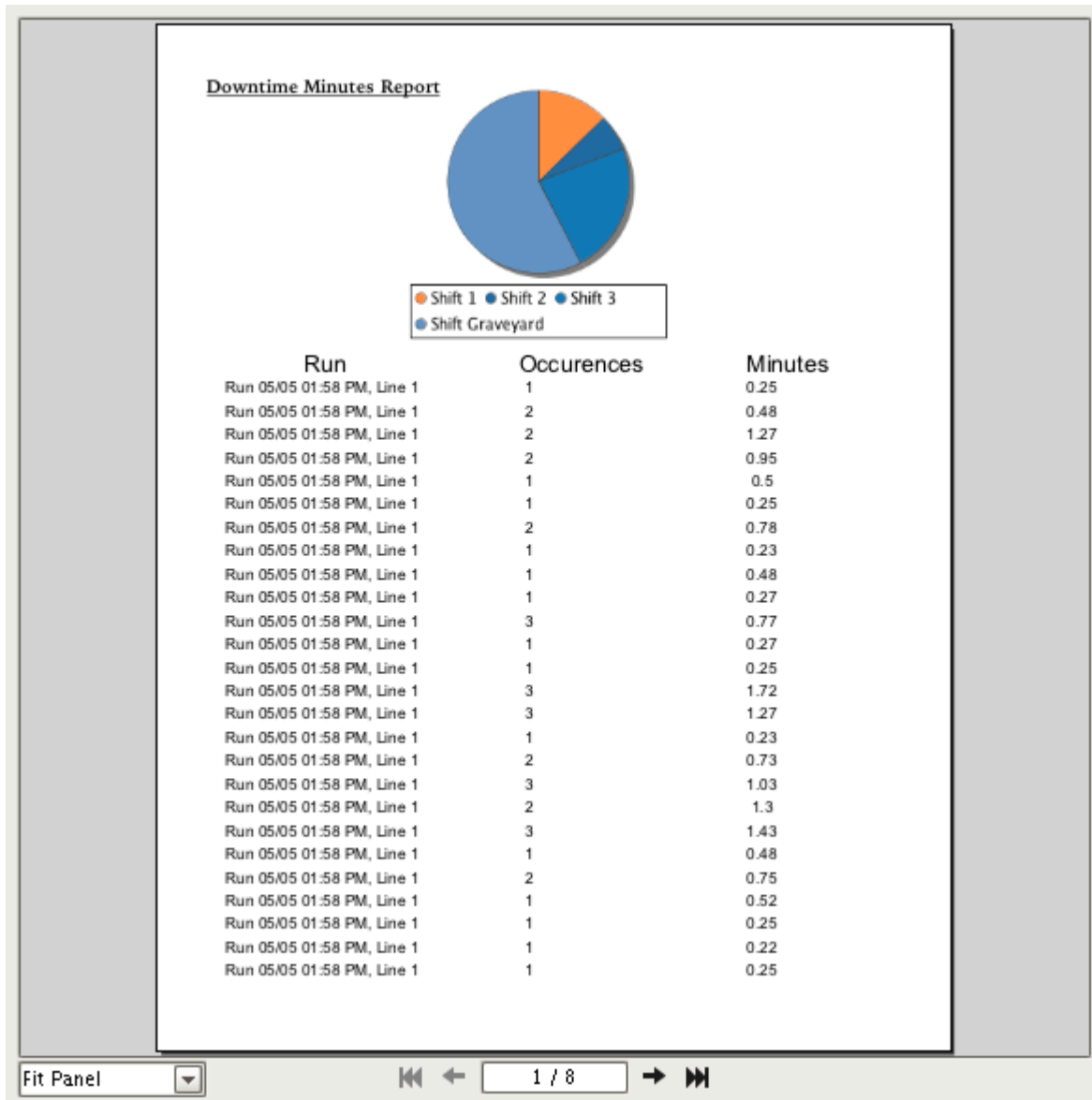
PDF Viewer

Report Design Components

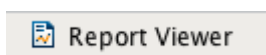
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Viewer



Component Palette Icon:



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.

Properties

Name	Description	Type	Scripting	Category
Background Color	Color that lays underneath the report.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Current Page	Current page in the report you would like to view.	Int	.currentPage	Data
Foreground Color	The foreground color the labels on the component.	Color	.foreground	Appearance
Name	The name of this component.	String	.name	Common
Page Count	Number of pages in the report.	Int	.pageCount	Data
Print Mode	Sets the printing mode. Vector is fastest and high-quality for printers that support it, but Raster mode can help spool size with older printers.	Int	.printingMode	Behavior
Report Path	Path in the Project to the Report you would like to view.	String	.reportPath	Data
Show Controls	Show the bar with the page and the zoom	Boolean	.showControls	Appearance

Name	Description	Type	Scripting	Category
	controls.			
Suggested Filename	The filename that will come up by default when the user saves the report to disk.	String	.suggestedFilename	Behavior
Visible	If disabled, the component will be hidden.	Boolean	.visible	Common
Zoom Factor	Zoom factor for this report.	Float	.zoomFactor	Data
Report Loading	New in 7.9.5 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	.reportLoading	N/A

Scripting

See the [Report Viewer Scripting Functions](#) page for the full list of scripting functions available for this component.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Examples

#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)
```

*#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")
```

New in 7.9.5

The following example utilizes the `reportLoading` property, which was introduced in 7.9.5.

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()
```

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Viewer Scripting Functions

This page details the various component and extension functions available for Vision's Report Viewer component.

Component Functions

`.print(printerName, showDialog)`

NOTE

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
 - None
- Scope
 - Client

getBytesPDF()

- 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.

NOTE

This function will return null if the trial has expired.

- Scope
 - Client

getBytesPNG()

- 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.

NOTE

This function will return null if the trial has expired.

- Scope
 - Client

saveAsPDF(fileName)

- 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
- Scope
 - Client

saveAsPNG(fileName)

- 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
- Scope
 - Client

saveAsXls(fileName)

- 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
- Scope
 - Client

Extension Functions

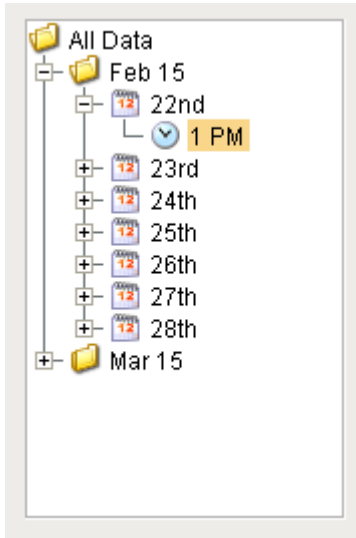
onReportGenerated

- 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
- Scope
 - Client

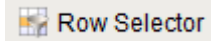
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Row Selector



Component Palette Icon:



Description



INDUCTIVE
UNIVERSITY

Row Selector

[Watch the video](#)

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.

 INFO

Additional information on the Row Selector can be found on the [Vision Reporting Components](#) page.

Properties

Name	Description	Property Type	Scripting	Category
Visible	If disabled, the component will be hidden.	Boolean	.visible	Common
Unknown Node Text	Text for any 'Unknown' nodes (nodes where the data didn't match the filter).	String	.unknownNodeText	Appearance
Unknown Node Icon	Icon for any 'Unknown' nodes (nodes where the data didn't match the filter).	String	.unknownIconPath	Appearance
Show Root Handles	Should root-level nodes have collapse handles?	boolean	.showRootHandles	Behavior
Show Node Size	If true, the number of rows in each node will be shown.	boolean	.showNodeSize	Behavior
Show All Data Node	Should the 'All Data' (root) node be shown or hidden?	boolean	.showAllDataNode	Behavior
Selection Background	The background color of the selected node.	Color	.selectionBackground	Appearance
Opaque	If false, backgrounds are not drawn. If	boolean	.opaque	Common

Name	Description	Property Type	Scripting	Category
	true, backgrounds are drawn.			
Name	The name of this component.	String	.name	Common
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.tooltiptext	Common
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance
Font	Font of text on this component.	Font	.font	Appearance
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
Data Out	The output of the row selection tree. Changes based on user selection in the filter tree.	Dataset	.dataOut	Data
Data In	The input of the row selection tree. The filter tree changes based on this Dataset.	Dataset	.dataIn	Data

Name	Description	Property Type	Scripting	Category
Cursor	The mouse cursor to use when hovering over this component.	Cursor	.cursor	Common
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Background Color	The background color of the component.	Color	.background	Appearance
All Data Node Text	Text for the 'All Data' node, if it is displayed.	String	.allDataNodeText	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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page.

Customizers

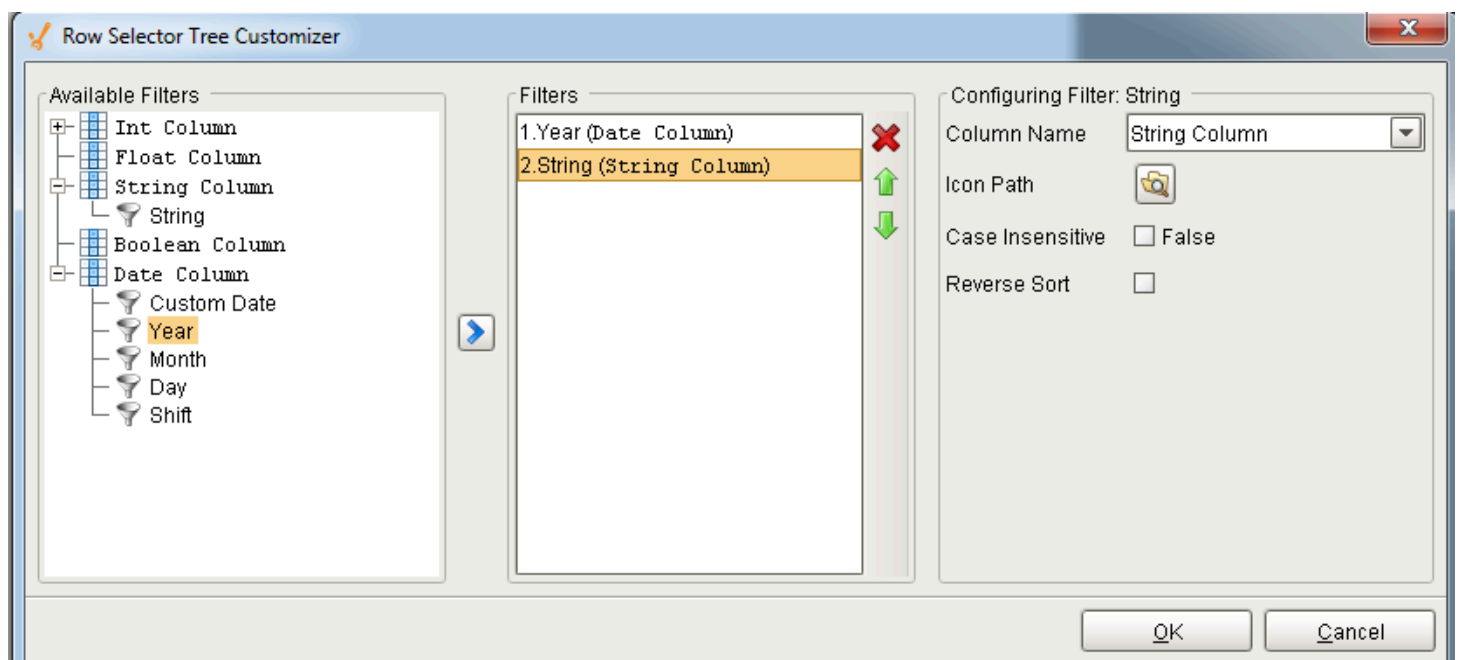
The Row Selector has its own Row Selector Tree Customizer and allows users to customize the row filtering. The customizer provides some default filters which you can use, or customize based on the dataset.

Row Selector Tree Customizer

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 () button 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.



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 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:

- Shift Filter Type - Contains a table allowing you to specify shifts with start and end times.
- Custom Date Filter Type - Contains a format string allowing you to specify the format of the custom date.
- String Filter Type - Contains a Case Insensitive property which when checked will not check case when comparing strings.
- Discrete Filter Type - Contains a format string property allowing you to specify how the integer value will be displayed. Place '%d' in the format string as a placeholder for the string value. The default is 'Item %d'.

Examples

Refer to the example on the [Vision Reporting Components](#) page.

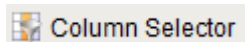
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Column Selector



Component Palette Icon:



Description



INDUCTIVE
UNIVERSITY

Column Selector

[Watch the video](#)

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.



INFO

Additional 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 thier text.	Boolean	.alphabetize	Behavior
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	Int	.cursorCode	Common
Data In	Input dataset. *Note: 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.*Note: 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

Name	Description	Property Type	Scripting	Category
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance
Group By Dataset	If true, checkboxes will be grouped by thier 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

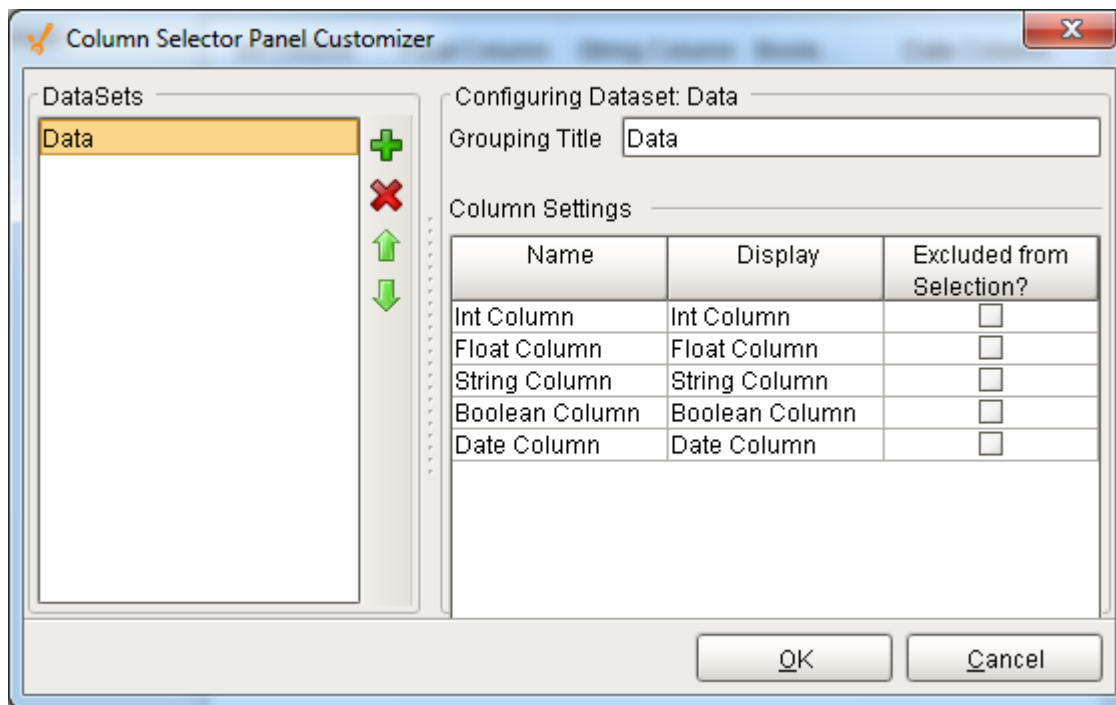
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page.

Customizers

The Column Selector component has its own Column Selector Panel Customizer that allows you to configure how the Column Selector filters columns.

Column Selector Customizer

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.



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.

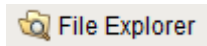
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

File Explorer



Component Palette Icon:



Description



INDUCTIVE
UNIVERSITY

File Explorer

[Watch the video](#)

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.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	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.	Color	.foreground	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on	String	.toolTipText	Common

Name	Description	Property Type	Scripting	Category
	mouseover of this component.			
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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page.

Customizers

The File Explorer component does not have a customizer.

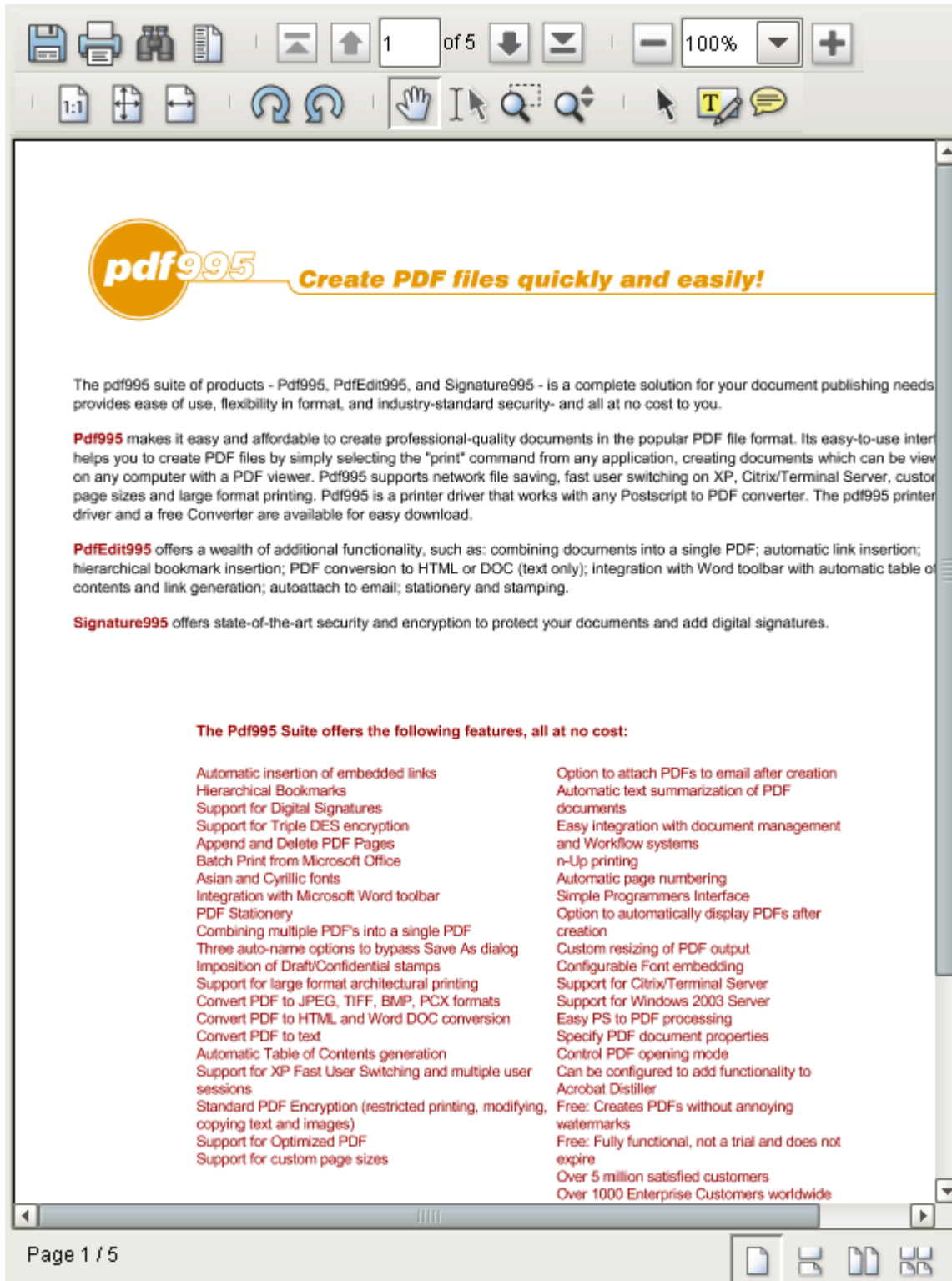
Examples

Refer to the example on the [Vision Reporting Components](#) page.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

PDF Viewer



Component Palette Icon:



Description



INDUCTIVE
UNIVERSITY

PDF Viewer

[Watch the video](#)

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 or view dynamically generated reports use the [Reporting Module](#).

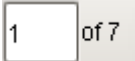
This component is typically used in conjunction with the File Explorer component, in order to create a PDF viewing window. Simply bind the Selected Path property in the PDF Viewer to the File Explorer's Selected Path property. See the [File Explorer's documentation](#), as well as the [File Explorer and PDF Viewer](#) page for further instructions on how to put these two components together.

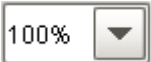
Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. NOTE that 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	Boolean	.visible	Common

Name	Description	Property Type	Scripting	Category
	hidden.			

PDF Viewer Toolbar

Toolbar Buttons	Name	Function
	Save As	Will save the currently loaded pdf to the local computer.
	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 phrase. *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 will 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.
	Current Page	Will show the current page number out of the total number of pages, also allowing a page number to be entered which

Toolbar Buttons	Name	Function
	Number	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.
	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.
	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.
	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.

Toolbar Buttons	Name	Function
	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.

Scripting

See the [PDF Viewer Scripting Functions](#) page for the full list of scripting functions available for this component.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples

Refer to the example on the [Vision Reporting Components](#) page.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

PDF Viewer Scripting Functions

Scripting Functions

.loadPDFBytes(bytes, name)

- 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
- Scope
 - Client

.print(showDialog)

- Description
 - This function will print the PDF.
- Parameters
 - boolean **showDialog** - If true, shows the user a print dialog. Default is true [optional]
- Return

- Nothing
- Scope
 - Client

.setZoomFactor(zoom)

- 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
- Scope
 - Client

Extension Functions

This component does not have extension functions associated with it.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Design Components

The Report module features many different report-specific components. This section contains property listings for each component.

The following components are used exclusively inside the Report Module.

Report Component - Report Object

The report itself. This component has properties that control the dimensions, margins, and more. These settings i...

Report Component - Page Object

Each report may consist of multiple pages. The Page component may be selected by clicking on a Page item in th...

Report Component - Table

The Report Table displays data in a structured, repetitious format which gives reporting users the ability to flexibl...

Report Component - Crosstab Table

CrossTab Tables are commonly used to summarize the relationship between two categories of data.

Report Component - Simple Table

The Simple Table is a grid-like table structure that dynamically creates new rows and columns for rows returned ...

Report Component - Labels

Labels can be used to print out a host of different types of labels depending on how you configure the Label prop...

Report Component - Barcode

The Barcode component allows reports to contain dynamically generated barcodes based on how you configure i...

Report Component - Images

The Image component lets you embed images into reports.

Report Component - Timeseries Chart

Report Component - XY Chart

The XY Chart, while similar to the Timeseries Chart, uses a numerical value for the Domain Axis instead of a Datet...

Report Component - Bar Chart

A Bar Chart is used to display data in a chart using bars.

Report Component - Pie Chart

The Pie Chart takes a Data Source with two keys: a Label (typically a string) and a Value (typical a numerical value).

Report Component - Text Shape

The Text shape component creates a text area for static or data key bound content.

Report Component - Line Shape

The Line shape component can be used to display a straight line any where in a report.

Report Component - Rectangle Shape

The Rectangle shape component creates a square or rectangle and can be used anywhere on a report.

Report Component - Ellipse Shape

The Ellipse shape component creates circles and ellipses that can be used in a report.

Report Component - Star Shape

The Star shape component creates a star and can be used anywhere on a report.

Report Component - Polygon Shape

The Polygon shape component creates custom polygon shapes by drawing and connecting lines, and can be use...

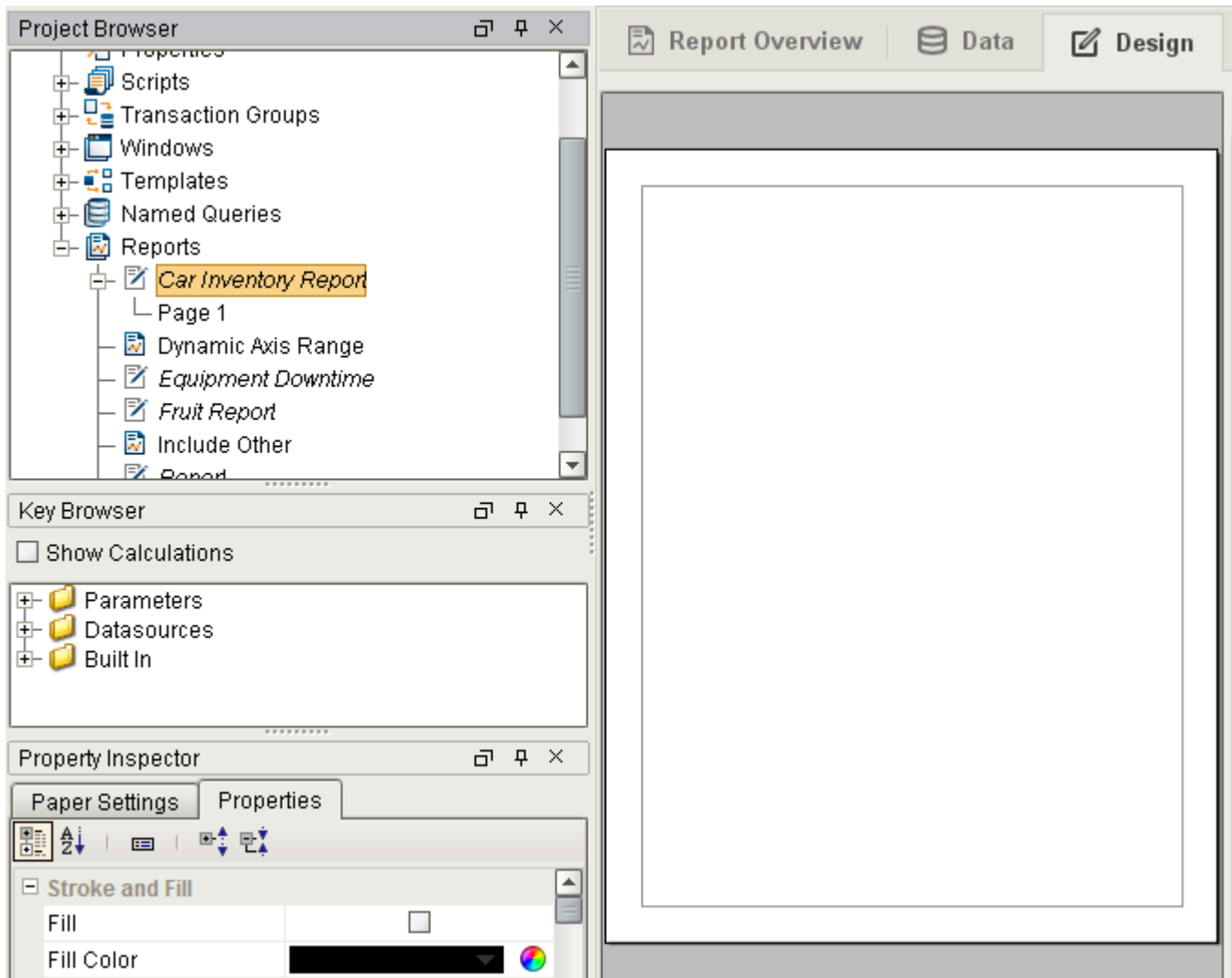
Report Component - Pencil Shape

The Pencil shape component lets you draw free-hand path segments that automatically smooth the curves.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

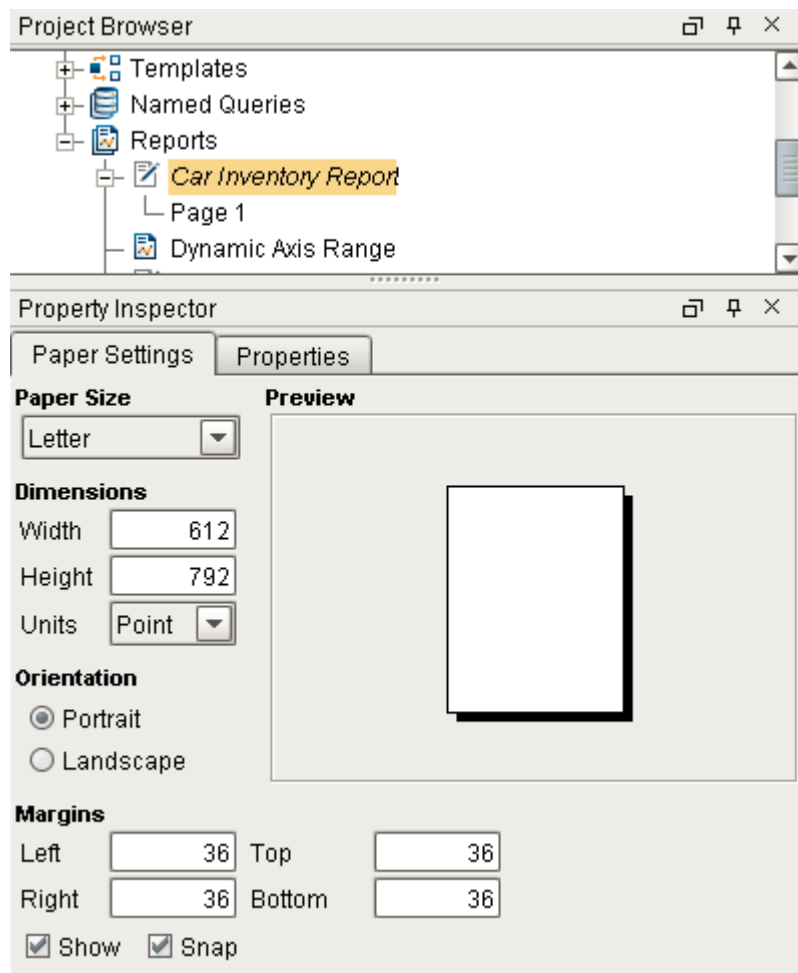
Report Component - Report Object



Description

The report itself. This component has properties that control the dimensions, margins, and more. These settings impact all pages in the report. It may be selected by clicking on the **Report** object in the Project Browser.

Paper Settings



Property	Description
Paper Size	The desired paper size for the report. Many common sizes are available. When a size is selected, the width and height of the selected size are loaded into the Dimensions property.
Dimensions	Contains the Width and Height of the page. These properties work in conjunction with the Units property. The Dimensions may be changed manually here, or by modifying the Paper Size property. This section also contains a Units drop down with the following units of measurement: • Inch. • Point (Printer Point). 1 inch = 72 points • Centimeter. • Millimeter. • Pica. 1 inch = 6 picas
Orientation	Determines if the report should appear as Portrait (taller reports) or Landscape (wider reports)
Margins	Allows you to specify the position of the margin on the report. If the Show property is enabled, the margin is visible on the report's Design Panel. The margin will never appear on the Preview Panel or on the generated reports. The Snap property, when enabled, will cause report components to snap to the margin if moved within close proximity.

Properties Tab

Property Inspector

Paper Settings

Properties

2

Grid Options

Grid Spacing

9

Show Grid

☐

Snap to Grid

☐

Document Options

Page Layout

Single

String for Null

<NA>

Basic Properties

Roll

0

Scale X

1

Scale Y

1

Visible

☒

X

0

Y

0

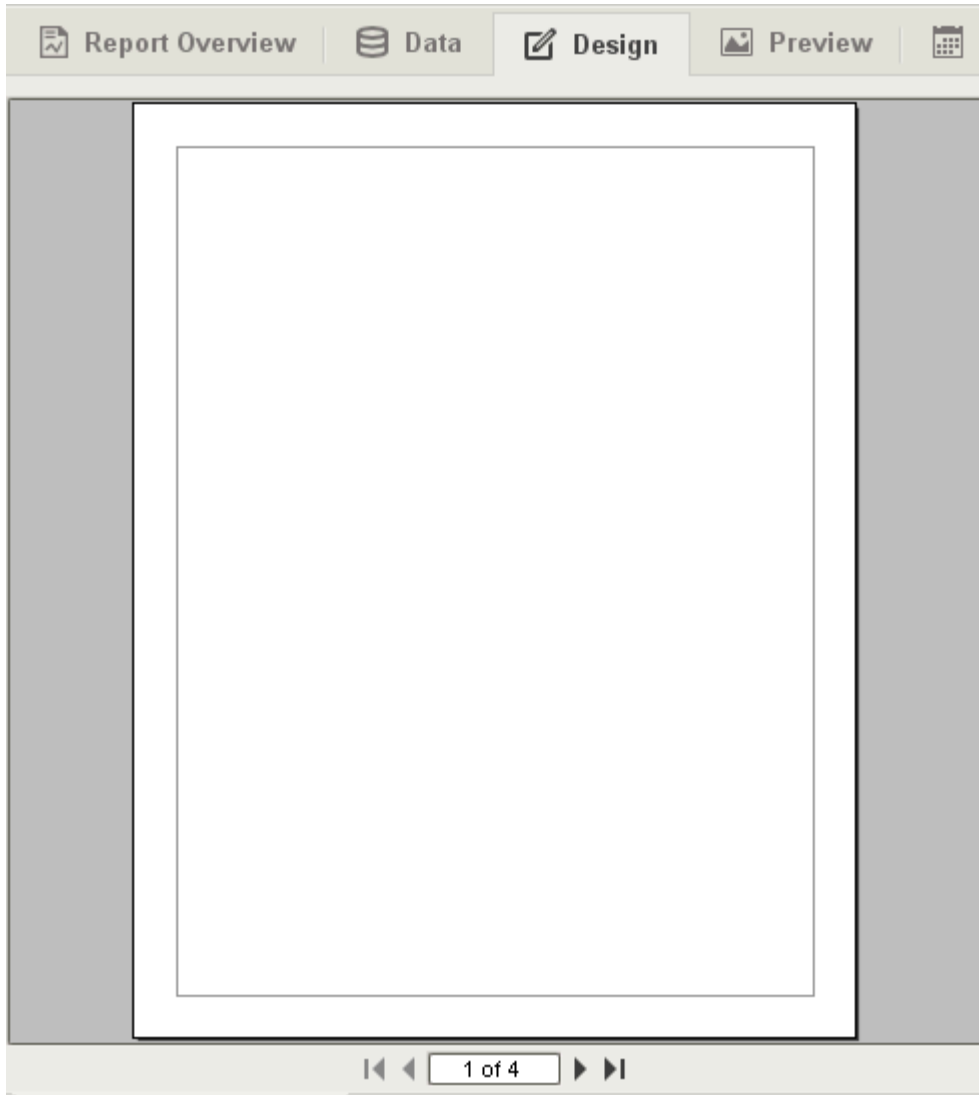
Property	Description	Category
Grid Spacing	The space between each grid in Printer Points. Defaults to 9, which is 1/8th of an inch.	Grid Options
Show Grid	Enables a grid on the Design Panel. Useful for spacing out each component on the report. The grid is not visible on the Preview Panel, or on the resulting report.	Grid Options
Snap to Grid	When dragging a component, enabling this property will cause the component to snap to the grid.	Grid Options
Page Layout	Specifies how the page looks in the designer. Has the following options: • Single: The default value. Only a single page of the report appears in the Design Panel at a time. • Double: Shows two pages in the Design Panel, horizontally adjacent, with page 1 starting on the left. • Quadruple: Shows four pages in the Design Panel, horizontally adjacent • Facing: Shows two pages at a time. Cycling through pages moves through two pages at a time, starting with page 1 on the right (similar to reading through a book). • Continuous: Stacks the pages vertically adjacent, with page 1 at the top. • ContinuousDouble: A combination of Double and Continuous; two horizontally adjacent pages are stacked vertically, with pages 1 and 2 at the top.	Document Options
String for Null	Specifies the value to use should a key return a null value. Note that individual Text Shapes also have a String for Null property that will override this property.	Document Options
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties

Property	Description	Category
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties
Height	Height of this shape in pixels.	Basic Properties
X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page.	Basic Properties
Y	Vertical distance in pixels between the top edge of this shape and the top edge of the page.	Basic Properties

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

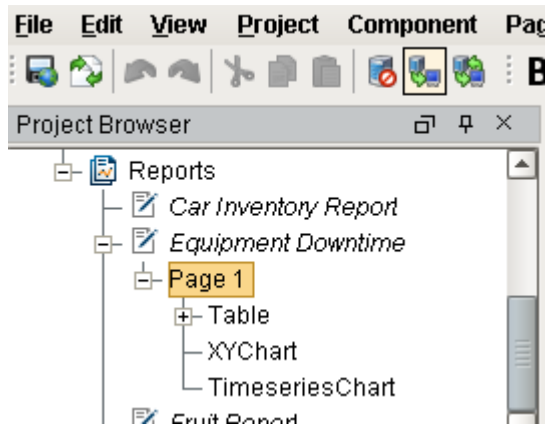
For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Component - Page Object

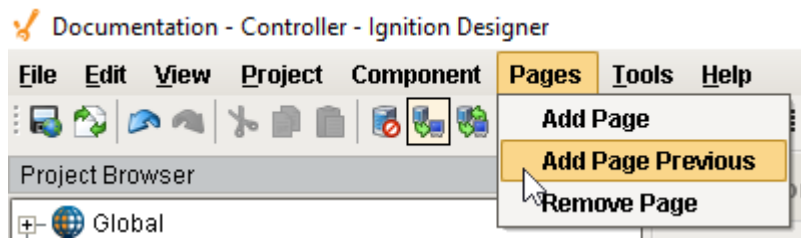


Description

Each report may consist of multiple pages. The Page component may be selected by clicking on a Page item in the Project Browser.

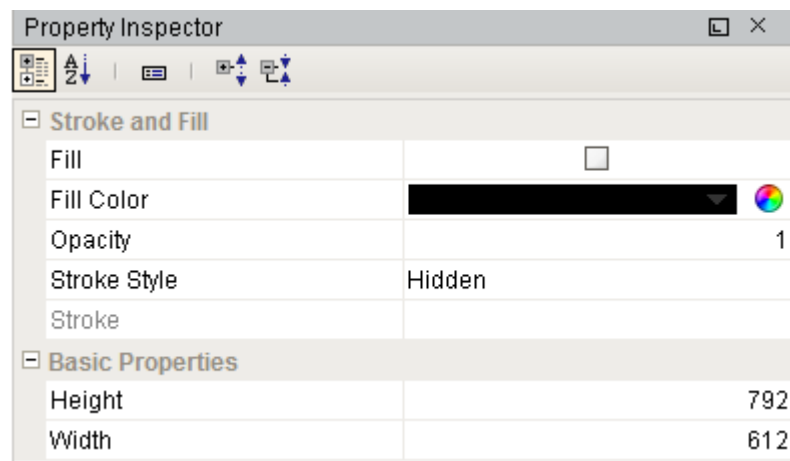


Pages are added and removed from the the menu bar at the top of the browser.



Menu Item	Description
Add Page	Adds a new page to the report after the currently selected page.
Add Page Previous	Adds a new page to the report prior to the currently selected page. A cover page may easily be added to a pre existing report by using Add Page Previous.
Remove Page	Deletes the page that is currently visible in the main workspace.

Properties



Property	Description	Category
Fill	If true, the shape will fill its space with color.	Stroke and Fill
Fill Selected	If fill is selected, the color that will fill the shape.	Stroke and Fill
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Height	Height of this shape in pixels	Basic Properties
Width	Width of this shape in pixels.	Basic Properties

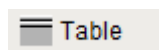
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Component - Table

Objects Details			

Report Design Component Palette Icon:



Description

The **Report Table** displays data in a structured, repetitious format which gives reporting users the ability to flexibly layout and organize tabular data in a variety of ways. The Reporting engine automatically creates new pages to fit all data within the table's boundaries while applying formatting preferences such as font, size, layout, and alignment. Combine this feature with powerful data manipulation and layout tools, you get an object that forms the basis of your reports.

Configure Table Properties

In the Property Inspector, use the Configure Table and Properties tabs to configure your table. Here is a list of all the Table properties and their descriptions.

Configure Table Tab

Property Inspector

Configure Table

Properties

Data Key

Objects

Grouping

Objects

Rows

☐ Header

☒ Details

☐ Summary

Sorting

☒ Default

☐ Basic

☐ TopN

Options

☐ Page Break

Make TableGroup

Name	Description
Data Key	Unique identifier or placeholder that points to your data. Typically a data source is used for this property.
Grouping	Allows you to group the data by columns in the datasource. Will always at least group by the entire datasource.
Rows	Rows in a table. There are three types: Header Row: A single row inserted at the start of the Detail Rows. Commonly used to display column headers. Details Row: Represents the majority of the data on a table. Summary Row: A single row inserted after all the Detail Rows. Commonly used to display totals or summary values.
Sorting	Orders data in the table. Sorting occurs after the data has already been retrieved in the data source. There are three options: Default: Data sorted based on the order in which it is retrieved. When selected, the table will not sort the rows. Basic: Allows a list of keys to be configured. The data is sorted in the order the keys are selected. TopN: Uses a single key path, with a Count value that allows a limit to the number of rows that are processed.
Page Break	Creates a page break between each grouping.
Make Table Group	Turns the Table into a TableGroup , where you can add 'child' and 'peer' tables.

Table Properties

Property Inspector

Configure TableProperties

2

Table

Data KeyUsed Car Inventory

Column Count1

Column Spacing10

Filter Key

Starting Page Break☐

Stroke and Fill

Fill☐

Fill Color

Opacity1

Stroke StyleHidden

Stroke

Basic Properties

Roll0

Scale X1

Scale Y1

Visible☒

Width540

Height157

X36

Y569

Name	Description	Category
Data Key	Data Key associated with the table. Should be the data key of the data source that contains rows such as a query and should match the Data Key of the Configure Table Tab.	Table
Column Count	The number of columns that the table will repeat horizontally on a page before paginating. Size your table to be the width of a single column.	Table
Column Spacing	The space between table columns.	Table
Filter Key	A key or expression used to prune the underlying data source's rows down. If not blank, only rows for which this key expression evaluates to true will be included.	Table
Starting Page Break	If true, the table will always start on a new page if part of a table group.	Table
Fill	If true, the shape will fill its space with color.	Stroke and Fill
Fill Color	If Fill is selected, the color that will fill the shape.	Stroke and Fill
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties

Name	Description	Category
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties
Height	Height of this shape in pixels.	Basic Properties
X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page.	Basic Properties
Y	Vertical distance in pixels between the upper edge of this shape and the top edge of the page.	Basic Properties

Table Group Properties

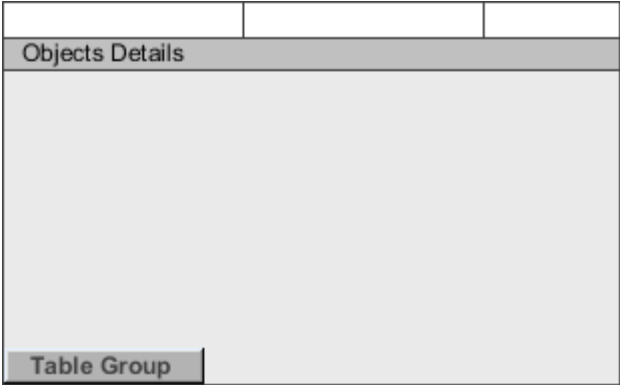


Table Groups are a collection of multiple tables represented by a single component. Examples of their use can be found on the [Table Groups](#) page.

Property Inspector

Configure TableGroup

Properties

Top Object

└ Child Object

Peer Object

+

🗑

↑

↓

←

→

✖

Key

Top Object

✔

Start on new page

☐ false

New page per row

☐ false

Name	Icon	Description
Add Child/Peer Table	+	Adds a new table to the group. Allows for the creation of a Child or Peer table. Child Table: A copy of this table will be inserted in between each row of the parent table Peer Table: This table will appear after any preceding peers. Note that the terms "child" and "peer" are relative to the adjacent tables. More details can be found on the Table Groups page.
Delete Table		Removes the selected table from the group.
Move Up	↑	Moves the selected table up in the hierarchy. Tables are always presented in a Top-to-Bottom order, so your initial table should always be at the top.
Move Down	↓	Moves the selected table down in the hierarchy.
Move Left	←	Moves the selected table left in the hierarchy, potentially changing the peer/child relationship with any tables listed above the selected table.
Move Right	→	Moves the selected table right in the hierarchy, potentially changing the peer/child relationship with any tables listed above the selected table.
Destroy Table Group		Removes the table group. Note that this will delete any configurations on all tables below the top-most table.
Start on new page	N/A	Adds an explicit page break between the selected table and the prior peer table.
New page per row	N/A	Adds an explicit page break for each row of the selected table.
Key	N/A	The Key of the selected table.

Row Properties

Factory	Capacity (Units)	Run Time (Minutes)	Units Produced
Factory Production Header			
@Facility@	@Capacity@	@Minutes@	@Produced@
Factory Production Details			
			Total Units Produced: @total.Produced@
Factory Production Summary			

All **Table Rows** (**Header**, **Details**, and **Summary**) have the same properties, and all rows in the table use its own settings. You can click on each row to see the property values for that row. Here is a list of all the Table Row properties and their descriptions.

Property Inspector		
2		
Table Row		
Structured Columns	☒	
Column Count		4
Move to Bottom	☐	
Print Empty Group	☒	
Sync Alternate Versions	☐	
Sync Row Above	☐	
Version Key		
Stroke and Fill		
Fill	☐	
Fill Color		
Opacity		1
Stroke Style	Hidden	
Stroke		
Pagination		
Min. Remainder Height		72
Min. Split Height		72
Page Break		
Reprint When Wrapped	☒	
Stay with children		1
Basic Properties		
Roll		0
Scale X		1
Scale Y		1
Visible	☒	
Width		511
Height		18
X		0
Y		34

Name	Description	Category
Structured Columns	If true, this row will be organized into columns, otherwise, the row will allow free form design.	Table Row
Column Count	If this row is structured, this is the number of columns in the row.	Table Row
Move to Bottom	If true, this row will be moved to the bottom of the table.	Table Row
Print Empty Group	If true, row will print even if it has no children. Only applicable for header or summary rows.	Table Row
Sync Alternate Versions	If checked, the columns widths of this row will be synchronized with alternate row versions.	Table Row
Sync Row Above	If checked, the column widths of this row and the row above it will be synchronized as long as they have the same number of fields.	Table Row
Version Key	Data Key to drive which version of the table row to display.	Table Row
Fill	If true, the shape will fill its space with color.	Stroke and Fill
Fill Color	If Fill is selected, the color that will fill the shape.	Stroke and Fill
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Min. Remainder Height	Minimum distance in points between the bottom of the row that will split onto a new page. The default value of 72 points is equivalent to 1 inch.	Pagination

Name	Description	Category
	ⓘ This and Min. Split Height are how the table determines when to add another row on to a page, or split the row to a new page. If using rows with a large height (such as those that contain another component, like a chart) you may find that the row is "split" horizontally across multiple pages. Setting this and Min. Split Height to a larger number will help prevent these splits.	
Min. Split Height	Minimum distance in points between the top of the row that will split onto a new page. The default value of 72 points is equivalent to 1 inch. Typically set to the same number as Min. Remainder Height. See the information box on the Min. Remainder Height property above.	Pagination
Page Break	Data Key or expressions used to determine whether row should force a page break.	Pagination
Reprint When Wrapped	If true, this row will be reprinted on a new page when its children cross a page boundary.	Pagination
Stay with Children	The number of children this row must be accompanied by, in the case where some children run off the bottom of a page.	Pagination
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties

Name	Description	Category
Height	Height of this shape in pixels.	Basic Properties
X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page.	Basic Properties
Y	Vertical distance in pixels between the upper edge of this shape and the top edge of the page.	Basic Properties

Cell Properties

The cells on the Table are simply **Text Shape** components embedded directly in a row. They can be multi selected with shift or alt, to change the properties of multiple cells at once. For more information, please see the **Text Shape** page.

Examples

The Table component has a lot of functions and features. Examples of how you can use these functions and features are included in each of the sections below.

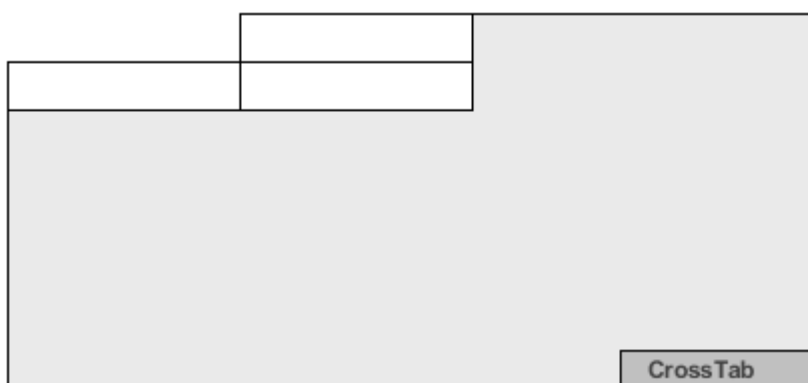
- **Table Rows**
- **Table Row Versions**
- **Charts Inside of Tables**
- **Grouping Data Inside of Tables**
- **Table Groups**

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

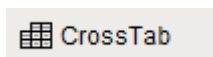
For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Component - Crosstab Table

CrossTab Table



Component Palette Icon:



Description



INDUCTIVE
UNIVERSITY

CrossTab Table

[Watch the video](#)

CrossTab Tables are commonly used to summarize the relationship between two categories of data by showing summaries of cross sections of the data source. The CrossTab Table allows you to very quickly show cross sections of data where the other table types are focused on a dynamic number of rows and columns. CrossTab data should contain the following:

- Lots of repetitious data. Sums, Averages, and other Aggregating functions are well represented by CrossTab Tables.

- A data source that provides at least two columns of data which are repetitious compared to the number of rows.
- One or more columns that represent a value that requires calculation. Examples are: summing money, displaying average response times, counting occurrences, etc. These calculations may be provided as columns or calculations of the data source, or as **Keychain Expressions**.

CrossTab tables can be customized using table and cell properties. It also has a number of style templates. Select any one of the style templates, use as is, or double click on any of the cells to change the individual cell's properties. Configure the style in the Design panel, and view the table with your data in the Preview panel.

Table Properties

Property Inspector		
2		
Table		
Data Key		
Columns		1
Filter Key		
Header Columns		1
Header Rows		1
Reprint Header Rows	☒	
Rows		1
Style		Elegant
Stroke and Fill		
Fill	☐	
Fill Color		
Opacity		1
Stroke Style		Hidden
Stroke		
Basic Properties		
Roll		0
Scale X		1
Scale Y		1
Visible	☒	
Width		354
Height		172
X		59
Y		132

Property	Description	Category
Data Key	The data key associated with this Tabular component. Should be the data key of a data source that contains rows, such as a query.	Table
Columns	Number of columns.	Table
Filter Key	A key or expression used to prune the underlying data source's rows down. If not blank, only rows for which this key expression evaluates to true will be included.	Table
Header Columns	Number of Header Columns.	Table
Header Rows	Number of Header Rows.	Table
Reprint Header Rows	Enable header row printing when pagination occurs.	Table
Rows	Number of Rows.	Table
Style	Give the CrossTab or Simple Table a pre-configured style. • None • Classic 1 through Classic 3 • Default • Elegant - with double border around the table. • Grid 1 through Grid 8 • List 1 through List 6 • Professional • Simple 1 through Simple 6 The Style section below shows several examples of some of the style templates you can choose from.	Table
Fill	If true, the shape will fill its space with color.	Stroke and Fill

Property	Description	Category
Fill Color	If Fill is selected, the color that will fill the shape.	Stroke and Fill
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties
Height	Height of this shape in pixels.	Basic Properties
X	Horizontal distance in pixels between the left edge of this shape to the left edge of the page.	Basic Properties
Y	Vertical distance in pixels between the upper edge of this shape to the top edge of the page.	Basic Properties

Cell Properties

Property Inspector	
2	
Text Properties	
Text	
Text Color	
Character Spacing	0
Coalesce Newlines	☐
Font	Arial 12.0 (Arial)
Horizontal Alignment	Center
Line Spacing	1
Margin	5; 5; 5; 5
Overflow Behavior	Grow row
Underlined	☐
Vertical Alignment	Middle
Data Key Format Properties	
Date Format	MMM d, yyyy
Null Format	<N/A>
Number Format	#0.##
Negative In Red	☐
Stroke and Fill	
Fill	☐
Fill Color	
Opacity	1
Stroke Style	Hidden
Stroke	
Grouping	
Grouping Key	
Rectangle Shape	
Radius	0
Basic Properties	
Roll	0
Scale X	1
Scale Y	1
Visible	☒
Width	120
Height	25
X	0
Y	25

Property	Description	Category
Text	Text to display in the shape.	Text Properties
Text Color	Color of the text.	Text Properties
Character Spacing	Amount of extra spacing to add between characters. Negative spacing makes characters closer together.	Text Properties
Coalesce Newlines	If true, consecutive line breaks will be merged into one.	Text Properties
Font	Font to use for the text.	Text Properties
Horizontal Alignment	Horizontal Alignment for text: Left; Right; Center; Full.	Text Properties
Line Spacing	Spacing between lines of text. 1 is single spacing, 2 is double spacing, etc.	Text Properties
Margin	Margin between texts and bounds of shape in pixels. Format is top: left: bottom; right.	Text Properties
Overflow Behavior	How text that overflows the bounds of the shape should be handled. • Paginate - text is pushed to the next page if the number of characters exceed the shape. • Shrink text to fit - text fits into the shape, and the font is smaller when previewed or printed. • Grow row - text fits into the shape, and shows when the shape is expanded, previewed, or printed.	Text Properties
Underlined	If true, use underlined text.	Text Properties
Vertical Alignment	Vertical alignment for text: Top; Middle; Bottom.	Text Properties
Date Format	Formats to use for dates in this shape.	Data Key and Format Properties

Property	Description	Category
Null Format	Formats to use for null values in this shape.	Data Key and Format Properties
Number Format	Formats to use for numbers in this shape.	Data Key and Format Properties
Negative in Red	If true, show negative numbers in red.	Data Key and Format Properties
Fill	If true, the shape will fill its space with color.	Stroke and Fill
Fill Color	If Fill is selected, the color that will fill the shape.	Stroke and Fill
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Grouping	Grouping Key will add a row or column for each Grouping the key generates.	Grouping
Radius	Amount to radius the corners of this shape.	Rectangle Shape
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties

Property	Description	Category
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties
Height	Height of this shape in pixels.	Basic Properties
X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page.	Basic Properties
Y	Vertical distance in pixels between the upper edge of this shape and the top edge of the page.	Basic Properties

Cell Functions

Cells have a functions feature. Double click on a cell and then right click to see the functions. All the functions are the same for any cell that is selected.

Clear Contents
Add Row Above
Add Row Below
Add Column Before
Add Column After
Remove Row
Remove Column
Merge Cells
Split Cell

Function	Description
Clear Contents	Deletes all contents from the selected cell.
Add Row Above	Adds a row above the selected cell.
Add Row Below	Adds a row below the selected cell.
Add Column Before	Adds a column before the selected cell.
Add Column After	Adds a column after the selected cell.
Remove Row	Removes the selected row.
Remove Column	Removes the selected column.
Merge Cells	Combines the selected cells.
Split Cell	Splits the selected cell into individual cells.

Style Templates

Here is a sampling of several pre-configured style templates. There are many more templates to choose from in the dropdown list of the Table's Style property. You can make changes to any of the pre-configured styles by double clicking on any cell in the CrossTab Table and editing the cell's properties.

Default

	<i>Conveyor Line</i>	<i>Filler</i>	<i>Labeler</i>
DT Count	2	2	2
AVG DT	78	68	84

Elegant

	Conveyor Line	Filler	Labeler
DT Count	2	2	2
AVG DT	78	68	84

Professional

	Conveyor Line	Filler	Labeler
DT Count	2	2	2
AVG DT	78	68	84

Classic 1

	<i>Conveyor Line</i>	<i>Filler</i>	<i>Labeler</i>
DT Count	2	2	2
AVG DT	78	68	84

Grid 2

	Conveyor Line	Filler	Labeler
DT Count	2	2	2
AVG DT	78	68	84

List 6

	Conveyor Line	Filler	Labeler
DT Count	2	2	2
AVG DT	78	68	84

Simple 3

	Conveyor Line	Filler	Labeler
DT Count	2	2	2
AVG DT	78	68	84

Example

The Elegant Style template was used for this CrossTab table in a report. This CrossTab Table shows the downtime in minutes for each piece of equipment at each site.

Equipment Downtime by Site				
	Motor	Conveyor Line	Pallet Wrapper	Palletizer
Site A	15	56	58	20
Site B	100	148	55	40
Site C	96	50	45	30

Downtime Dataset - used for this CrossTab Table example

```
Equipment,Time,Site
Motor, 15, Site A
Motor, 23, Site A
Conveyor Line, 148, Site B
Pallet Wrapper, 58, Site A
Motor, 96, Site C
Conveyor Line, 23, Site B
Palletizer, 40, Site B
Palletizer, 30, Site C
Conveyor Line, 56, Site A
Conveyor Line, 50, Site C
Pallet Wrapper, 45, Site C
Motor, 43, Site C
Palletizer, 20, Site A
Pallet Wrapper, 55, Site B
Motor, 100, Site B
```

The data keys were used to populate the CrossTab Table with data. The property values used to configure the CrossTab Table are shown in the property list.

	@Equipment@
@Site@	@Time@

CrossTab

Property	Value
Style	Elegant
Text Color (for Equipment and Site Headers)	Red
Font	14
Horizontal Alignment	Center
Horizontal Alignment (for Site)	Right

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Component - Simple Table

Report Design Component Palette Icon:

 Simple Table

Description



INDUCTIVE
UNIVERSITY

Simple Table

[Watch the video](#)

With the **Simple Table**, you can very quickly add a table inside a report. The Simple Table is a grid-like table structure that dynamically creates new rows and columns for rows returned by the Data Keys on the component. It can be customized using table properties, and has quite a few pre-configured style templates you can use to get started. Select any one of the style templates, use as is, or double click on any of the cells to change the individual cell's properties. Configure the style in the Design panel, and view the table with your data in the Preview panel.

Table Properties

Property Inspector		
2		
Table		
Data Key		
Columns		3
Filter Key		
Header Columns		0
Header Rows		1
Rows		3
Style	none	
Stroke and Fill		
Fill		
Fill Color		
Opacity		1
Stroke Style	Hidden	
Stroke		
Basic Properties		
Roll		0
Scale X		1
Scale Y		1
Visible		
Width		360
Height		100
X		62.642
Y		97.824

Property	Description	Category
Data Key	The data key associated with this Tabular component. Should be the data key of a data source that contains rows, such as a query.	Table
Columns	Number of columns.	Table
Filter Key	A key or expression used to prune the underlying data source's rows down. If not blank, only rows for which this key expression evaluates to true will be included.	Table
Header Columns	Number of Header Columns.	Table
Header Rows	Number of Header Rows.	Table
Reprint Header Rows	Enable header row printing when pagination occurs.	Table
Rows	Number of Rows.	Table
Style	Give the CrossTab or Simple Table a pre-configured style. • None • Classic 1 through Classic 3 • Default • Elegant - with double border around the table. • Grid 1 through Grid 8 • List 1 through List 6 • Professional • Simple 1 through Simple 6 The Style section below shows several examples of some of the style templates you can choose from.	Table
Fill	If true, the shape will fill its space with color.	Stroke and Fill

Property	Description	Category
Fill Color	If Fill is selected, the color that will fill the shape.	Stroke and Fill
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties
Height	Height of this shape in pixels.	Basic Properties
X	Horizontal distance in pixels between the left edge of this shape to the left edge of the page.	Basic Properties
Y	Vertical distance in pixels between the upper edge of this shape to the top edge of the page.	Basic Properties

Cell Properties

Property Inspector

2

Text Properties

Text	
Text Color	
Character Spacing	0
Coalesce Newlines	☐
Font	Arial 12.0 (Arial)
Horizontal Alignment	Center
Line Spacing	1
Margin	5; 5; 5; 5
Overflow Behavior	Grow row
Underlined	☐
Vertical Alignment	Middle

Data Key Format Properties

Date Format	MMM d, yyyy	
Null Format	<N/A>	
Number Format	#0.##	
Negative In Red	☐	

Stroke and Fill

Fill	☐
Fill Color	
Opacity	1
Stroke Style	Hidden
Stroke	

Grouping

Grouping Key	
--------------	--

Rectangle Shape

Radius	0
--------	---

Basic Properties

Roll	0
Scale X	1
Scale Y	1
Visible	☒
Width	120
Height	25
X	0
Y	25

Property	Description	Category
Text	Text to display in the shape.	Text Properties
Text Color	Color of the text.	Text Properties
Character Spacing	Amount of extra spacing to add between characters. Negative spacing makes characters closer together.	Text Properties
Coalesce Newlines	If true, consecutive line breaks will be merged into one.	Text Properties
Font	Font to use for the text.	Text Properties
Horizontal Alignment	Horizontal Alignment for text: Left; Right; Center; Full.	Text Properties
Line Spacing	Spacing between lines of text. 1 is single spacing, 2 is double spacing, etc.	Text Properties
Margin	Margin between texts and bounds of shape in pixels. Format is top: left: bottom; right.	Text Properties
Overflow Behavior	How text that overflows the bounds of the shape should be handled. • Paginate - text is pushed to the next page if the number of characters exceed the shape. • Shrink text to fit - text fits into the shape, and the font is smaller when previewed or printed. • Grow row - text fits into the shape, and shows when the shape is expanded, previewed, or printed.	Text Properties
Underlined	If true, use underlined text.	Text Properties
Vertical Alignment	Vertical alignment for text: Top; Middle; Bottom.	Text Properties
Date Format	Formats to use for dates in this shape.	Data Key and Format Properties

Property	Description	Category
Null Format	Formats to use for null values in this shape.	Data Key and Format Properties
Number Format	Formats to use for numbers in this shape.	Data Key and Format Properties
Negative in Red	If true, show negative numbers in red.	Data Key and Format Properties
Fill	If true, the shape will fill its space with color.	Stroke and Fill
Fill Color	If Fill is selected, the color that will fill the shape.	Stroke and Fill
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Grouping	Grouping Key will add a row or column for each Grouping the key generates.	Grouping
Radius	Amount to radius the corners of this shape.	Rectangle Shape
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties

Property	Description	Category
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties
Height	Height of this shape in pixels.	Basic Properties
X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page.	Basic Properties
Y	Vertical distance in pixels between the upper edge of this shape and the top edge of the page.	Basic Properties

Cell Functions

Cells have a functions feature. Double click on a cell and then right click to see the functions. All the functions are the same for any cell that is selected.

Clear Contents
Add Row Above
Add Row Below
Add Column Before
Add Column After
Remove Row
Remove Column
Merge Cells
Split Cell

Function	Description
Clear Contents	Deletes all contents from the selected cell.
Add Row Above	Adds a row above the selected cell.
Add Row Below	Adds a row below the selected cell.
Add Column Before	Adds a column before the selected cell.
Add Column After	Adds a column after the selected cell.
Remove Row	Removes the selected row.
Remove Column	Removes the selected column.
Merge Cells	Combines the selected cells.
Split Cell	Splits the selected cell into individual cells.

Style Templates

Here is a sampling of several pre-configured style templates. There are many more templates to choose from in the dropdown list of the Table's Style property. You can make changes to any of the pre-configured styles by double clicking on any cell in the Simple Table and editing the cell's properties.

Default

	<i>Conveyor Line</i>	<i>Filler</i>	<i>Labeler</i>
DT Count	2	2	2
AVG DT	78	68	84

Elegant

	Conveyor Line	Filler	Labeler
DT Count	2	2	2
AVG DT	78	68	84

Professional

	Conveyor Line	Filler	Labeler
DT Count	2	2	2
AVG DT	78	68	84

Classic 1

	<i>Conveyor Line</i>	<i>Filler</i>	<i>Labeler</i>
DT Count	2	2	2
AVG DT	78	68	84

Grid 2

	Conveyor Line	Filler	Labeler
DT Count	2	2	2
AVG DT	78	68	84

List 6

	Conveyor Line	Filler	Labeler
DT Count	2	2	2
AVG DT	78	68	84

Simple 3

	Conveyor Line	Filler	Labeler
DT Count	2	2	2
AVG DT	78	68	84

Example

This example uses a Simple Table in a report. The Simple Table shows the total number of downtime events, total downtime in minutes, and average downtime for each piece of equipment.

Equipment Downtime Events				
	Conveyor Line	Filler	Palletizer	Labeler
Downtime Count	5	5	5	5
Total Downtime	399	286	234	390
AVG Downtime	79.8	57.2	46.8	78

The Simple Table uses the **Show Calculation** property in the Key Browser and several of the **Built-in Keys**: 'count', 'total', and 'average.' Other properties were modified to make the column headers and row headers stand out. Double click on the cell to change the property values for that cell. The property values for this example are shown in the property list.

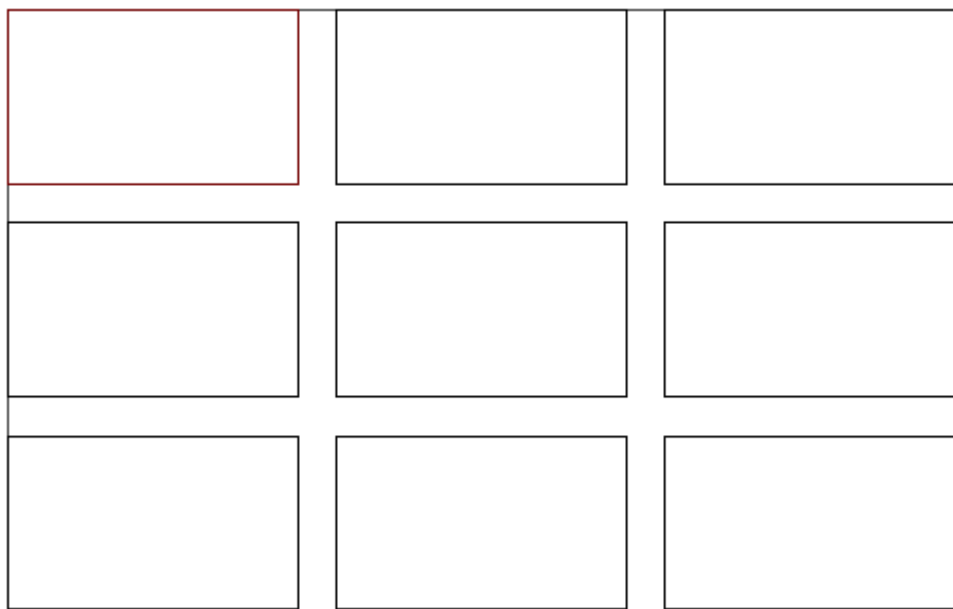
	@Equipment@
Downtime Count	@count@
Total Downtime	@total.Downtime@
AVG Downtime	@average.Downtime@

Property	Value
Style	Default
Columns	1
Header Columns	1
Rows	3
Header in Row 1	Downtime Count
Header in Row 2	Total Downtime
Header in Row 3	AVG Downtime
Header Rows Color	Blue, Bold
Header Column Color	Blue, Bold

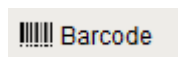
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Component - Labels



Component Palette Icon:



Description

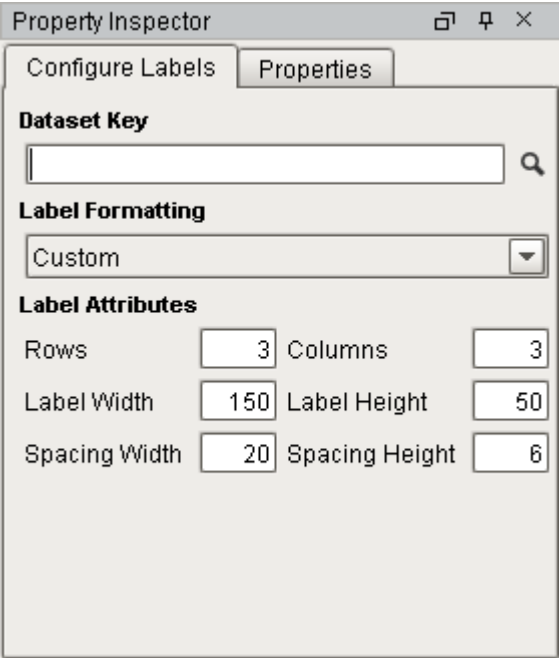
Labels can be used to print out a host of different types of labels depending on how you configure the Label properties. You can create mailing labels, name tags, or any other generic type of labels. You can use standard Avery label sheets, or specify your own dimensions. When the Labels component is selected, the label size and spacing can be set in the Labels Attributes area of the Configure Labels tab in the Property Inspector. The size of the Labels component automatically update based on the values in the Label Attributes area. There are more properties in under the Properties tab that you can set to customize your labels.

When creating labels, all you have to do is create one template label which is always the top-left label in the component. To edit the label itself, **super-select** the template label to add **Text Shapes**, **Data Keys** or **Keychain Expressions**, **Images** or **Barcodes**. After you create your template, go to the Preview Panel and the you'll see all the labels will be populated with your data. Keep in mind that any data keys or expressions within the Label component are driven by the Data Key configured in the component.

Labels Properties

In the Property Inspector, use the Configure Labels tab to configure your table. Here is a list of all the Label properties and their descriptions.

Configure Labels Tab



Property	Description
Data Key	Name of DataSet that will populate the labels.
Label Formatting	Choose from a list of Avery Label Formats, or create a custom sized label.
Rows/Columns	Defines the number of rows and columns on the page.
Label Width/Height	Width and height of labels in pixels.
Spacing Width/Height	Distance between labels on the page in pixels.

Labels Properties Tab

Property Inspector

Configure LabelsProperties

2

Labels

Dataset Key	
Columns	3
Label Height	50
Label Width	150
Rows	3
Spacing Height	6.122
Spacing Width	20

Stroke and Fill

Fill	
Fill Color	
Opacity	1
Stroke Style	Hidden
Stroke	

Basic Properties

Roll	0
Scale X	1
Scale Y	1
Visible	☒
Width	490
Height	162.245
X	56.199
Y	91.236

Property	Description	Category
Dataset Key	Dataset key for labels.	Labels
Columns	Number of label columns.	Labels
Label Height	Height of each label in pixels.	Labels
Label Width	Width of each label in pixels.	Labels
Rows	Number of label rows.	Labels
Spacing Height	Spacing between label rows.	Labels
Spacing Width	Spacing between label columns.	Labels
Fill	If true, the shape will fill its space with color.	Stroke and Fill
Fill Color	If fill is selected, the color that will fill its shape.	Stroke and Fill
Opacity	How opaque the fill color is between, 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of the shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of the shape. 1 is scale to 100%.	Basic Properties

Property	Description	Category
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of the shape in pixels.	Basic Properties
Height	Height of the shape in pixels.	Basic Properties
X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page.	Basic Properties
Y	Vertical distance in pixels between the upper edge of this shape and the top edge of the page.	Basic Properties

Text Properties

Property Inspector

Edit Text Properties

2

Text Properties

Text

Text Color

Character Spacing

Coalesce Newlines

Font

Horizontal Alignment

Line Spacing

Margin

Overflow Behavior

Underlined

Vertical Alignment

0

☐

Arial Bold Italic 18.0 (A...

Left

1

1; 2; 0; 2

Grow row

☐

Top

Data Key Format Properties

Date Format

Null Format

Number Format

Negative In Red

MMM d, yyyy

<N/A>

#0.##

☐

Stroke and Fill

Fill

Fill Color

Opacity

Stroke Style

Stroke

☐

1

Hidden

Rectangle Shape

Radius

0

Basic Properties

Roll

Scale X

Scale Y

Visible

Width

Height

X

Y

0

1

1

☒

249

83

9

33

Property	Description	Category
Text	Text to display in the shape.	Text Properties
Text Color	Color of the text.	Text Properties
Character Spacing	Amount of extra spacing to add between characters. Negative spacing makes characters closer together.	Text Properties
Coalesce Newlines	If true, consecutive line breaks will be merged into one.	Text Properties
Font	Font to use for the text.	Text Properties
Horizontal Alignment	Horizontal Alignment for text: Left; Right: Center; Full.	Text Properties
Line Spacing	Spacing between lines of text. 1 is single spacing, 2 is double spacing, etc.	Text Properties
Margin	Margin between texts and bounds of shape in pixels. Format is Top: Left: Bottom; Right.	Text Properties
Overflow Behavior	How text that overflows the bounds of the shape should be handled. • Paginate - text is pushed to the next page if the number of characters exceed the shape. • Shrink text to fit - text fits into the shape, and the font is smaller when previewed or printed. • Grow row - text fits into the shape, and shows when the shape is expanded, previewed, or printed.	Text Properties
Underlined	If true, use underlined text.	Text Properties
Vertical Alignment	Vertical alignment for text: Top; Middle; Bottom.	Text Properties
Date Format	Formats to use for dates in this shape. View available format strings in the dropdown.	Data Key and Format Properties

Property	Description	Category
Null Format	Formats to use for null values in this shape.	Data Key and Format Properties
Number Format	Formats to use for numbers in this shape. View available format strings in the dropdown.	Data Key and Format Properties
Negative in Red	If true, show negative numbers in red.	Data Key and Format Properties
Fill	If true, the shape will fill its space with color.	Stroke and Fill
Fill color	If Fill is selected, the color that will fill the shape.	Stroke and Fill
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Radius	Amount to radius the corners of this shape.	Rectangle
Roll	Amount to radius the corners of this shape.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties

Property	Description	Category
Height	Height of this shape in pixels.	Basic Properties
X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page	Basic Properties
Y	Vertical distance in pixels between the upper edge of this shape and the top edge of the page.	Basic Properties

Example

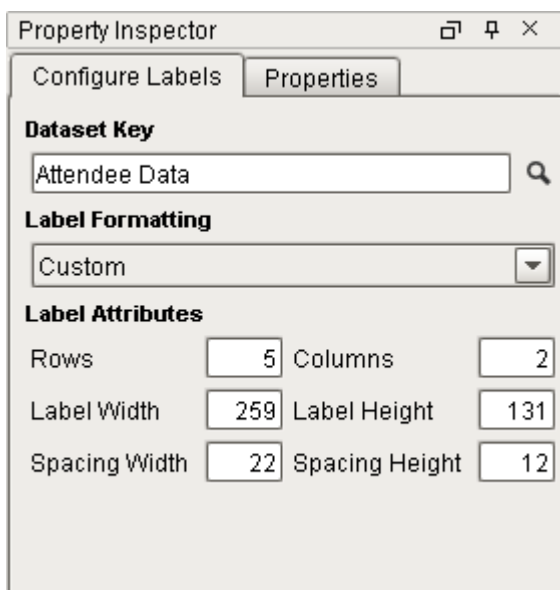
Name Tags with Logo

 Jennifer Stewart Marketing Manager ABC Company	 John Doe Communications Director Paint Company
 Sally Johnson Chief Operating Officer Pure Water	 Howard Hunter Engineer Sticker Shock
 Alice Honey Control Specialist Signs and More	 Rocky Rooney Event Planner All Occasions
 Gracie Moon Engineering Specialist Lighting Comany	 Candy Miller Designer Speciality Cards
 Jimmy Rush Marketing Specialist Vegan Foods	 Valerie Rain Landscape Designer Elegant Landscapes

Label Dataset

AttendeeName, Title, CompanyName
Jennifer Stewart, Marketing Manager, ABC Company
John Doe, Communications Director, Paint Company
Sally Johnson, Chief Operating Officer, Pure Water
Howard Hunter, Engineer, Sticker Shock
Alice Honey, Control Specialist, Signs and More
Rocky Rooney, Event Planner, All Occasions
Gracie Moon, Engineering Specialist, Lighting Comany
Candy Miller, Designer, Speciality Cards
Jimmy Rush, Marketing Specialist, Vegan Foods
Valerie Rain, Landscape Designer, Elegant Landscapes

Configure Labels Tab



Property Inspector

Configure Labels Properties

Dataset Key

Attendee Data

Label Formatting

Custom

Label Attributes

Rows	5	Columns	2
Label Width	259	Label Height	131
Spacing Width	22	Spacing Height	12

Property	Value
Dataset Key	Attendee Data
Label Formatting	Custom
Rows	5
Columns	2
Label Width	2 pixels
Label Height	131 pixels
Spacing Width	22 pixels
Spacing Height	12 pixels

Label Template

Property Inspector

Edit Text

Properties

@AttendeeName@

@Title@

@CompanyName@

T

B

I

U

x²

x₂



Property	Value
Image	Image Management > Builtin/icons/24/user3.png
Text	AttendeeName - Bold and Italic
Font	18

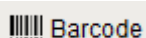
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Component - Barcode



Component Palette Icon:



Description

The Barcode component allows reports to contain dynamically generated barcodes based on how you configure its properties. Barcodes are commonly embedded into unstructured Table Rows, or **Labels** for printing. When used in set-driven components such as these, each barcode can be encoded to a unique value provided by the driving data source.

There are several types of barcodes formats that serve different uses and industries, and fall into two categories: Standard Barcodes and 2D Barcodes. Standard barcodes represent data by varying the widths and spacings of parallel lines. 2D Barcodes represent data using two-dimensional symbols and shapes, and hold more data than Standard barcodes. Each Barcode Format has specific requirements for what it will encode. The tables below show the different formats that are supported for each category, and provide a brief description including some of the industries they are use in. If you are unsure which code you should be using, please consult your existing barcode documentation.

Barcode Formats

Standard Barcode Formats



Barcode Format	Description	Industry Used
Codabar	Also known as Ames code, encodes a variable length code that supports a limited character set of the following 16 characters, digits 0-9, and symbols -.:\$/. It's easy to print and can be produced by any impact style printer.	Logistics, Healthcare, Education
Code 39	Encodes a variable length barcode with a limited number, letter and special character set, including capital letters A-Z, digits 0-9, symbols -.\$/%, and a space character. The asterisk (*) must be used as a start and stop delimiter. They are used to label goods across many industries.	Automotive, Defense
Code 128	Encodes a variable length barcode consisting of any of the 128 ASCII character set. They are powerful and can store diversified information.	Supply Chain
EAN_8	Encodes an 8 digit value used to label consumer goods primarily in Europe. Only used when limited space is available.	Retail
EAN_13	Encodes 13 digit value, used often as a product identification numbers to label consumer goods primarily in Europe.	Retail
ITF	Encodes numeric digits as pairs, must be supplied an even number of digits. Used to label packaging materials and are good for printing on corrugated cardboard.	Packaging
UPC_A	Is the most common form of UPC code consisting of 12 total digits: 11 digits and a checksum. Used to label consumer goods.	Retail

2D Barcode Formats



Barcode Format	Description	Industry Used
Aztec	Encodes up to 3832 digits, 3067 letters or 1914 bytes of data. Used by the transportation system for tickets and airline boarding passes.	Transportation
Data_Matrix	Variable length and generate 2D barcodes with advanced encoding error checking and correction. Used to label small items, and documents	Electronics, Retail, Government
PDF417	Specialized linear scan barcode that can encode alphanumeric test. Used in applications that require storage of huge amounts of data.	Logistics, Government
QR_Code	Can generate an image encoding a string of text up to 4,296 alphanumeric characters in length. Supports four different modes of data: numeric, alphanumeric, byte/binary, and Kanji.	Retail, Entertainment, Advertising

Properties

Configure Barcode Tab

Property Inspector

Configure Barcode

Properties

Datakey

Barcode Format

CODE_39

☐ Show Text

Barcode Format Info

Code 39 encodes a variable length barcode with a limited number, letter and special character set, including capital letters A-Z, digits 0-9, and symbols -. \$/ % and space character. * must be used as a start and stop delimiter.

Property	Description
Data Key	Key that will populate the barcode.
Barcode Format	Barcode format to display.
Show Text	Determines if the value of the Data Key should appear on the barcode.
Font	Enabling the Show Text option allows the Font configuration to be edited for the displayed text.
Barcode Format Info	Describes the selected barcode format.

Properties Tab

Property Inspector

Configure Barcode

Properties

2

Barcode

Datakey	
Barcode Format	CODE_39
Font	Monospaced, Plain, 12
Show Text	☐

Stroke and Fill

Opacity	1
Stroke Style	Hidden
Stroke	

Basic Properties

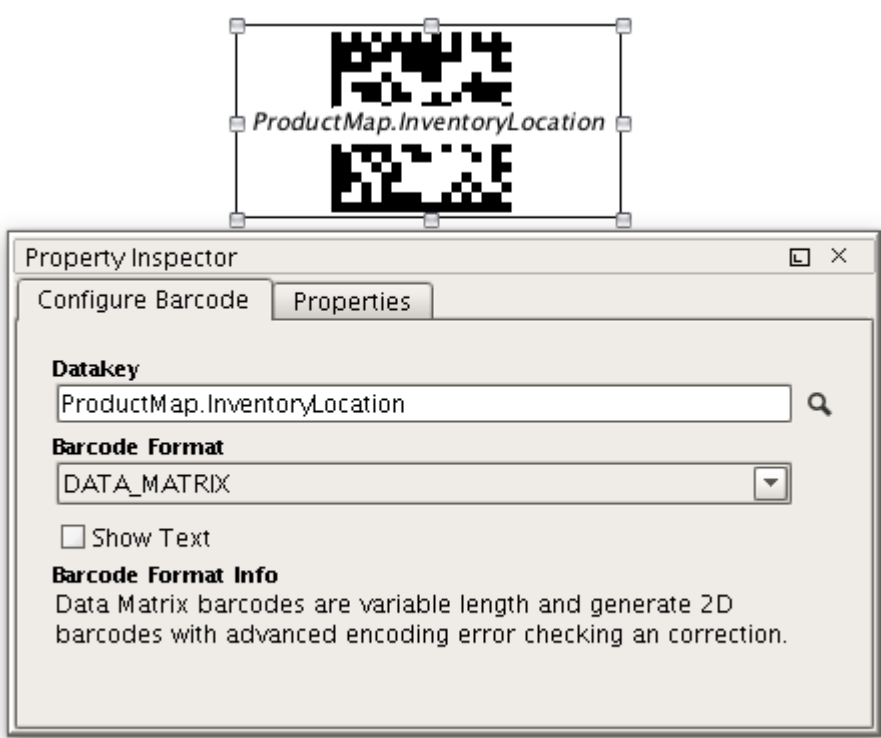
Roll	0
Scale X	1
Scale Y	1
Visible	☒
Width	154.701
Height	72.492
X	113.299
Y	160.508

Property	Description	Category
Datakey	Datakey for barcode.	Barcode
Barcode Format	Barcode format to use, see options above.	Barcode
Font	Font type, size, and style for barcode.	Barcode
Show Text	If true, show barcode value as text. Not available for all barcode types.	Barcode
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties
Height	Height of this shape in pixels.	Basic Properties

Property	Description	Category
X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page.	Basic Properties
Y	Vertical distance in pixels between the upper edge of this shape and the top edge of the page.	Basic Properties

Examples

2D Data_Matrix Barcode



Property	Value
Datakey	ProductMap.InventoryLocation
Barcode Format	Data_Matrix

To add a barcode to a report, simply drag the Barcode component from the Report Design Palette on to your Design panel workspace. You will see a placeholder barcode with its driving data key displayed, and a description of the encoding for the type specified in the **Barcode Format** field of the Configure Barcode tab. Data Keys can be dropped or typed into the **DataKey** field, and the driving

key will be shown in its appropriate barcode placeholder. The actual barcode, as well as any text to be displayed under the barcode, is generated and rendered when the report is executed or previewed.

Barcodes require a **Data Key** or Text to encode.

i NOTE

Each **Barcode Format** encoder has specific requirements for what it will encode. **Barcode Format Info** displays some brief information about the selected format, but is not comprehensive.

Embedded Barcode

To see an example of embedding a barcode in a DataSource driven component, refer to the [Labels with Embedded Barcodes](#) page. The example embeds a barcode in the repeating Labels component to dynamically create coded labels to track items.

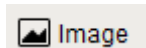
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Component - Images



Component Palette Icon:



Description

The Image component lets you [embed images into reports](#). Images can be dragged from a local computer, shared drive, webpage, or the Image Management Tool as files onto reports. They can also be pulled as binary data from a datasource. If no **Key** is associated with the image component, the dropped image will be used. If the **Key** does not link to a valid image, no image will be shown. The image displayed in the Report Designer is a placeholder. It will be replaced with the image associated with the image's **Key** when the report is generated or previewed.

In the image component's **Property Inspector**, you can drag a [parameter](#) or a [datasource](#) column from the **Key Browser** into the **Key** field. The Data Key can resolve to a byte array, such as an image retrieved from a database, or a URL that points to an image on the web, or an image file on a disk. Remember that reports are generated on the Gateway, so the image source needs to be accessible from the Gateway computer.

[Keychain Expressions](#) can not be used in the data key, but the Key can be a parameter or datasource that dynamically constructs a path to an image. This allows you to easily change images in reports without changing the image component.

Properties

Property Inspector	
2	
Stroke and Fill	
Fill	
Fill Color	
Opacity	1
Stroke Style	Hidden
Stroke	
Image Shape	
Grow To Fit	☒
Key	
Padding	0
Page Index	0
Preserve Aspect Ratio	☒
Rectangle Shape	
Radius	0
Basic Properties	
Roll	0
Scale X	1
Scale Y	1
Visible	☒
Width	326
Height	140
X	111
Y	56

Property	Description	Category
Fill	If true, the shape will fill its space with color.	Stroke and Fill
Fill Selected	If fill is selected, the color that will fill the shape.	Stroke and Fill
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Grow to Fit	If true, rescale the original image to fit the shape.	Image Shape
Key	Data Key for this image.	Image Shape
Padding	Padding between the image and the shape's bounds.	Image Shape
Page Index	For multi-page images, sets the page index to use.	Image Shape
Preserve Aspect Ratio	If true, preserve the aspect ratio when resizing the image.	Image Shape
Radius	The amount to radius the corners of this shape.	Rectangle Shape
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties

Property	Description	Category
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties
Height	Height of this shape in pixels.	Basic Properties
X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page.	Basic Properties
Y	Vertical distance in pixels between the top edge of this shape and the top edge of the page.	Basic Properties

Example

Image in a Report



Squash	10
Zucchini	18
Carrots	25
Potatoes	8
Beans	33
Peppers	9

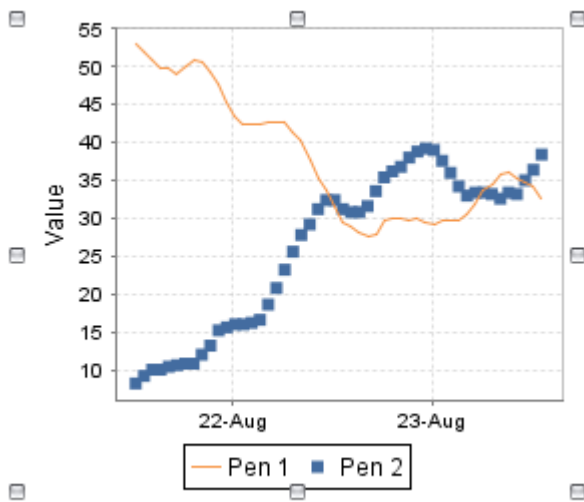
Property	Value
Data Key	Image Path

The image in this report was created using a **Parameter** that resolved to a URL.

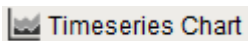
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Component - Timeseries Chart



Component Palette Icon:



Description



INDUCTIVE
UNIVERSITY

Timeseries Chart

[Watch the video](#)

The **Timeseries Chart**, while similar to the **XY Chart**, uses a datetime value for the Domain Axis instead of a numerical value. This chart is ideal when trying to display **Tag Historian** data in a Report.

Properties

Chart Options

Property Inspector

Chart Options

Properties

Data Key

Domain Key

t_stamp

Pens

Range Key	Pen Name	Preview
Test1	Pen 1	
Test2	Pen 2	

Axes

Name	Label	Auto?	Low	High
Default Axis	Value	☒	0	100

Scripting

☐ Enabled ☒ Edit Script

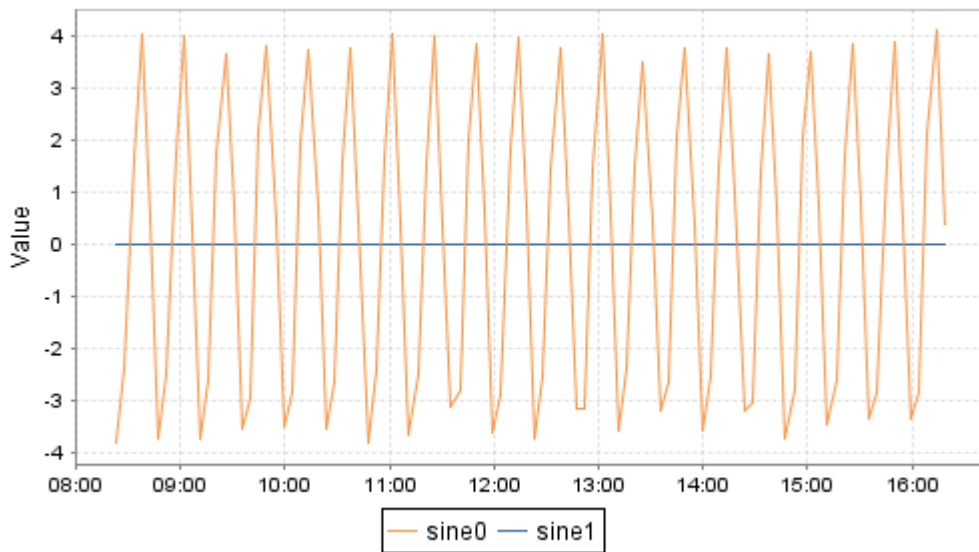
Property/Symbol	Description
Data Key	Unique identifier or placeholder to the data source that will populate the chart.
Domain Key	The data key that should represent the X axis on the chart. The Timeseries Chart expects a timestamp, while the XY chart can use any numeric value as the domain key.
Pens	Pens represent a series of data. There are several properties for each pen exposed on the Chart Options tab: Range Key: The key the pen should retrieve data from. Pen Name: The name of the pen that should appear on the chart. Preview: Visual representation of how the pen will be rendered. The trend is simply a preview, and does not take the data from the range key into account.
Add Pen 	Adds a pen.
Edit Pen 	Opens Pen Configuration area for editing.
Remove Pen 	Deletes a pen.
Axes	There are several properties for each axis exposed on the Chart Options tab: Name: The name of the axis. Label: Text next to the axis. Commonly used to describe what the values of each pen represent. Auto?: Determines if the axis should automatically adjust the upper and lower bound of the axis to include all data from each pen. When enabled, the High and Low properties are ignored. When disabled, High and Low will be used. Low: The low range value the axis will use when 'Auto?' is disabled. High: The high range value the axis will use when 'Auto?' is disabled.
Add Axis 	Add an axis.
Edit Axis 	Opens Axis Configuration area for editing.

Property	Description	Category
Data Key	Data key of the dataset that drives this chart.	Chart Options
Axis Label Font	Font to use for axis labels.	Chart Options
Axis Tick Font	Font to use for axis tick marks.	Chart Options
Bar Width	The width of any bar datapoints, specified in milliseconds.	Chart Options
Date Format	Optional format string to override the domain axis date format.	Chart Options
Domain Key	Key name of the column inside the dataset that contains the domain (x) value.	Chart Options
Gap Threshold	Multiplier for the average space between data points to be considered a data gap for the "line with gaps" render style.	Chart Options
No Data Msg	Message to show when no data is available.	Chart Options
Plot Background	Color to use for chart background.	Chart Options
Show Legend	If true, show a legend for the chart.	Chart Options
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill

Property	Description	Category
Stroke	Details for the chosen stroke. Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties
Height	Height of this shape in pixels.	Basic Properties
Show X-Axis	If true, show the x-axis.	X-Axis
X-Axis Label	Label text for the x-axis.	X-Axis

Examples

Example 1



Property Inspector

Chart Options Properties

Data Key tag_history **Domain Key** t_stamp

Pens

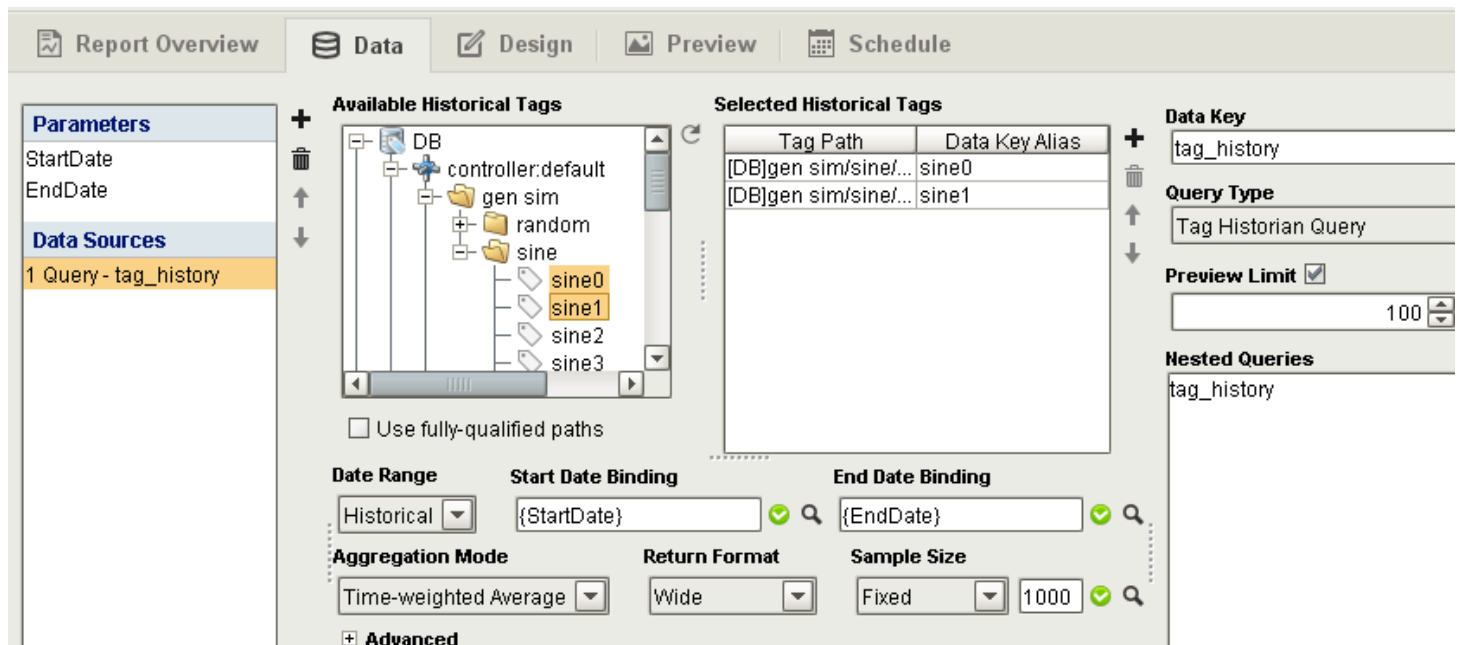
Range Key	Pen Name	Preview
sine0	sine0	
sine1	sine1	

Axes

Name	Label	Auto?	Low	High
Default Axis	Value	☒	0	100

Scripting

☐ Enabled ☒ Edit Script



Example 2

Dynamically set the range on your chart in the report. This script includes a minimum and maximum boundary so you can set operating ranges. It will throw out values that don't land in the correct range. IE: You don't want to display oven temperatures below 500 degrees, or values of 0 represent bad data.

This code should be placed in the `configureChart` function definition of your chart. You can access it by clicking the **Edit Script** button at the bottom of the Property Inspector.

```
configureChart
```

```
# set boundaries to throw out 'bad' values that land outside this range
minBoundary = 50
maxBoundary = 100

from org.jfree.data.general import DatasetUtilities
# get the chart, axis, data
plot = chart.getPlot()
rangeAxis = plot.getRangeAxis()
rangeAxis.setAutoRange(False)
ds = plot.dataset

# Find the initial range of the dataset
rangeBounds = DatasetUtilities.findRangeBounds(ds)
rangeMin = rangeBounds.getLowerBound()
rangeMax = rangeBounds.getUpperBound()
# working minimum and maximum values for dataset. these are swapped with the
initial values on purpose
currentMin = rangeMax
currentMax = rangeMin

# Loop through the dataset to find desired min and max
series = 0 # which pen to search
numItems = ds.getItemCount(series)
for i in range(numItems):
    val = ds.getYValue(series,i)
    # find lowest value that is above the minBoundary
    if val > minBoundary and val < currentMin:
        currentMin = val
    # find highest value that is below the maxBoundary
    if val < maxBoundary and val > currentMax:
        currentMax = val

# calculate 10% padding
padding = (currentMax - currentMin) * 0.1

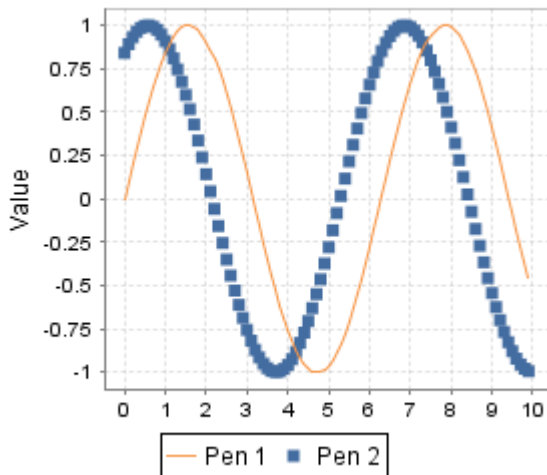
# set bounds in chart
rangeAxis.setLowerBound(currentMin - padding)
rangeAxis.setUpperBound(currentMax + padding)

# for troubleshooting: look in the console after going to the preview pane to see
real values
# comment these lines when done with testing
print "Min: ", currentMin
print "Max: ", currentMax
print "Padded Min: ", currentMin - padding
print "Padded Max: ", currentMax + padding
```

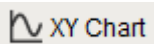
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Component - XY Chart



Component Palette Icon:



Description



INDUCTIVE
UNIVERSITY

XY Chart

[Watch the video](#)

The **XY Chart**, while similar to the **Timeseries Chart**, uses a numerical value for the Domain Axis instead of a Datetime. The XY chart is ideal when trying to detail the relationship between two keys.

Properties

Chart Options

Property Inspector

Chart Options

Properties

Data Key

Domain Key

x

Pens

Range Key	Pen Name	Preview
Test1	Pen 1	
Test2	Pen 2	

Axes

Name	Label	Auto?	Low	High
Default Axis	Value	☒	0	100

Scripting

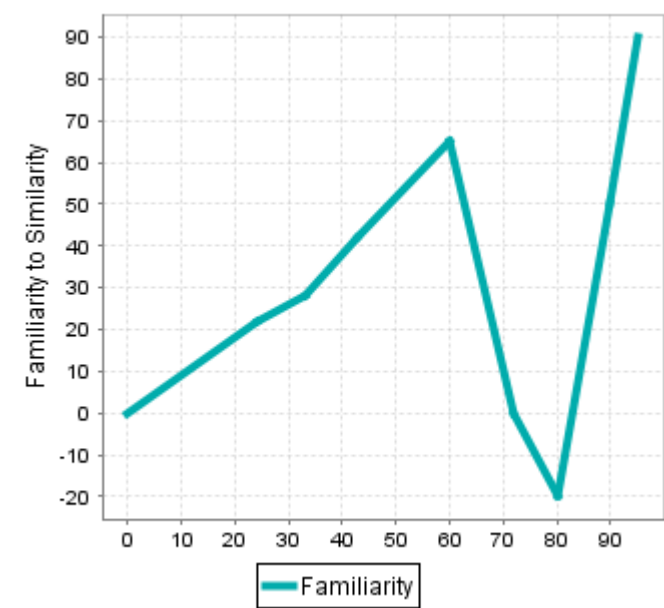
☐ Enabled ☒ Edit Script

Property/Symbol	Description
Data Key	Unique identifier or placeholder to the data source that will populate the chart.
Domain Key	The data key that should represent the X axis on the chart. The Timeseries Chart expects a timestamp, while the XY chart can use any numeric value as the domain key.
Pens	Pens represent a series of data. There are several properties for each pen exposed on the Chart Options tab: Range Key: The key the pen should retrieve data from. Pen Name: The name of the pen that should appear on the chart. Preview: Visual representation of how the pen will be rendered. The trend is simply a preview, and does not take the data from the range key into account.
Add Pen 	Adds a pen.
Edit Pen 	Opens Pen Configuration area for editing.
Remove Pen 	Deletes a pen.
Axes	There are several properties for each axis exposed on the Chart Options tab: Name: The name of the axis. Label: Text next to the axis. Commonly used to describe what the values of each pen represent. Auto?: Determines if the axis should automatically adjust the upper and lower bound of the axis to include all data from each pen. When enabled, the High and Low properties are ignored. When disabled, High and Low will be used. Low: The low range value the axis will use when 'Auto?' is disabled. High: The high range value the axis will use when 'Auto?' is disabled.
Add Axis 	Add an axis.
Edit Axis 	Opens Axis Configuration area for editing.

Property	Description	Category
Data Key	Data key of the dataset that drives this chart.	Chart Options
Axis Label Font	Font to use for axis labels.	Chart Options
Axis Tick Font	Font to use for axis tick marks.	Chart Options
Bar Width	The width of any bar datapoints.	Chart Options
Domain Key	Key name of the column inside the dataset that contains the domain (x) value.	Chart Options
Gap Threshold	Multiplier for the average space between data points to be considered a data gap for the "line with gaps" render style.	Chart Options
No Data Msg	Message to show when no data is available.	Chart Options
Plot Background	Color to use for chart background.	Chart Options
Show Legend	If true, show a legend for the chart.	Chart Options
Auto Range	If true, automatically determine range of X-axis.	X-Axis
Axis High	If autorange is turned off, the high value to use for the axis.	X-Axis
Axis Low	If autorange is turned off, the low value to use for the axis.	X-Axis
Margin	Margin for the x-axis edges.	X-Axis
Number Format	A optional number format string to override the automatic formatting on the x-axis.	X-Axis

Property	Description	Category
Show X-Axis	If true, show the x-axis.	X-Axis
X-Axis Label	Label text for the x-axis.	X-Axis
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties
Height	Height of this shape in pixels.	Basic Properties
X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page.	Basic Properties
Y	Vertical distance in pixels between the upper edge of this shape and the top edge of the page.	Basic Properties

Example



Property Inspector

Chart Options Properties

Data Keyuncanny_valley

Domain KeyHuman_Similarity

Pens

Range Key	Pen Name	Preview
Familiarity	Familiarity	

Axes

Name	Label	Auto?	Low	High
Default Axis	Familiarity to S...	☒	0	100

Scripting

☐ Enabled ☒ Edit Script

XY Chart Sample CSV Data - uncanny_valley

Familiarity, Human_Similarity

0, 0

22, 24

28, 33

42, 43

65, 60

0, 72

-20, 80

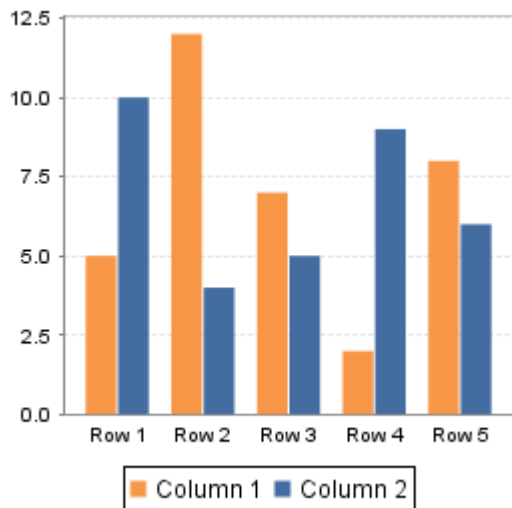
50, 90

90, 95

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Component - Bar Chart



Component Palette Icon:



Description



INDUCTIVE
UNIVERSITY

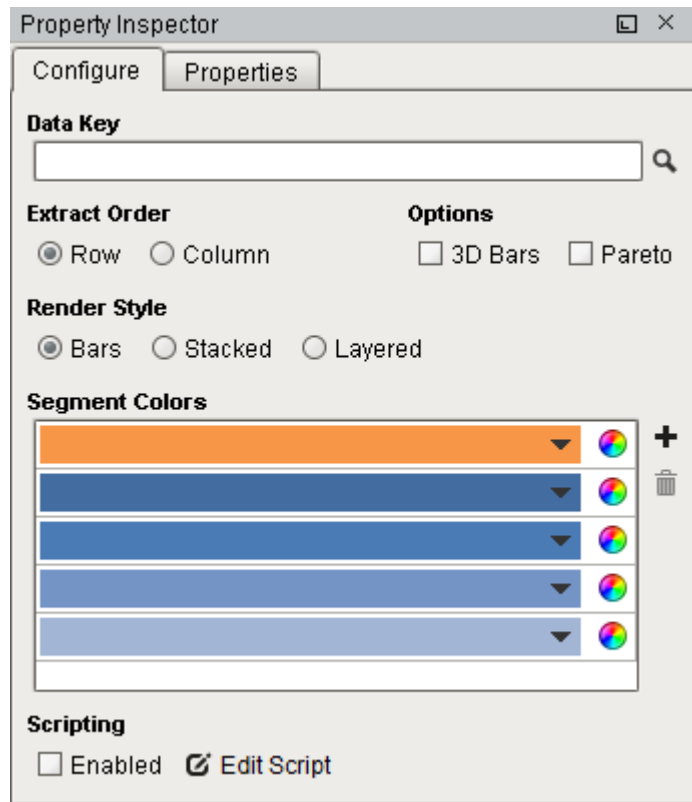
Bar Chart

[Watch the video](#)

A **Bar Chart** is used to display data in a chart using bars. A Bar Chart typically contains a single string column that gets used as the Domain with as many other columns representing the Range axis values. The order of the dataset matters, and the string values used for the domain should be in the first column of the dataset.

Properties

Configure Tab



Property/Symbol	Description
Data Key	Unique identifier or placeholder to the data source that will populate the chart.
Extract Order	The order in which the data should be extracted from the data key. Two options are available: Row: Data is extracted by row. Use this when columns in the data key define the series of data. Column: data is extracted by column. Use this when rows in the data key define the series of data; each column is a new value for the same series. This is sometimes referred to as data 'turned on its side'
Options	3D: Columns and bars with 3D appearance using angle and depth. Pareto: When enabled, turns the chart into a Pareto chart. Contains both lines and a bar graph where values are in descending order.
Render Style	Provides three rendering styles: Bars: Displays data as individual colored bars. Stacked: Displays data as a stacked bar in different colors. Layered: Displays data as a layered bar in different colors.
Segment Colors	A unique color in the RGBA color space for each segment (bar).
Color Wheel 	Tool for creating color combinations for bars.
Add Segment 	Adds a color / row.
Remove Segment 	Deletes a color / row.
Scripting Enabled	Enables the script for this component to run.
Edit Script	Allows the chart to be modified before the report is rendered.

Properties Tab

Property Inspector

Configure

Properties

2

Chart Options

Data Key

Category Margin

Extract Order

Item Margin

Render Style

3D Bars

Show Legend

Legend Font

Vertical Chart

0.2

Row

0.02

Bars

☐

☒

Dialog, Plain, 12

☒

Pareto Options

Pareto

80% Line

Pareto Color

Pareto Series Label

☐

☒

Percent

Axis Options

Axis Autorange

Axis Font

Axis Format

Axis Label

Axis Low

Axis High

Label Position

☒

Dialog, Plain, 12

0

100

Standard

Label Options

Bar Labels

Label Color

Label Font

Label Offset

☐

Dialog, Plain, 12

2

Stroke and Fill

Opacity

Stroke Style

Stroke

1

Hidden

Basic Properties

Roll

Scale X

Scale Y

Visible

Width

Height

X

Y

0

1

1

☒

261

253

74

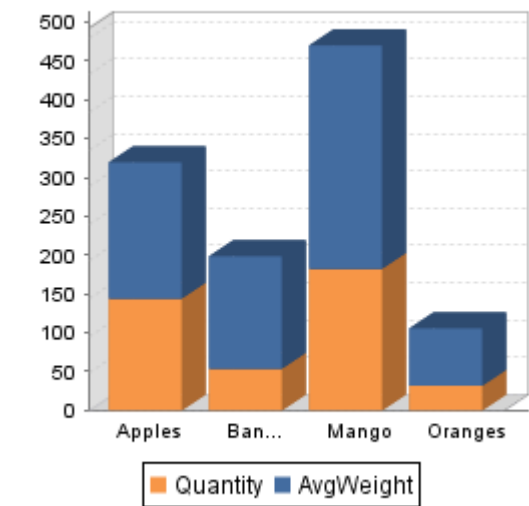
244

Property	Description	Category
Data Key	Data key of the dataset that drives this bar chart.	Chart Options
Category Margin	Spacing between groups of bars on the chart.	Chart Options
Extract Order	Determines whether the series are defined by rows or columns in the underlying dataset.	Chart Options
Item Margin	Spacing between bars within a group.	Chart Options
Render Style	Style for this bar chart. Options are bars (0), stacked (1), or layered (2).	Chart Options
3D Bars	If true, use 3D bars to draw chart.	Chart Options
Show Legend	If true, show the chart's legend.	Chart Options
Legend Font	Font to use for the chart's legend.	Chart Options
Vertical Chart	If true, draw bars vertically.	Chart Options
Pareto	If enabled, the chart will act as a pareto chart; ordering the bars descending and adding a percentage line.	Pareto Options
80% Line	If true, draw a line showing where 80% of the total is.	Pareto Options
Pareto Color	Color for the Pareto line and 80% line.	Pareto Options
Pareto Series Label	Text to use in legend for the Pareto line.	Pareto Options

Property	Description	Category
Axis Autorange	If true, set the axis range automatically, otherwise use the Axis Low and Axis High values.	Axis Options
Axis Font	Font to use for axis and axis label.	Axis Options
AxisFormat	A optional number format string to override the automatic formatting on the y-axis.	Axis Options
Axis Label	Text to place on the axis label. May be blank.	Axis Options
Axis Low	If autorange is turned of, the low value to use for the axis.	Axis Options
Axis High	If autorange is turned off, the high value to use for the axis.	Axis Options
Label Position	Orientation of category labels.	Axis Options
Bar Labels	If true, add labels to each bar.	Label Options
Label Color	Color to use for bar label text.	Label Options
Label Font	Font to use for bar labels.	Label Options
Label Offset	Space to add between the end of the bar and the label. Negative numbers will put the label onto the bar.	Label Options
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill

Property	Description	Category
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties
Height	Height of this shape in pixels.	Basic Properties
X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page.	Basic Properties
Y	Vertical distance in pixels between the top edge of this shape and the top edge of the page.	Basic Properties

Example



Property Inspector

Configure Properties

Data Key

fruit_data

Extract Order

☒ Row ☐ Column

Options

☒ 3D Bars ☐ Pareto

Render Style

☐ Bars ☒ Stacked ☐ Layered

Segment Colors

Scripting

☐ Enabled ☒ Edit Script

Property	Value
3D Bars	True

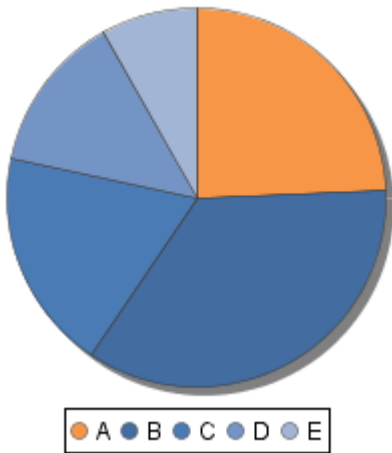
Bar Chart Sample CSV Data

```
Fruit, Quantity, AvgWeight
Apples, 150, 183
Bananas, 56, 151
Mango, 190, 300
Oranges, 33, 77
```

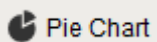
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Component - Pie Chart



Component Palette Icon:



Description



INDUCTIVE
UNIVERSITY

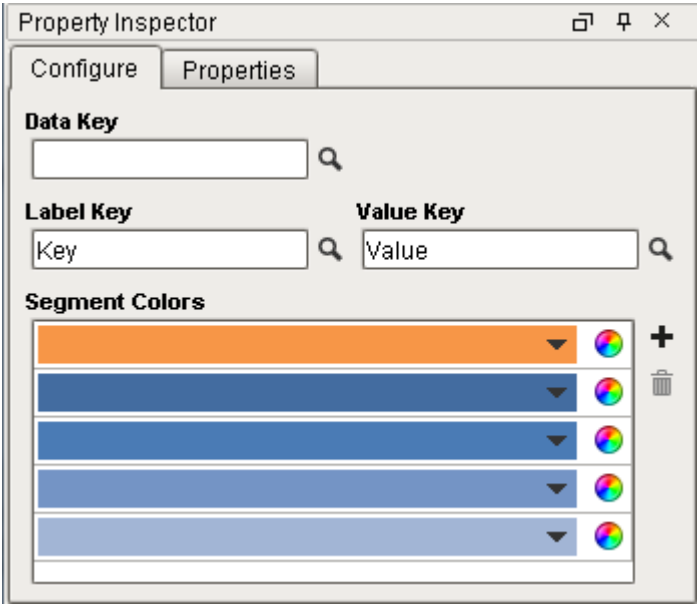
Pie Chart

[Watch the video](#)

The **Pie Chart** takes a Data Source with two keys: a Label (typically a string) and a Value (typical a numerical value).

Properties

Configure Tab



Property	Description
Data Key	Unique identifier or placeholder to the data source that will populate the chart.
Label Key	Key that will list the names of each segment in the legend on the chart.
Value Key	The key that will populate the chart with data.
Segment Colors	A unique color in the RGBA color space for each segment (wedge). Will repeat the colors if there are not enough to cover all of the different segments, so it is important to list a lot of colors here.
Add Segment 	Adds a color wedge to the chart.
Remove Segment 	Deletes a color wedge from the chart.

Properties Tab

Property Inspector

Configure

Properties

2

Pie Options

Data Key	fruit_data
Label Key	Fruit
Label Style	Outset
Label Format	{0} = {1} ({2})
Label Font	Dialog, Plain, 12
Legend Font	Dialog, Plain, 12
Section Outline	
Show Legend	☒
Sort Order	None
Style	Pie
Value Key	Quantity

Stroke and Fill

Opacity	1
Stroke Style	Hidden
Stroke	

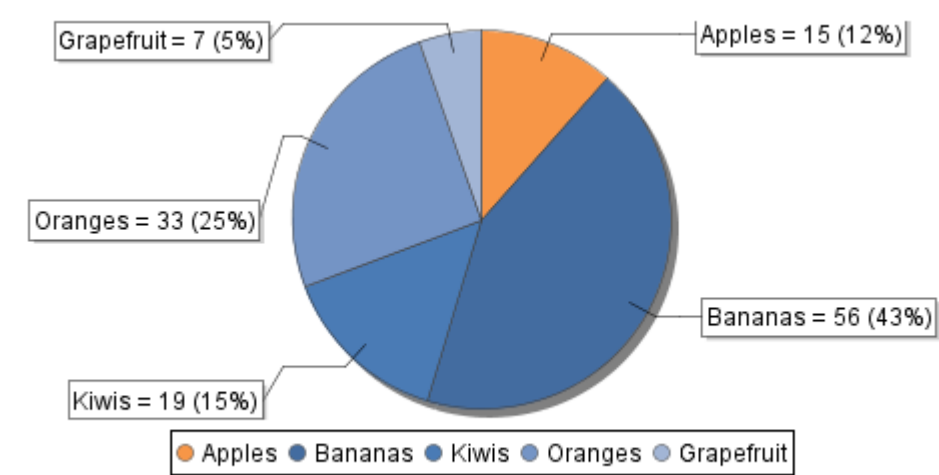
Basic Properties

Roll	0
Scale X	1
Scale Y	1
Visible	☒
Width	483
Height	232
X	36
Y	290

Property	Description	Category
Data Key	Data key of the dataset that drives this pie chart.	Pie Options
Label Key	Column name inside the dataset represented by the Data Key which holds the pie wedge's label.	Pie Options
Label Style	Style for labels on the pie chart. Options are None (0), Simple (1), and Outset (2).	Pie Options
Label Format	Format of labels, if enabled. Use 0 = wedge name, 1 = value, 2 = percentage.	Pie Options
Label Font	Font used for this chart's labels.	Pie Options
Legend Font	Font used for this chart's legend.	Pie Options
Section Outline	Color to use for outline dividing sections.	Pie Options
Show Legend	If true, show the chart's legend.	Pie Options
Sort Order	Order to sort chart categories. This will determine colors used and order of the legend.	Pie Options
Style	Style for this pie chart. Options are Pie, 3D Pie, and Ring.	Pie Options
Value Key	Column name inside the dataset represented by the Data Key which holds the pie wedge's value.	Pie Options
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill

Property	Description	Category
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties
Height	Height of this shape in pixels.	Basic Properties
X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page.	Basic Properties
Y	Vertical distance in pixels between the top edge of this shape and the top edge of the page.	Basic Properties

Example



Property Inspector

Configure

Properties

Data Key

fruit_data

Label Key

Fruit

Value Key

Quantity

Segment Colors

Property	Value
Label Style	Outset

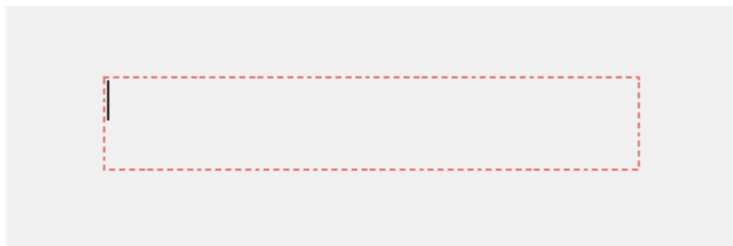
Pie Chart Sample CSV Data

```
Fruit, Quantity
Apples, 15
Bananas, 56
Kiwis,19
Oranges, 33
Grapefruit, 7
```

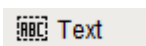

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Component - Text Shape



Component Palette Icon:



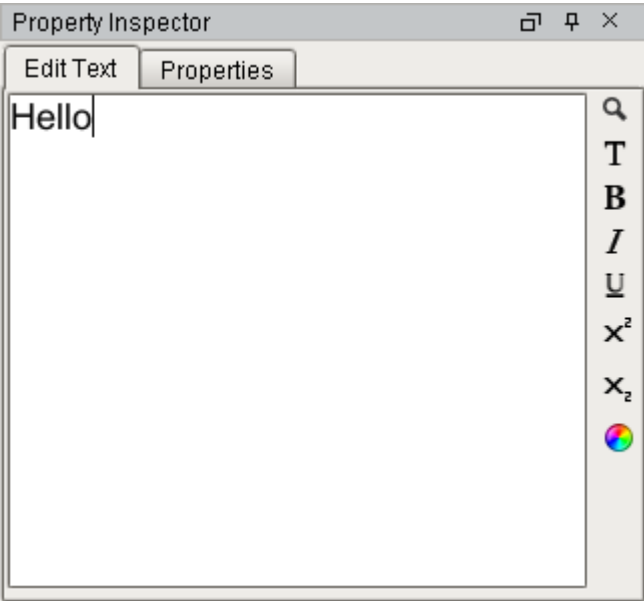
Description

The **Text shape** component creates a text area for static or data key bound content. It is used for report titles, customized page headers and footers, and of course, any additional text you want to add to a report. Select the Text Shape on the Report Component Palette to make it active, draw a rectangle in your Report Designer, and begin entering text. After it is created, you can move the Text component around, expand it, or shrink it by using the handles when the component is selected. The Text Shape component has two tabs; the **Edit Text** and **Properties** tabs.

The **Edit Text** tab has a set of functions on the right side of the tab to quickly edit the text, like changing the font type, size, style, and color. These same properties may also be modified from the Properties tab. Properties that are modified will usually affect all text in that object regardless of specific text selection. The **Properties** tab has a lot of properties associated with Text Shape component, including Data Key Format Properties for Date and Number formats by simply choosing a format from the list of available templates.

Properties

Edit Text Tab



Icon	Description
	Insert data key
	Select font and size
	Bold
	Italic
	Underline
	Superscript
	Subscript
	Choose font color

Properties Tab

Property Inspector

Edit Text Properties

2

Text Properties

Text

Text Color

Character Spacing

0

Coalesce Newlines

☐

Font

Arial 18.0 (Arial)

Horizontal Alignment

Left

Line Spacing

1

Margin

1; 2; 0; 2

Overflow Behavior

Grow row

Underlined

☐

Vertical Alignment

Top

Data Key Format Properties

Date Format

MMM d, yyyy

Null Format

<N/A>

Number Format

#0.##

%

Negative In Red

☐

Stroke and Fill

Fill

☐

Fill Color

Opacity

1

Stroke Style

Hidden

Stroke

Rectangle Shape

Radius

0

Basic Properties

Roll

0

Scale X

1

Scale Y

1

Visible

☒

Width

277

Height

47

X

145

Y

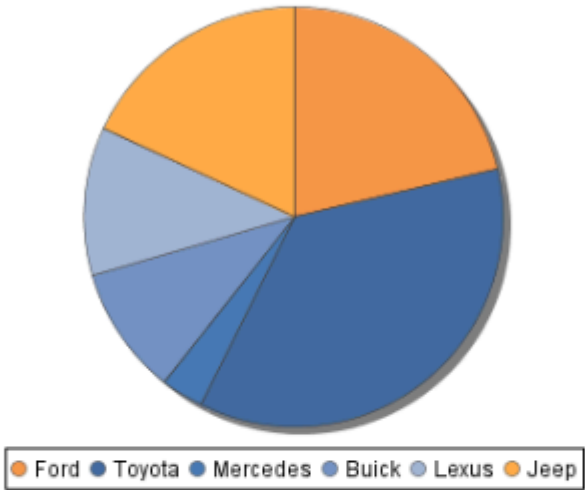
465

Property	Description	Category
Text	Text to display in the shape.	Text Properties
Text Color	Color of the text.	Text Properties
Character Spacing	Amount of extra spacing to add between characters. Negative spacing makes characters closer together.	Text Properties
Coalesce Newlines	If true, consecutive line breaks will be merged into one.	Text Properties
Font	Font to use for the text.	Text Properties
Horizontal Alignment	Horizontal alignment for text.	Text Properties
Line Spacing	Spacing between lines of text. 1 is single spacing, 2 is double spacing, etc.	Text Properties
Margins	Margin between text and bounds of shape in pixels. Format is top; left; bottom; right.	Text Properties
Overflow Behavior	How text that overflows the bounds of the shape should be handled. • Paginate - text is pushed to the next page if the number of characters exceed the shape. • Shrink text to fit - text fits into the shape when expanded, and the font is smaller when previewed or printed. • Grow row - text fits into the shape and shows when the shape is expanded, previewed, or printed.	Text Properties
Underlined	If true, use underlined text.	Text Properties
Vertical Alignment	Vertical alignment for text.	Text Properties
Date Format	Format to use for dates in this shape.	Data Key Format Properties

Property	Description	Category
Null Format	Format to use for null values in this shape.	Data Key Format Properties
Number Format	Format to use for numbers in this shape.	Data Key Format Properties
Negative in Red	If true, show negative numbers in red.	Data Key Format Properties
Fill	If true, the shape will fill its space with color.	Stroke and Fill
Fill Color	If Fill is selected, the color that will fill the shape.	Stroke and Fill
Opacity	How opaque the Fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Radius	Amount to radius the corners of this shape if using a Stroke.	Rectangle Shape
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties

Property	Description	Category
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties
Height	Height of this shape in pixels.	Basic Properties
X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page.	Basic Properties
Y	Vertical distance in pixels between the upper edge of this shape and the top edge of the page.	Basic Properties

Used Car Inventory Report
August 2017



Make	Quantity
Ford	13
Toyota	22
Mercedes	2
Buick	6
Lexus	7
Jeep	11

Text Shape - Used Car Inventory Report

Property Inspector

Edit Text

Properties

2

Text Properties

Text

Used Car Inventory Rep...

Text Color

Character Spacing

0

Coalesce Newlines

☐

Font

Arial Bold 24.0 (Arial)

Horizontal Alignment

Left

Line Spacing

1

Margin

1; 2; 0; 2

Overflow Behavior

Shrink text to fit

Underlined

☐

Vertical Alignment

Top

Used Car Inventory Report

Property	Value
Text	Used Car Inventory Report
Text Color	Blue, Bold
Font Size	24 pixels

Text Shape - August 2017

Property Inspector

Edit Text

Properties

2

Text Properties

Text

August 2017

Text Color

Character Spacing

0

Coalesce Newlines

☐

Font

Arial 18.0 (Arial)

Horizontal Alignment

Left

Line Spacing

1

Margin

1; 2; 0; 2

Overflow Behavior

Grow row

Underlined

☐

Vertical Alignment

Top

August 2017

Property	Value
Text	August 2017
Text Color	Blue
Font Size	18 pixels

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Component - Line Shape



Component Palette Icon:



Description

The Line shape component can be used to display a straight line any where in a report. It can run north-south, east-west, or diagonally. Select the Line shape on the Report Component Palette to make it active, and click and drag to draw a Line any where in your Report Designer. Hold shift while dragging to force the angle of the line to a 45 or 90 degree angle. You can change the stroke style, color, width, pattern, and more.

Properties

Property Inspector

Stroke and Fill		
Fill		
Fill Color		
Opacity		1
Stroke Style	Shape Outline	
Stroke		
Color		
Dash Pattern		
Width		1
Basic Properties		
Roll		0
Scale X		1
Scale Y		1
Visible	☒	
X		222
Y		103
Line Properties		
Stroke Color		
Stroke Width		1

Property	Description	Category
Fill	If true, the shape will fill its space with color.	Stroke and Fill
Fill Color	If true is selected, the color that will fill the shape.	Stroke and Fill
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties
Visible	If true, the shape will be visible.	Basic Properties
X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page.	Basic Properties
Y	Vertical distance in pixels between the upper edge of this shape and the top edge of the page.	Basic Properties
Stroke Color	Color of the line.	Line Properties

Property	Description	Category
Stroke Width	Width of the stroke in pixels.	Line Properties

Example - Two Lines

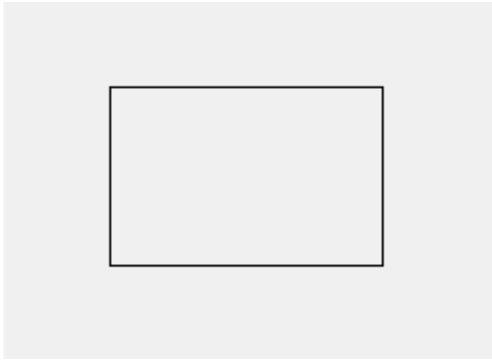


Property	Value
Stroke Style	Border
Stroke Color	Amber
Border Bottom	True
Border Top	True
Stroke Width	5 pixels

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Component - Rectangle Shape



Component Palette Icon:

 Rectangle

Description

The Rectangle shape component creates a square or rectangle and can be used anywhere on a report. Select the Rectangle shape on the Report Component Palette to make it active, and click and drag anywhere in your Report Designer to create a rectangle shape. Once the rectangle component is created, you can use the handles to change its height and width. Hold Shift while dragging to create a square instead of a rectangle. You can also change its stroke style, color, dash pattern, and much more.

Properties

Property Inspector	
2	
Stroke and Fill	
Fill	☒
Fill Color	
Opacity	1
Stroke Style	Hidden
Stroke	
Color	
Dash Pattern	
Width	1
Rectangle Shape	
Radius	0
Basic Properties	
Roll	0
Scale X	1
Scale Y	1
Visible	☒
Width	257
Height	187
X	135
Y	246

Property	Description	Category
Fill	If true, the shape will fill its space with color.	Stroke and Fill
Fill Color	If Fill is selected, the color that will fill the shape	Stroke and Fill
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Radius	Amount to radius the corners of this shape.	Rectangle Shape
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties
Height	Height of this shape in pixels.	Basic Properties

Property	Description	Category
X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page.	Basic Properties
Y	Vertical distance in pixels between the upper edge of this shape and the top edge of the page.	Basic Properties

Example

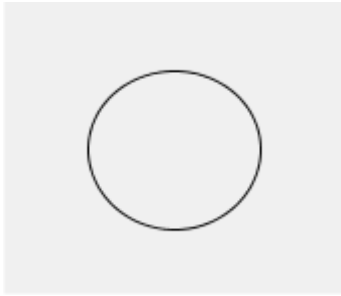


Property	Value
Fill Color	Yellow
Stroke Style	Double
Stroke Color	Blue
Stroke Inner Width	3 pixels
Stroke Outer Width	10 pixels
Stroke Position	Inner on path
Stroke Separation	5 pixels

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Component - Ellipse Shape



Component Palette Icon:



Description

The Ellipse shape component creates circles and ellipses that can be used in a report. Select the Rectangle shape on the Report Component Palette to make it active, and click and drag to draw an ellipse anywhere in the Report Designer. Once an ellipse or circle is created, you can use the handles to change its width and height. Hold Shift to force the shape into a circle. You can also change its stroke style, color, start angle, sweep angle, and more.

Properties

Property Inspector

2

Stroke and Fill

Fill

Fill Color

Opacity

1

Stroke Style

Shape Outline

Stroke

Color

Dash Pattern

Width

1

Basic Properties

Roll

0

Scale X

1

Scale Y

1

Visible

Width

90

Height

83

X

400

Y

100

Oval Shape

Start Angle

0

Sweep Angle

360

Property	Description	Category
Fill	If true, the shape will fill its space with color.	Stroke and Fill
Fill Color	If Fill is selected, the color that will fill the shape.	Stroke and Fill
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties
Height	Height of this shape in pixels.	Basic Properties
X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page.	Basic Properties

Property	Description	Category
Y	Vertical distance in pixels between the upper edge of this shape and the top edge of the page.	Basic Properties
Start Angle	Where to start the sweep angle, in degrees. 0 is at 3 o'clock.	Oval Shape
Sweep Angle	How many degrees the oval is swept. 0 shows just a line, 360 shows the full oval.	Oval Shape

Example

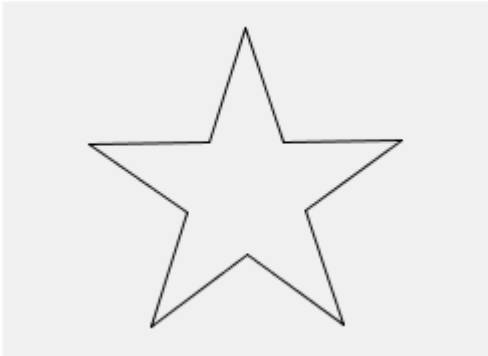


Property	Value
Stroke Style	Shape Outline
Stroke - Color	Red
Stroke - Width	5 pixels
Sweep Angle	360

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Component - Star Shape



Component Palette Icon:



Description

The Star shape component creates a star and can be used anywhere on a report. Select the Star shape on the Report Component Palette to make it active, and draw a star anywhere in your Report Designer. Once the star is created you can use the handles to resize it. You can change the number of star points, bloat, color, stroke, and much more using its properties.

Properties

Property Inspector	
2	
Star Shape	
Bloat	0.382
Points	5
Star Type	Star
Stroke and Fill	
Fill	
Fill Color	
Opacity	1
Stroke Style	Shape Outline
Stroke	
Color	
Dash Pattern	
Width	1
Basic Properties	
Roll	-90.666
Scale X	1
Scale Y	1
Visible	☒
Width	172.012
Height	172.012
X	96.994
Y	231.994

Property	Description	Category
Bloat	How wide each arm of the star is. 0 will give lines for each arm, 1 makes the star a polygon with twice as many sides as the points property.	Star Shape
Points	Number of points in the star shape.	Star Shape
Star Type	Switches the star shape from a Star to a Polygon.	Star Shape
Fill	If true, the shape will fill its space with color.	Stroke and Fill
Fill Color	If Fill is selected, the color that will fill the shape.	Stroke and Fill
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties

Property	Description	Category
Height	Height of this shape in pixels.	Basic Properties
X	Horizontal distance in pixels between the left edge of this shape to the left edge of the page.	Basic Properties
Y	Vertical distance in pixels between the upper edge of this shape to the top edge of the page.	Basic Properties

Example

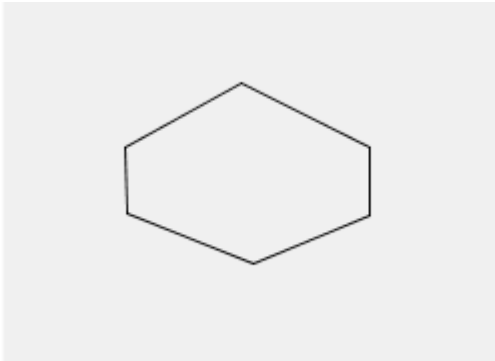


Property	Value
Points	8
Fill Color	Pink
Stroke Color	Blue
Roll	20

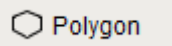
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Component - Polygon Shape



Component Palette Icon:



Description

The Polygon shape component creates custom polygon shapes by drawing and connecting lines, and can be used anywhere on a report. Select the Polygon shape on the Report Component Palette to make it active, click once to start a line and click to anywhere else to add additional corners. Click on the starting location to form a polygon shape. Once the polygon component is created, you can use the handles to change its size. Double click on any of the individual lines to change the size and shape of the selected line. You can also change its stroke style, color, dash pattern, and much more by using its properties.

Properties

Property Inspector

2

Stroke and Fill

Fill

Fill Color

Opacity

1

Stroke Style

Shape Outline

Stroke

Color

Dash Pattern

Width

1

Basic Properties

Roll

0

Scale X

1

Scale Y

1

Visible

Width

141

Height

102

X

94

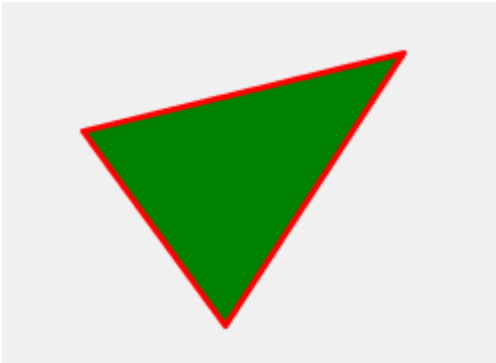
Y

93

Property	Description	Category
Fill	If true, the shape will fill its space with color.	Stroke and Fill
Fill Color	If Fill is selected, the color that will fill the shape.	Stroke and Fill
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties
Height	Height of this shape in pixels.	Basic Properties
X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page.	Basic Properties

Property	Description	Category
Y	Vertical distance in pixels between the upper edge of this shape and the top edge of the page.	Basic Properties

Example



Draw a triangle by clicking in 3 places on your screen, then once more at your start location.

Property	Value
Fill Color	Green
Stroke Style	Shape Outline
Stroke Color	Red
Stroke Width	3 pixels

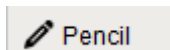
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Report Component - Pencil Shape



Component Palette Icon:



Description

The Pencil shape component lets you draw free-hand path segments that automatically smooth the curves. Using a Pencil shape can emphasize an area or an item on a report. Select the Pencil shape on the Report Design Palette to make it active, and draw a free-hand path anywhere in the Report Designer. If you stop drawing inside the small square that is placed at the shape's origin, then you will create a closed path, otherwise, you'll create an open path (line). Once a path is created, you can use the handles to change its width and height. You can also change its stroke style, color, and more.

Properties

Property Inspector

2

Stroke and Fill

Fill

Fill Color

Opacity

1

Stroke Style

Shape Outline

Stroke

Color

Dash Pattern

Width

1

Basic Properties

Roll

0

Scale X

1

Scale Y

1

Visible

Width

242

Height

77.845

X

53

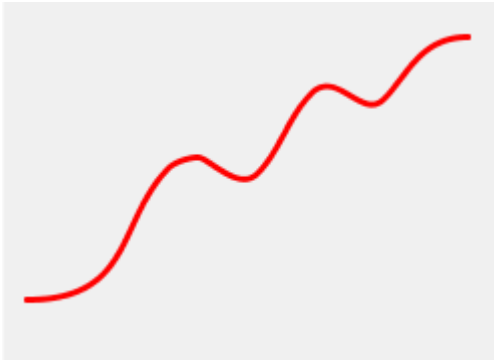
Y

135.155

Property	Description	Category
Fill	If true, the shape will fill its space with color.	Stroke and Fill
Fill Color	If Fill is selected, the color that will fill the shape.	Stroke and Fill
Opacity	How opaque the fill color is, between 0 and 1.	Stroke and Fill
Stroke Style	What style of stroke or border to use: Hidden, Shape Outline, Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	Stroke and Fill
Stroke	Details for the chosen stroke . Each Stroke has its own properties depending on the Stroke Style chosen.	Stroke and Fill
Roll	Number of degrees this shape is rotated clockwise.	Basic Properties
Scale X	Amount to scale the width of this shape. 1 is scale to 100%.	Basic Properties
Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.	Basic Properties
Visible	If true, the shape will be visible.	Basic Properties
Width	Width of this shape in pixels.	Basic Properties
Height	Height of this shape in pixels.	Basic Properties
X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page.	Basic Properties

Property	Description	Category
Y	Vertical distance in pixels between the upper edge of this shape and the top edge of the page.	Basic Properties

Example



Property	Value
Stroke Style	Shape Outline
Stroke Color	Red
Stroke Width	3 pixels

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Admin

The following components give you administrative access to various Gateway systems.

 **Roster Management**

 **User Management**

 **Schedule Management**

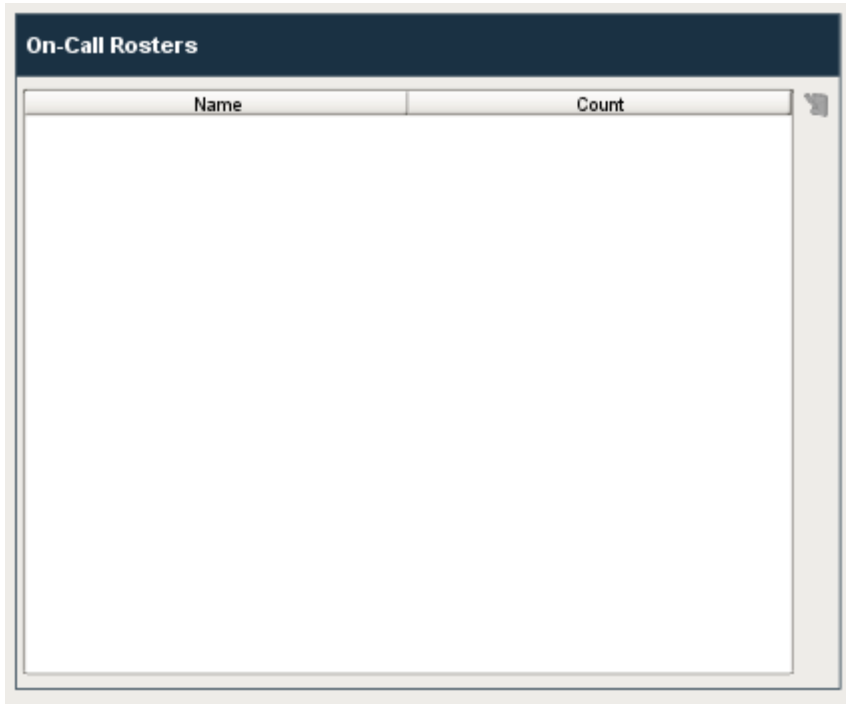
 **SFC Monitor**

A component to monitor SFC performance.

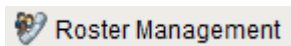
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Roster Management



Component Palette Icon:



Description

The user management panel provides a built-in way to edit rosters from a client.

Properties

Name	Description	Property Type	Scripting	Category
Antialias	Draw with antialias on? Makes text smoother.	boolean	.Antialias	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.Border	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.Data Quality	Data
Enabled	If disabled, a component cannot be used.	boolean	.Enabled	Common
Name	The name of this component.	String	.Name	Common
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	.User Source	Behavior
Visible	If disabled, the component will be hidden.	boolean	.Visible	Common

Scripting

See the [Roster Management Scripting Functions](#) page for the full list of scripting functions available for this component.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page.

Customizers

- [Vision Component Customizers](#)

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Roster Management Scripting Functions

This page details the various component and extension functions available for Vision's Roster Management component.

Component Functions

This component does not have scripting functions associated with it.

Extension Functions

filterRoster

- 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.
 - User Object **roster** - The name of the roster.
- Return
 - Boolean
- Scope
 - Client

filterAvailableUser

- 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
- Scope
 - Client

onSaveRoster

- 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
- Scope

- Client

onCreateRoster

- 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
- Scope
 - Client

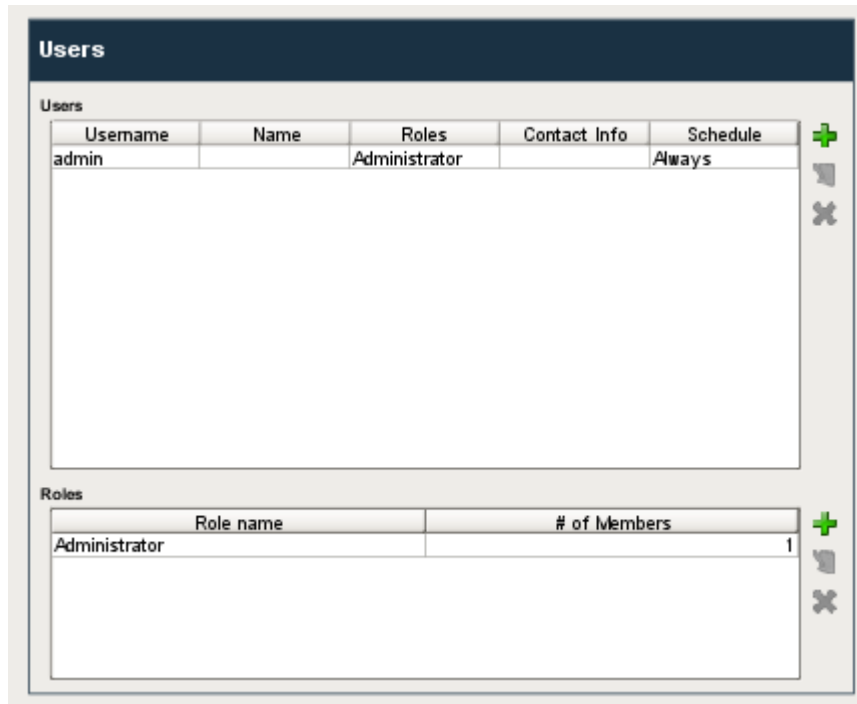
onDeleteRoster

- 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
- Scope
 - Client

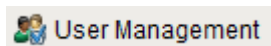
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

User Management



Component Palette Icon:



Description

The user management panel provides a built-in way to edit users and roles from a client. To use this component, you should be aware that it is only editing the users and roles from a single User Source. By default, the component will use the user source of the containing project. You can change this by typing in the name of another user source into the component's "User Source" property.

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 Configuration 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.

 CAUTION

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	Type	Scripting	Category
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Border	The border surrounding this component. Note that the border is unaffected by rotation.	Border	.border	Component
Contact Info Editing Enabled	If true, a user's contact info will be editable.	boolean	.allowContactInfoEditing	Behavior
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Editing Schedule Available Color	New in 7.9.5 Changes the color of the available times in the schedule.	Color	.schedulePreviewAvailableColor	Appearance

Name	Description	Type	Scripting	Category
Editing Schedule Available Text Color	New in 7.9.5 Changes the text color of events on the schedule preview.	Color	.eventForeground	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Communication
Font	New in 7.9.5 Font of the text on this component.	Font	.font	Appearance
Mode	Affects what mode the user management component runs in	int	.mode	Behavior
Name	The name of this component.	String	.name	Communication
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

Name	Description	Type	Scripting	Category
Row Height	New in 7.9.5 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

Name	Description	Type	Scripting	Category
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	New in 7.9.5 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	New in 7.9.5 Changes the	Color	.tableHeaderBackground	Appearance

Name	Description	Type	Scripting	Category
	background color of the table headers.			
Table Header Text Color	New in 7.9.5 Changes the text color of the table headers.	Color	.tableHeaderTextColor	Appearance
Table Text Color	New in 7.9.5 Changes the text color of the tables. Note: When a row is selected, it will revert to black.	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	String	.userProfile	Behavior

Name	Description	Type	Scripting	Category
	default user source.			
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	Component
Window Color	New in 7.9.5 Changes the window background color.	Color	.windowBackground	Appearance
Window Header Color	New in 7.9.5 Changes the window header	Color	.windowHeaderBackground	Appearance

Name	Description	Type	Scripting	Category
	background color.			
Window Header Save Button Background Color	New in 7.9.5 Changes the window header save button background color.	Color	.windowHeaderSaveButtonBackground	Appearance
Window Header Save Button Text Color	New in 7.9.5 Changes the window header save button text color.	Color	.windowHeaderSaveButtonForeground	Appearance
Window Header Text Color	New in 7.9.5 Changes the window header text color.	Color	.windowHeaderForeground	Appearance
Window Text Color	New in 7.9.5 Changes the text color of the window.	Color	.windowForeground	Appearance

Scripting

See the [User Management Scripting Functions](#) page for the full list of scripting functions available for this component.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page.

Customizers

- [Vision Component Customizers](#)

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

User Management Scripting Functions

This page details the various component and extension functions available for Vision's User Management component.

Component Functions

This component does not have scripting functions associated with it.

Extension Functions

filterUser

- 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 inspect. Common properties: 'username', 'schedule', 'language', user.getRoles() for a list of rolenames.
- Return
 - Boolean
- Scope
 - Client

filterRole

- 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
- Scope
 - Client

filterSchedule

- 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
- Scope
 - Client

onCreateUser

New in 7.9.4

- 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
- Scope
 - Client

onDeleteUser

New in 7.9.1

- 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
- Scope
 - Client

onSaveUser

- 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
- Scope
 - Client

onCreateRole

New in 7.9.4

- 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
- Scope
 - Client

onDeleteRole

New in 7.9.4

- 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
- Scope
 - Client

onSaveRole

- 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')`.
 - 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

- Scope
 - Client

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Schedule Management

Schedules

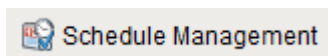
Schedules

Name	Description
Alternate Weekdays	Regular Day 1 Shift weekday schedule
Always	Built-in schedule that is always available: 24x7x365
Example	An example of a M-F 8am-5pm schedule with a lunch break

Holidays

Name
Christmas Eve
Christmas Day

Component Palette Icon:



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.

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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Border	The border surrounding this component. Note that the border is unaffected by rotation.	Border	.border	Component
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Component
Name	The name of this component.	String	.name	Component
Schedule Available Color	New in 7.9.5 Changes the color of	Color	.schedulePreviewAvailableColor	Appearance

Name	Description	Property Type	Scripting	Categ
	the available times in the schedule.			
Schedule Available Text Color	New in 7.9.5 Changes the text color of events on the schedule preview.	Color	.eventForeground	Appea
Styles	Contains the component's styles.	Dataset	.styles	Appea
Table Color	New in 7.9.5 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	Appea
Table Header Color	New in 7.9.5 Changes the	Color	.tableHeaderBackground	Appea

Name	Description	Property Type	Scripting	Category
	background color of the table headers.			
Table Header Text Color	New in 7.9.5 Changes the text color of the table headers.	Color	.tableHeaderTextColor	Appearance
Table Text Color	New in 7.9.5 Changes the text color of the tables. Note: When a row is selected, it will revert to black.	Color	.tableForeground	Appearance
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior

Name	Description	Property Type	Scripting	Categ
Visible	If disabled, the component will be hidden.	boolean	.visible	Comm
Window Color	New in 7.9.5 Changes the window background color.	Color	.windowBackground	Appea
Window Header Color	New in 7.9.5 Changes the window header background color.	Color	.windowHeaderBackground	Appea
Window Header Save Button Background Color	New in 7.9.5 Changes the window header save button background color.	Color	.windowHeaderSaveButtonBackground	Appea
Window Header Save Button Text Color	Changes the window header save button text color.	Color	.windowHeaderSaveButtonForeground	Appea

Name	Description	Property Type	Scripting	Category
Window Header Text Color	New in 7.9.5 Changes the window header text color.	Color	.windowHeaderForeground	Appearance
Window Text Color	New in 7.9.5 Changes the text color of the window.	Color	.windowForeground	Appearance

Scripting

See the [Schedule Management Scripting Functions](#) page for the full list of scripting functions available for this component.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page.

Customizers

- [Vision Component Customizers](#)

Example

Here is an example of the schedule management component and its property table.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Schedule Management Scripting Functions

This page details the various component and extension functions available for Vision's Schedule Management component.

Component Functions

This component does not have scripting functions associated with it.

Extension Functions

filterSchedule

- 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
- Scope
 - Client

filterHoliday

- 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
- Scope
 - Client

onCreateSchedule

New in 7.9.4

- 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 `saveContext.rejectSave('reason')`.
- Return
 - Nothing
- Scope
 - Client

onDeleteSchedule

New in 7.9.4

- 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 add by calling `saveContext.rejectSave('reason')`
 - String **name** - The name of the schedule to be deleted.
- Return
 - Nothing
- Scope
 - Client

onSaveSchedule

- 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 `saveContext.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

- Scope
 - Client

onCreateHoliday

New in 7.9.4

- 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 edit by calling `saveContext.rejectSave('reason')`.
- Return
 - Nothing
- Scope
 - Client

onDeleteHoliday

New in 7.9.4

- 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 add by calling `saveContext.rejectSave('reason')`
 - String **name** - The name of the holiday to be deleted.
- Return

- Nothing
- Scope
 - Client

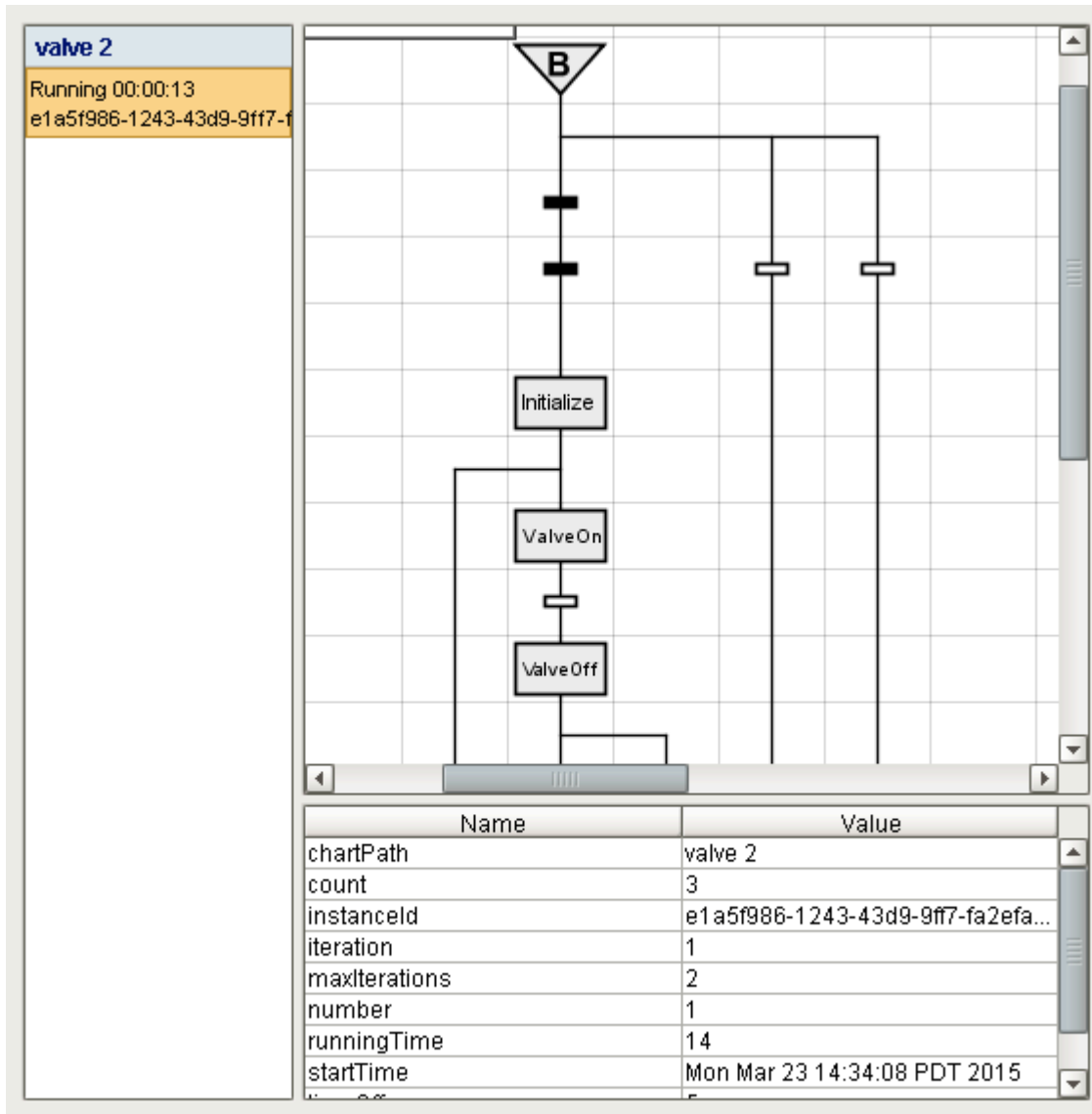
onSaveHoliday

- 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 by calling `saveContext.rejectSave('reason')`. If more than one role is rejected, reasons will be concatenated.
 - 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
- Scope
 - Client

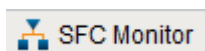
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

SFC Monitor



Component Palette Icon:



Description

A component to monitor SFC 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. NOTE that 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	float	.zoom	Appearance

Name	Description	Property Type	Scripting	Category
	status at.			

Scripting

Component 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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page.

Customizers

- [Vision Component Customizers](#)

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Alarming

The following components give you options for displaying Alarm information.

Alarm Status Table

Alarm Journal Table

The alarm journal table provides a built-in view to explore alarm history that has been stored in an alarm journal.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

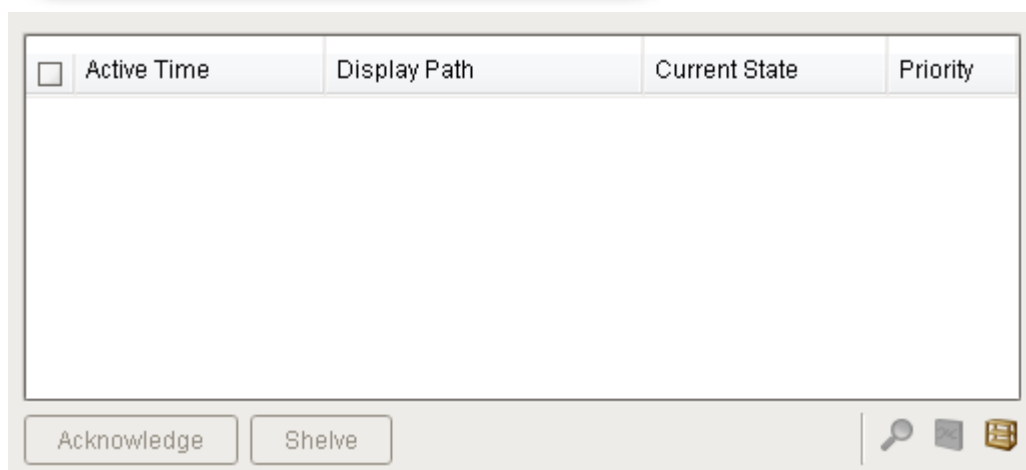
Alarm Status Table



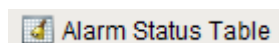
INDUCTIVE
UNIVERSITY

Alarm Status Table

[Watch the video](#)



Component Palette Icon



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.

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.

NOTE

An Example of configuring the Alarm Status Table can be found on the [Alarm Status Table Component](#) page.

Properties

Name	Description	Property Type	Scripting	Category
Antialias	Draw the component with antialias on. This makes text look smoother	boolean	.antialias	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Chart Resolution	The resolution for the ad-hoc tag historian chart.	int	.chartResolution	Behavior
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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	String	.displayPathFilter	Filters

Name	Description	Property Type	Scripting	Category
	if a custom display path isn't set. Specify multiple paths by separating them with commas. This field supports the wildcard "*" to match any number of characters.			
Duration Format	New in 7.9.12 Format for the time units on the Active Duration column: Long, Short, Compact, and Abbreviated.	String	.durationFormat	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Flash Interval	The time interval to use for flashing row styles.	int	.flashInterval	Appearance
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	.alarmJournalName	Behavior

Name	Description	Property Type	Scripting	Category
Marquee Mode	Turn the table into a scrolling marquee	boolean	.marqueeMode	Behavior
Min Priority	The minimum priority alarm to be displayed by this table.	int	.minPriority	Filters
Multi Select	Allow multi select. Will show/hide the checkbox column.	boolean	.multiSelect	Behavior
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
Refresh Rate	The rate at which this table will poll	long	.refreshRate	Behavior

Name	Description	Property Type	Scripting	Category
	changes to the alarm status, in milliseconds.			
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
Scroll Delay	The time (in mSec) to wait between performing each step in a scroll	int	.scrollDelay	Behavior
Selected Alarms	A dataset containing each selected alarm. (Read-only)	Dataset	.selectedAlarms	Data
Selection Color	The color of the selection border	Color	.selectionColor	Appearance
Selection Thickness	The size of the selection border	int	.selectionThickness	Appearance
Shelving Times	This dataset holds the times that are suggested when shelving an alarm. Allowable units are "second",	Dataset	.shelvingTimes	Data

Name	Description	Property Type	Scripting	Category
	"minute", or "hour".			
Show Ack Button	Show the acknowledge button on the footer panel.	boolean	.showAck	Appearance
Show Active and Acked	Show alarms that are active and acknowledged.	boolean	.activeAndAked	Filters
Show Active and Unacked	Show alarms that are active and unacknowledged.	boolean	.activeAndUnacked	Filters
Show Chart Button	Show the chart button on the footer panel.	boolean	.showChart	Appearance
Show Clear and Aked	Show alarms that are cleared and acknowledged.	boolean	.clearAndAked	Filters
Show Clear and Unacked	Show alarms that are cleared and unacknowledged.	boolean	.clearAndUnacked	Filters
Show Details Button	Show the view details button on the footer panel.	boolean	.showDetails	Appearance
Show Footer	Show a footer with acknowledge and shelf functions below the alarms.	boolean	.showFooterPanel	Appearance

Name	Description	Property Type	Scripting	Category
Show Header Popup	Toggles the table header's built-in column selection popup menu.	boolean	.showTableHeaderPopup	Appearance
Show Manage Shelf Button	Show the manage shelf button on the footer panel.	boolean	.showManageShelf	Appearance
Show Shelve Button	Show the shelve button on the footer panel.	boolean	.showShelve	Appearance
Show Table Header	Toggles visibility of the table's header.	boolean	.showTableHeader	Appearance
Sort Oldest First	Sort times by oldest first.	boolean	.sortOldestFirst	Behavior
Sort Order	The default sort order for alarms in the status table.	int	.sortOrder	Behavior
Source Filter	Filter alarms by alarm source path. Specify multiple paths by separating them with commas. This field supports the wildcard "*" to match any number of characters.	String	.sourceFilter	Filters

Name	Description	Property Type	Scripting	Category
Stay Delay	The time (in mSec) to wait between scrolls	int	.stayDelay	Behavior
Table Background	The background of the alarm table.	Color	.tableBackground	Appearance
Table Header Font	The font for the table header.	Font	.tableHeaderFont	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

Scripting

Scripting Functions

.print(fitWidth, headerFormat, footerFormat, showDialog, landscape)

- 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.
- Scope
 - Client

New in 7.9.7 .getAlarms()

- 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.
- Scope
 - Client

Extension Functions

createPopupMenu

- 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.
- Scope
 - Client

filterAlarm

- 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.
- Scope
 - Client

isAcknowledgeEnabled

- 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.
- Scope
 - Client

isShelvedEnabled

- 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.
- Scope
 - Client

onDoubleClicked

- 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
- Scope
 - Client

onAcknowledge

- 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.
- Scope
 - Client

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- Vision Component Customizers

Example

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
```

Alarm Status Table with a Single Alarm

☐	Active Time	Display Path	Current State	Priority
☐	1/20/15 8:26 AM	London	Active, Unacknowledged	Low

Acknowledge

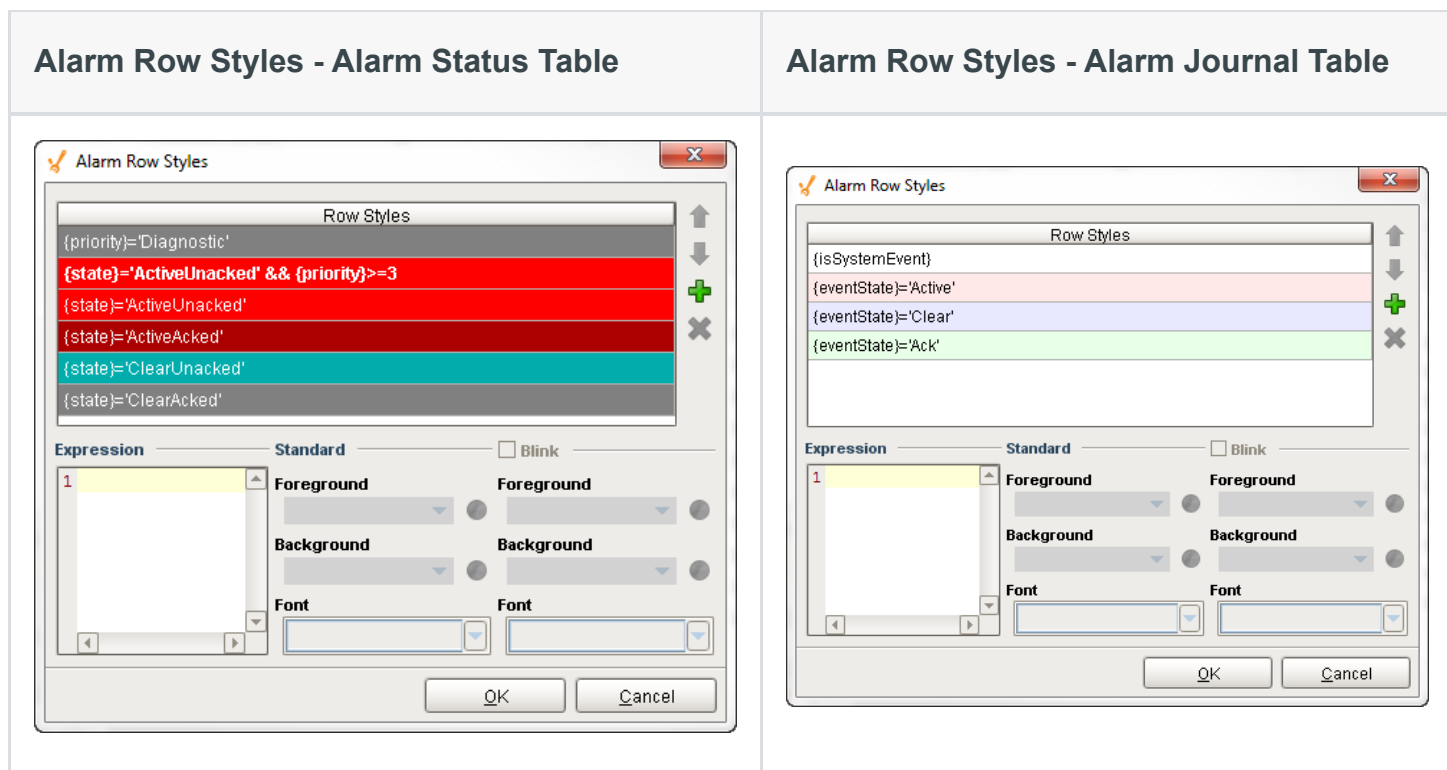
Shelve



This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Alarm Row Style Customizer



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.

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, Clear Time, and many more.
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

Alarm Row Styles

Row Styles

{state}='ActiveUnacked' && {priority}>=4

{state}='ActiveUnacked' && {priority}>=3

{state}='ActiveUnacked'

{state}='ActiveAcked'

{state}='ClearUnacked'

{state}='ClearAcked'

{priority}='Diagnostic'

↑

↓

+

×

Expression

1 {state}='ActiveUnacked' && {priority}>=4

Standard

Foreground

Background

Font

Foreground

Background

Font

☒ Blink

OK

Cancel

Alarm Journal Table - Alarm Row Styles

Alarm Row Styles

Row Styles

{eventState}='Active' && {priority}>=4

{isSystemEvent}

{eventState}='Active'

{eventState}='Clear'

{eventState}='Ack'

↑

↓

+

×

Expression

Standard

☐ Blink

1 {eventState}='Active' && {priority}>=4

Foreground

Background

Font

Foreground

Background

Font

OK

Cancel

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Alarm Journal Table



INDUCTIVE
UNIVERSITY

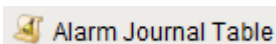
Alarm Journal Table

[Watch the video](#)

Event Time	Display Path	Event St...	Priority	Ack'd By	Event Value
3/12/15 11:25 AM	test	Clear	Low		-1
3/12/15 11:24 AM	test	Active	Low		0
3/12/15 11:24 AM	test	Clear	Low		-2
3/12/15 11:23 AM	test	Active	Low		1
3/12/15 11:23 AM	test	Clear	Low		-2
3/12/15 11:22 AM	test	Active	Low		1
3/12/15 11:22 AM	test	Clear	Low		-1
3/12/15 11:21 AM	test	Active	Low		0
3/12/15 11:21 AM	test	Clear	Low		-1
3/12/15 11:20 AM	test	Active	Low		0
3/12/15 11:20 AM	test	Clear	Low		-2
3/12/15 11:19 AM	test	Active	Low		1
3/12/15 11:19 AM	test	Clear	Low		-2
3/12/15 11:18 AM	test	Active	Low		2
3/12/15 11:18 AM	test	Clear	Low		-1

960 events

Component Palette Icon



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, please bind its start and end date properties to something that 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.

NOTE

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 acknowledged events.	boolean	.includeAckedEvents	Filters
Active Events	Show active events.	boolean	.includeActiveEvents	Filters
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Cleared Events	Show cleared events.	boolean	.includeClearedEvents	Filters
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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	String	.displayPathFilter	Filters

Name	Description	Property Type	Scripting	Category
	set. Specify multiple paths by separating them with commas. Supports the wildcard "*".			
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

Name	Description	Property Type	Scripting	Category
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
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

Name	Description	Property Type	Scripting	Category
Selected Alarms	A dataset containing each selected alarm. (Read-only)	Dataset	.selectedAlarms	Data
Selection Color	The color of the selection border.	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.	Color	.tableBackground	Appearance
Touchscreen Mode	Controls when this input component	int	.touchscreenMode	Behavior

Name	Description	Property Type	Scripting	Category
	responds if touchscreen mode is enabled.			
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

Scripting Functions

.print(fitWidth, headerFormat, footerFormat, showDialog, landscape)

- 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.
- Scope

- Client

New in 7.9.7 `.getAlarms()`

- 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.
- Scope
 - Client

Extension Functions

`createPopupMenu`

- 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 alarm Event, call `alarmEvent.get('propertyName')` to inspect. Common properties: 'name', 'source', 'priority'.
- Return
 - **JPopupMenu** - A popup menu that was created with [system.gui.createPopupMenu\(\)](#)
- Scope
 - Client

`filterAlarm`

- 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.
- Scope
 - Client

onDoubleClicked

- 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
- Scope
 - Client

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Buttons

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

Button

The Button component is a versatile component, often used for things like opening/closing windows, writing to ta...

2 State Toggle

This button is similar to the basic Toggle Button, but more finely tuned to work in realistic controls environments....

Momentary Button

Momentary buttons are used to set a value for either a fixed amount of time, or however long the button remains ...

Multi-State Button

This button is really a series of two or more buttons, arranged in a column, row, or grid. Each button represents a...

One-Shot Button

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...

Toggle Button

The Toggle button represents a bit: on (selected) or off (not selected).

Check Box

A CheckBox is a familiar component that represents a bit - it is either on (selected) or off (not selected). It is funct...

Radio Button

The radio button is similar to the CheckBox component, except for one special property. All radio buttons in the s...

Tab Strip

In general, a Tab Strip is just a single-selection multiple choice component. In practice it is used anywhere that a ...

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Button



Component Palette Icon



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 as buttons can be activated by tabbing over to them and hitting the space key, or 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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background 3D?	Should this button have a 3d type background, or a flat color one?	boolean	.background3D	Appearance
Background Color	The background color of the button.	Color	.buttonBG	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Border Painted?	Should the border of this button be displayed?	boolean	.borderPainted	Appearance
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Default Button	If true, this button will be activated	boolean	.defaultBtn	Behavior

Name	Description	Property Type	Scripting	Category
	when the user presses <i>enter</i> on the window.			
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.	Color	.foreground	Appearance

Name	Description	Property Type	Scripting	Category
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 ' <i>ALT-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?	boolean	.opaque	Common

Name	Description	Property Type	Scripting	Category
	Most aren't, so this should usually be false.			
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

Scripting

Scripting Functions

.doClick()

- Description

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

- Parameters
Nothing
- Return
Nothing
- Scope
Client

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Styled Button Examples



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



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

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

2 State Toggle



Component Palette Icon



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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background 3D?	Should this button have a 3d type background, or a flat color one?	boolean	.background3D	Appearance
Background Color	The background color of the button.	Color	.buttonBG	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Border Painted?	Should the border of this button 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	int	.controlValue	Data

Name	Description	Property Type	Scripting	Category
	is bound to the same location as Indicator Value).			
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.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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	boolean	.focusable	Behavior

Name	Description	Property Type	Scripting	Category
	interact with it with the keyboard. This means you can't "tab" over to it.			
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	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	int	.indicatorValue	Data

Name	Description	Property Type	Scripting	Category
	location as Control Value)			
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
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

Name	Description	Property Type	Scripting	Category
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

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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Example

2-State Toggle with Styles Configured



Property Name	Value					
Styles						
	state	animationIndex	animationDuration	text	buttonBG	foreground
	0	0	50	Blue		
	1	0	50	Purple		

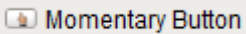
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Momentary Button

A rectangular button with a thin black border and the text "Momentary Button" inside.

Component Palette Icon

A rectangular button with a thin black border, a small icon of a button with a flame, and the text "Momentary Button" inside.

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.

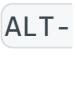
NOTE

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 [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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background 3D?	Should this button have a 3d type background, or a flat color one?	boolean	.background3D	Appearance
Background Color	The background color of the button.	Color	.buttonBG	Appearance
Border	The border surrounding this component.	Border	.innerBorder	Common
Control Value	Bind this to the tag that you want to control. (Typically, this is bound to the same location as <i>Indicator Value</i>).	int	.controlValue	Data
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Disabled Image Path	The relative path of the image to be	String	.disabledPath	Appearance

Name	Description	Property Type	Scripting	Category
	displayed when this component is not enabled.			
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.	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

Name	Description	Property Type	Scripting	Category
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 Control Value).	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".	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  .	String	.mnemonicChar	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops	String	.toolTipText	Common

Name	Description	Property Type	Scripting	Category
	up on mouseover of this component.			
Name	The name of this component.	String	.name	Common
Off Color	The color of the indicator border when the indicator value is off.	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.	Color	.onColor	Appearance
On Value	The value that will be written to the Control Value on mouse-down.	int	.onValue	Behavior
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

Name	Description	Property Type	Scripting	Category
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

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples



Momentary Button
waiting to be pressed



Activated Momentary
Button

**Momentary Button Activation States

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Multi-State Button



Component Palette Icon



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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background 3D?	Controls whether or not the buttons have a gradient-style background color.	boolean	.background3D	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 <i>Indicator Value</i>)	int	.controlValue	Data
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Display Style	The display style (rows or columns) for this N-state button.	int	.displayStyle	Appearance

Name	Description	Property Type	Scripting	Category
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 <i>Control Value</i>).	int	.indicatorValue	Data
Mouseover Text	The text that is displayed in the tooltip which pops up on	String	.toolTipText	Common

Name	Description	Property Type	Scripting	Category
	mouseover of this component.			
Name	The name of this component.	String	.name	Common
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

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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

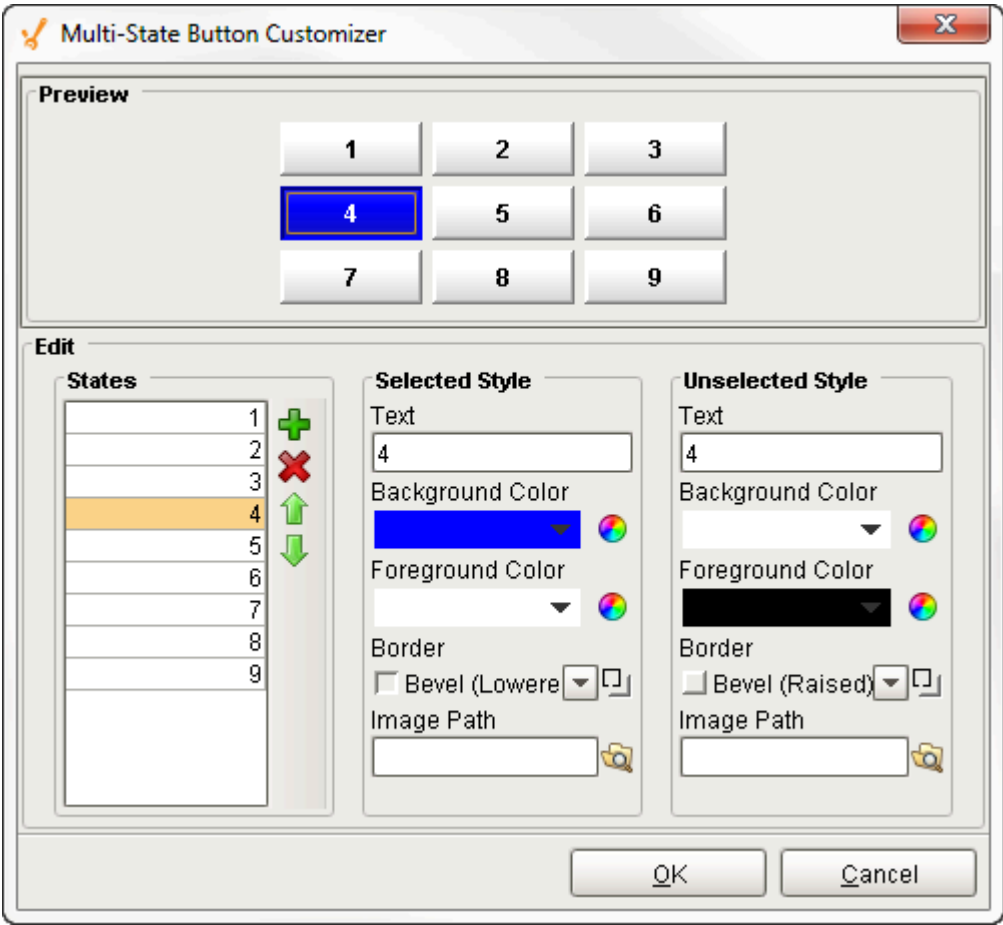
The Multi-State Button has its own Customizer. Here, you can define your states, change the order, values, and styles. You can organize your buttons to be a column, row, or grid by setting the **Display Style** property in the Property Editor. You'll notice that the Multi-State Button Customizer already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

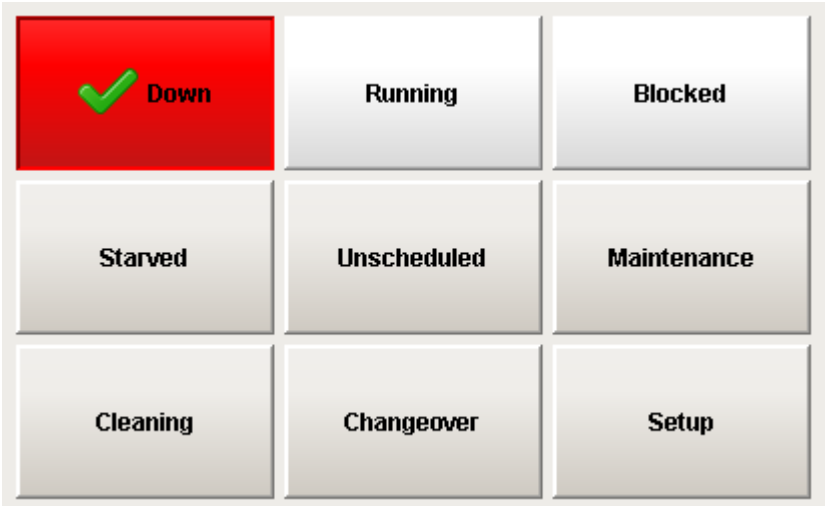
Property	Description
Preview	Lets you preview each button's display style, states, and the selected style and unselected style as you configure it.
States	Shows a list of all possible states. You can add, remove, and the change the order of each state listed. Each state also defines two visual styles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

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.



- [Vision Component Customizers](#)
- [Style Customizer](#)

Example



Property Name	Value																																																																																																														
Display Style	Grid																																																																																																														
Styles	<table><tr><th>value</th><th>selectedText</th><th>unselectedText</th><th>selectedBackgr</th><th>unselectedBack</th><th>selectedForegro</th><th>unselectedFore</th><th>selectedBorder</th><th>unselectedBorder</th><th>selectedImage</th><th>unselectedImage</th></tr><tr><td>0</td><td>Down</td><td>Down</td><td></td><td></td><td></td><td></td><td></td><td></td><td>border(bevel,1)</td><td>border(bevel,0)</td></tr><tr><td>1</td><td>Running</td><td>Running</td><td></td><td></td><td></td><td></td><td></td><td></td><td>border(bevel,1)</td><td>border(bevel,0)</td></tr><tr><td>2</td><td>Blocked</td><td>Blocked</td><td></td><td></td><td></td><td></td><td></td><td></td><td>border(bevel,1)</td><td>border(bevel,0)</td></tr><tr><td>3</td><td>Starved</td><td>Starved</td><td></td><td></td><td></td><td></td><td></td><td></td><td>border(bevel,1)</td><td>border(bevel,0)</td></tr><tr><td>4</td><td>Unscheduled</td><td>Unscheduled</td><td></td><td></td><td></td><td></td><td></td><td></td><td>border(bevel,1)</td><td>border(bevel,0)</td></tr><tr><td>5</td><td>Maintenance</td><td>Maintenance</td><td></td><td></td><td></td><td></td><td></td><td></td><td>border(bevel,1)</td><td>border(bevel,0)</td></tr><tr><td>6</td><td>Cleaning</td><td>Cleaning</td><td></td><td></td><td></td><td></td><td></td><td></td><td>border(bevel,1)</td><td>border(bevel,0)</td></tr><tr><td>7</td><td>Changeover</td><td>Changeover</td><td></td><td></td><td></td><td></td><td></td><td></td><td>border(bevel,1)</td><td>border(bevel,0)</td></tr><tr><td>8</td><td>Setup</td><td>Setup</td><td></td><td></td><td></td><td></td><td></td><td></td><td>border(bevel,1)</td><td>border(bevel,0)</td></tr></table>	value	selectedText	unselectedText	selectedBackgr	unselectedBack	selectedForegro	unselectedFore	selectedBorder	unselectedBorder	selectedImage	unselectedImage	0	Down	Down							border(bevel,1)	border(bevel,0)	1	Running	Running							border(bevel,1)	border(bevel,0)	2	Blocked	Blocked							border(bevel,1)	border(bevel,0)	3	Starved	Starved							border(bevel,1)	border(bevel,0)	4	Unscheduled	Unscheduled							border(bevel,1)	border(bevel,0)	5	Maintenance	Maintenance							border(bevel,1)	border(bevel,0)	6	Cleaning	Cleaning							border(bevel,1)	border(bevel,0)	7	Changeover	Changeover							border(bevel,1)	border(bevel,0)	8	Setup	Setup							border(bevel,1)	border(bevel,0)
value	selectedText	unselectedText	selectedBackgr	unselectedBack	selectedForegro	unselectedFore	selectedBorder	unselectedBorder	selectedImage	unselectedImage																																																																																																					
0	Down	Down							border(bevel,1)	border(bevel,0)																																																																																																					
1	Running	Running							border(bevel,1)	border(bevel,0)																																																																																																					
2	Blocked	Blocked							border(bevel,1)	border(bevel,0)																																																																																																					
3	Starved	Starved							border(bevel,1)	border(bevel,0)																																																																																																					
4	Unscheduled	Unscheduled							border(bevel,1)	border(bevel,0)																																																																																																					
5	Maintenance	Maintenance							border(bevel,1)	border(bevel,0)																																																																																																					
6	Cleaning	Cleaning							border(bevel,1)	border(bevel,0)																																																																																																					
7	Changeover	Changeover							border(bevel,1)	border(bevel,0)																																																																																																					
8	Setup	Setup							border(bevel,1)	border(bevel,0)																																																																																																					

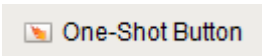
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

One-Shot Button

A rectangular button with a thin border and the text "One-Shot Button" centered inside.

Component Palette Icon

A small icon of the One-Shot Button component, showing a button with a red status indicator and the text "One-Shot Button".

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. Note that 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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background 3D?	Should this button have a 3d type background, or a flat color one?	boolean	.background3D	Appearance
Background Color	The background color of the button.	Color	.buttonBG	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Border Painted?	Should the border of this button 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	int	.cursorCode	Common

Name	Description	Property Type	Scripting	Category
	hovering over this component.			
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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

Name	Description	Property Type	Scripting	Category
Foreground Color	The foreground color of the component.	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	String	.toolTipText	Common

Name	Description	Property Type	Scripting	Category
	tooltip which pops up on mouseover of this component.			
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
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

Name	Description	Property Type	Scripting	Category
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

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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

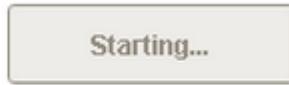
- [Vision Component Customizers](#)
- [Style Customizer](#)

Example

One Shot Button



A One-Shot button,
waiting to be pressed

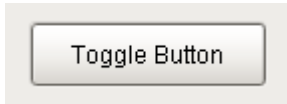


A One-Shot button,
waiting for a PLC reset

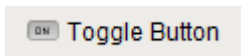
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Toggle Button



Component Palette Icon



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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background 3D?	Should this button have a 3d type background, or a flat color one?	boolean	.background3D	Appearance
Background Color	The background color of the button.	Color	.buttonBG	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Border Painted?	Should the border of this button be displayed?	boolean	.borderPainted	Appearance
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common

Name	Description	Property Type	Scripting	Category
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.	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
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	Set this to false if you want the button to be	boolean	.opaque	Appearance

Name	Description	Property Type	Scripting	Category
	completely opaque.			
Rollover?	If true, the button may indicate that the mouse is hovering over it.	boolean	.rolloverEnabled	Appearance
Selected	State of this tToggle 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

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

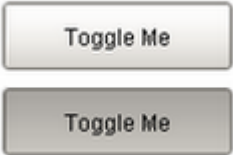
Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples



This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Check Box

☐ Check Box

Component Palette Icon

 Check Box

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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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.	boolean	.focusable	Behavior

Name	Description	Property Type	Scripting	Category
	This means you can't "tab" over to it.			
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	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
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
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

Name	Description	Property Type	Scripting	Category
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

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples

- ☐ Show Defective
- ☒ Show Normal
- ☒ Sort By Shift

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Radio Button

☐ Radio Button

Component Palette Icon

 Radio Button

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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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.	boolean	.focusable	Behavior

Name	Description	Property Type	Scripting	Category
	This means you can't "tab" over to it.			
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	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
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
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

Name	Description	Property Type	Scripting	Category
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

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Example



Selection

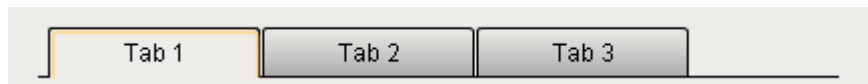
- ☐ Radio Button
- ☐ Radio Button
- ☒ Radio Button

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

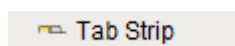
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Tab Strip



Component Palette Icon



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:

- **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.
- **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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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	int	.navigationMode	Behavior

Name	Description	Property Type	Scripting	Category
	swaps to the window whose name corresponds to the name of the selected tab, provided that window exists.			
Orientation	Orientation of the tab strip.	int	.orientation	Appearance
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.	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	int	.sizeMode	Appearance

Name	Description	Property Type	Scripting	Category
	decide its own size based on the size of its text.			
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Tab Data	Tab Data.	Dataset	.tabData	Data
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

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

The Tab Strip Customizer has it's 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.

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 <code>system.nav.swapTo()</code> 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 <code>propertyChange</code> 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.

Properties	Description
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 Section

Properties	Description
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

Properties	Description
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 Browser 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.

Tab Strip Customizer

Properties

Orientation

Left

Navigation Mode

Swap Windows

Size Mode

Individual

Style

Simple

+ Add Tab

Remove Tab

Move Up

Move Down

Text Alignment

Center

Text Offset

4

Text Padding

20

Intertab Space

1

Rounding Radius

10

Preview

Overview

User Management

Schedule Management

Roster Management

Tab 1

General

Window Name

Tab 1

Display Name

Tab 1

Mouseover Text

This is a tooltip

Hover Color

When Selected

Background Color

Foreground Color

Font

Dialog

Gradient Start Color

Gradient End Color

☒ Use Gradient

Tab Icon

Edit Tab Icon

When Unselected

Background Color

Foreground Color

Font

Dialog

Gradient Start Color

Gradient End Color

☒ Use Gradient

Tab Icon

Edit Tab Icon

Apply To All

OK

Cancel

- [Navigation - Tab Strip](#)
- [Vision Component Customizers](#)
- [Style Customizer](#)

Example

Horizontal Tabs

Properties	Description
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

+ Add Tab

Remove Tab

Move Left

Move Right

Text Alignment

Center

Text Offset

4

Text Padding

46

Intertab Space

4

Rounding Radius

40

Preview

Overview

User Management

Schedule

Tab 1

General

Window Name

Tab 1

Display Name

Tab 1

Mouseover Text

This is a tooltip

Hover Color

When Selected

Background Color

Foreground Color

Font

Dialog

Gradient Start Color

Gradient End Color

Use Gradient

Tab Icon

Edit Tab Icon

When Unselected

Background Color

Foreground Color

Font

Dialog

Gradient Start Color

Gradient End Color

Use Gradient

Tab Icon

Edit Tab Icon

Apply To All

OK

Cancel

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Calendars

The following components give you options for displaying and selecting dates and times.

Calendar

Displays a calendar and time input directly embedded in your window.

Popup Calendar

The popup calendar is a popular way to provide date/time choosing controls on a window.

Date Range

Day View

This component displays a timeline for a single day, similar to what you might find in a personal planner/organizer.

Week View

Displays a full week's worth of events on a calendar.

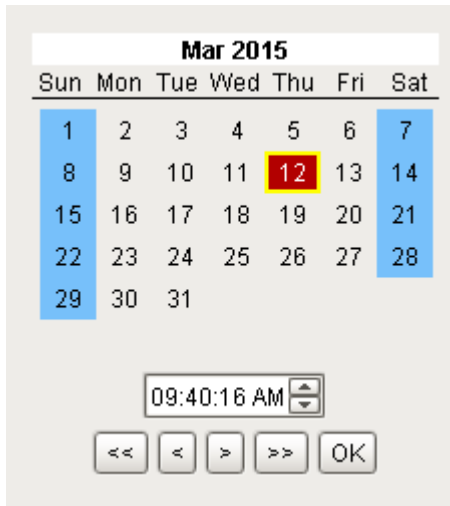
Month View

This component displays events for an entire month.

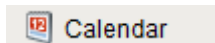
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Calendar



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**.

Properties

Name	Description	Property Type	Scripting	Category
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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

Name	Description	Property Type	Scripting	Category
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	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

Name	Description	Property Type	Scripting	Category
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.	Color	.titleBackground	Appearance
Today Background	Background color for the today indicator.	Color	.todayBackground	Appearance
Today Foreground	Foreground color for the today indicator.	Color	.todayForeground	Appearance

Name	Description	Property Type	Scripting	Category
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Weekend Background	Background color for the weekend indicators.	Color	.weekendBackground	Appearance
Weekend Foreground	Foreground color for the weekend indicators.	Color	.weekendForeground	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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

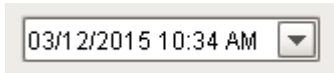
Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

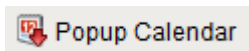
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Popup Calendar



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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Calendar Background	The background color for the popup calendar.	Color	.calendarBackground	Appearance
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Date	The date that this component represents.	Date	.date	Data
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common

Name	Description	Property Type	Scripting	Category
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	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
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

Name	Description	Property Type	Scripting	Category
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.	Color	.todayBackground	Appearance
Today Foreground	Foreground color for the today indicator.	Color	.todayForeground	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Weekend Background	Background color for the weekend indicators.	Color	.weekendBackground	Appearance
Weekend Foreground	Foreground color for the weekend indicators.	Color	.weekendForeground	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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

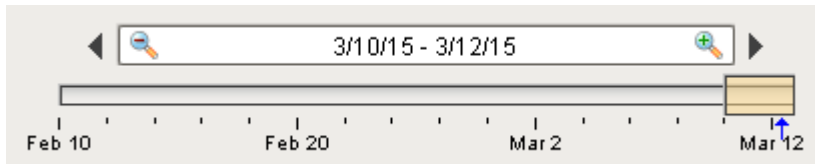
Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

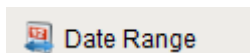
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Date Range



Component Palette Icon:



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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that 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.	int	.cursorCode	Common
Data Density	A dataset that is used to calculate a histogram of data density.	Dataset	.densityData	Data
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Date Style	The style to display dates in. For international support.	int	.dateStyle	Appearance

Name	Description	Property Type	Scripting	Category
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.	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

Name	Description	Property Type	Scripting	Category
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
Selection Highlight	The focus highlight color for the selection box.	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	String	.startupSelection	Behavior

Name	Description	Property Type	Scripting	Category
	be the starting selected range.			
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.	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

Scripting

Scripting Functions

See the [Date Range Scripting Functions](#) page for the full list of scripting functions available for this component.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Example

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}"
```

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Date Range Scripting Functions

This page details the various component and extension functions available for Vision's Report Viewer component.

Component Functions

.setRange(start, end)

- 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. IE: 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
- Scope
 - Client

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)
```

.setOuterRange(start, end)

- 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. IE: 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
- Scope
 - Client

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)
```

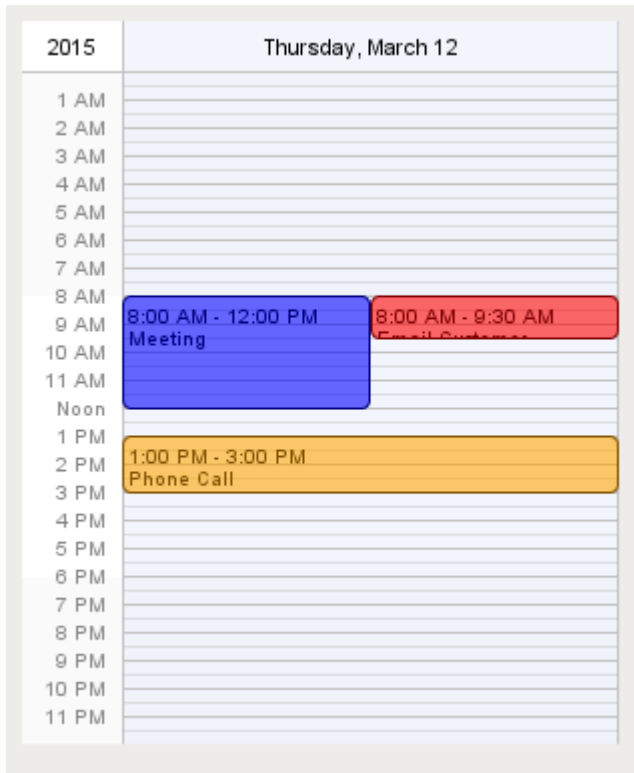
Extension Functions

This component does not have extension functions associated with it.

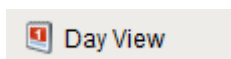
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Day View



Component Palette Icon:



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. NOTE that the border is unaffected by rotation.	Border	.border	Common
Calendar Background Color	The color of the calendar's background.	Color	.calendarBackground	Appearance
Calendar events	Contains the calendar events.	Dataset	.events	Data
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on	int	.dataQuality	Data

Name	Description	Property Type	Scripting	Category
	this component.			
Day	Set the calendar's day.	int	.day	Data
Day Outline Color	The color of the day's outline.	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.	Color	.hourForeground	Appearance
Hover Background Color	The background color of the hovered time.	Color	.hoverBackground	Appearance
Hovered Event	The calendar's hovered event.	int	.hoveredEvent	Data

Name	Description	Property Type	Scripting	Category
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.	Color	.nonWorkingHourBackground	Appearance
Selected Event	The calendar's selected event.	int	.selectedEvent	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance

Name	Description	Property Type	Scripting	Category
Today's Background Color	The color of the today's background.	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.	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.	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	boolean	.autoZoom	Appearance

Name	Description	Property Type	Scripting	Category
	zoom time-range.			
Zoomed End Hour	The end hour zoomed in.	int	.autoZoomEndHour	Appearance
Zoomed Start Hour	The start hour zoomed in.	int	.autoZoomStartHour	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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

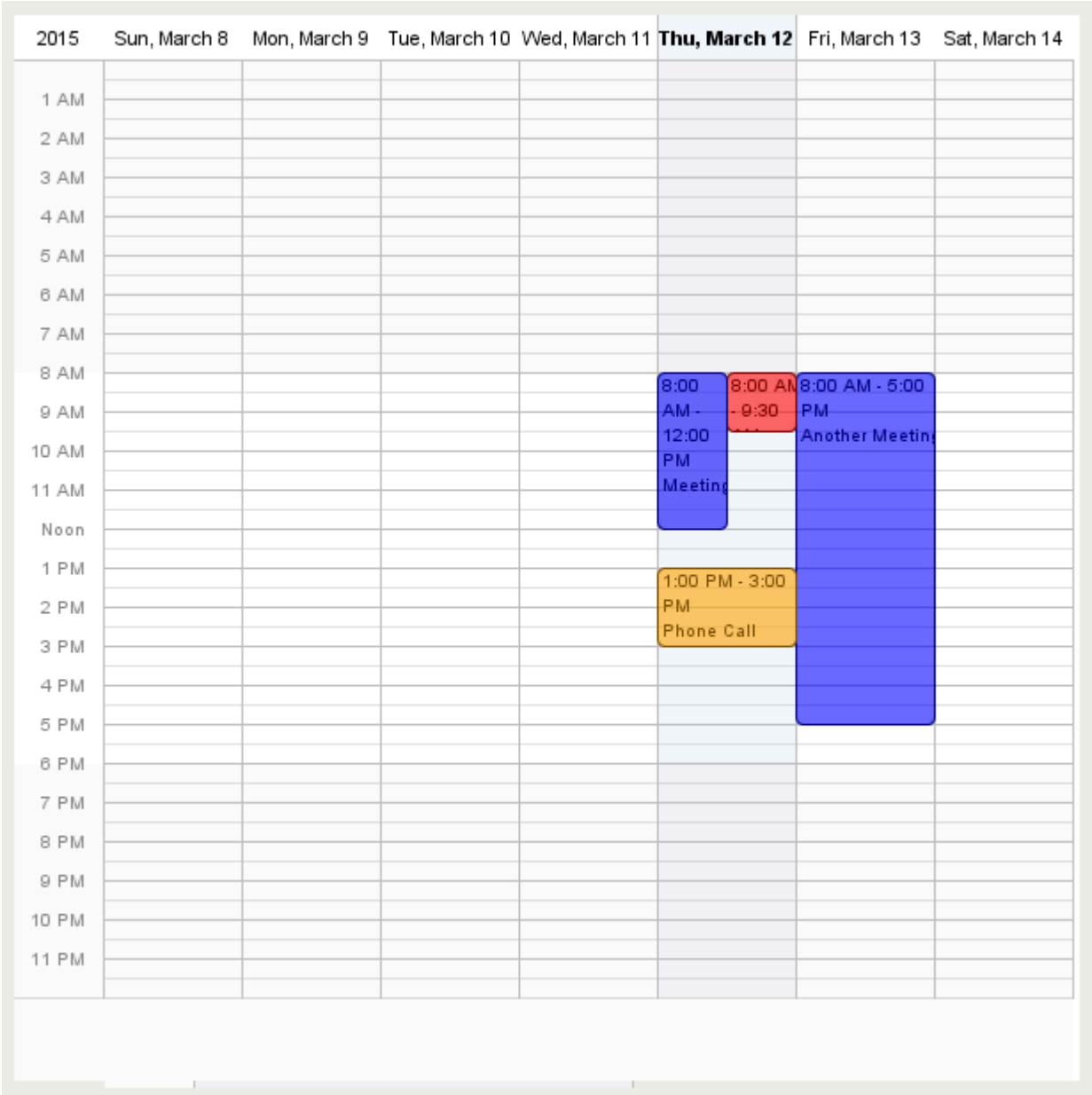
Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Week View



Component Palette Icon:



Description

Displays a full week's worth of events on a calendar. Configuration is achieved by populating the Calendar Events dataset. See the [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. NOTE that the border is unaffected by rotation.	Border	.border	Common
Calendar Background Color	The color of the calendar's background.	Color	.calendarBackground	Appearance
Calendar events	Contains the calendar events.	Dataset	.events	Data
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on	int	.dataQuality	Data

Name	Description	Property Type	Scripting	Category
	this component.			
Day	Set the calendar's day.	int	.day	Data
Day Outline Color	The color of the day's outline.	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.	Color	.hourForeground	Appearance
Hover Background Color	The background color of the hovered day and time.	Color	.hoverBackground	Appearance
Hovered Day	The calendar's hovered day.	String	.hoveredDay	Data

Name	Description	Property Type	Scripting	Category
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.	Color	.nonWorkingHourBackground	Appearance
Selected Background Color	The color of the selected day's background.	Color	.selectedBackground	Appearance

Name	Description	Property Type	Scripting	Category
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.	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.	Color	.weekDaysBackground	Appearance

Name	Description	Property Type	Scripting	Category
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.	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

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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Month View

March 2015						
Sunday	Monday	Tuesday	Wednesday	Thursday	Friday	Saturday
1	2	3	4	5	6	7
8	9	10	11	12	13	14
				● Meeting ● Email Customer ● Phone Call	● Another Meeting	
15	16	17	18	19	20	21
22	23	24	25	26	27	28
29	30	31	1	2	3	4
5	6	7	8	9	10	11

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. NOTE that the border is unaffected by rotation.	Border	.border	Common
Calendar Background Color	The color of the calendar's background.	Color	.calendarBackground	Appearance
Calendar events	Contains the calendar events.	Dataset	.events	Data
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Day Font	The font for the number representing	Font	.dayFont	Appearance

Name	Description	Property Type	Scripting	Category
	the day of the month.			
Day Foreground Color	The foreground color for days in this month.	Color	.dayOfMonthForeground	Appearance
Day Other Foreground Color	The foreground color for days not in this month.	Color	.dayOfMonthOtherForeground	Appearance
Day Outline Color	The color of the day's outline.	Color	.boxOutline	Appearance
Event Background Color	The background color of the selected event.	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	int	.displayMode	Appearance

Name	Description	Property Type	Scripting	Category
	the event highlight background color.			
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.	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.	Color	.monthHeaderForeground	Appearance
Hover Background Color	The background color of the hovered day.	Color	.hoverBackground	Appearance

Name	Description	Property Type	Scripting	Category
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
Selected Background Color	The color of the selected day's background.	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	Dataset	.styles	Appearance

Name	Description	Property Type	Scripting	Category
	styles.			
Today's Background Color	The color of the today's background.	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.	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.	Color	.weekDaysForeground	Appearance
Year	Set the calendar's year.	int	.year	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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Charts

The following components give you various charts for displaying data.

Easy Chart

Chart

Sparkline Chart

The sparkline chart is a minimalistic chart component that displays a line-chart history for a single datapoint.

Bar Chart

The Bar Chart is a very easy-to-use chart that provides a familiar bar representation of any numeric values.

Radar Chart

Radar charts, also known as web charts, spider charts, spider plots, and a few other names, display a dataset as ...

Status Chart

Pie Chart

The Pie Chart component displays a familiar-looking pie chart. A Pie Chart displays a list of named items, each of ...

Box and Whisker Chart

A Box and Whisker chart displays pertinent statistical information about sets of data.

Equipment Schedule

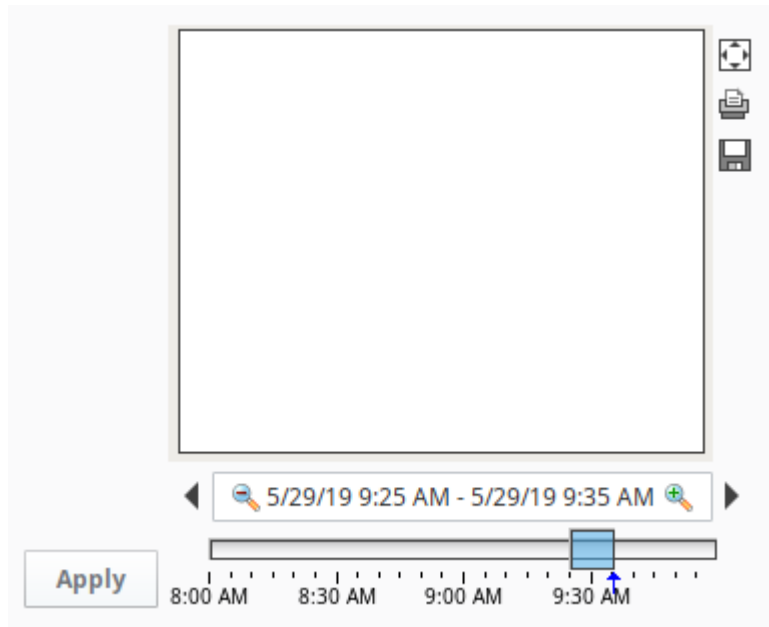


Gantt Chart

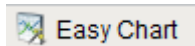
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Easy Chart



Component Palette Icon:



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 an SPC function such as the UCL (Upper Control Limit).

Modes: Realtime vs Historical

The Easy Chart can operate in 3 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 **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 of 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, etc is 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.

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

It 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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance

Name	Description	Property Type	Scripting	Category
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

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component.	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. NOTE that the border is unaffected by rotation.	Border	.border	Appearance
Box Fill	For historical-mode date range. The fill color for the selection box.	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

Name	Description	Property Type	Scripting	Category
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. 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.	int	.cursorCode	Common

Name	Description	Property Type	Scripting	Category
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.	Color	.editorBackgroundColor	Appearance
Date Editor Foreground	The foreground color for the date editor.	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	double	.digitalGap	Pen Style Options

Name	Description	Property Type	Scripting	Category
	use between digital pens.			
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.	Color	.foreground	Appearance
Gap Threshold	The relative threshold to use for determining continuity breaks for the 'Discontinuous Line' pen style.	double	.gapThreshold	Pen Style Options

Name	Description	Property Type	Scripting	Category
Gridline Color	The color of the gridlines.	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.	Color	.highDensityColor	Historical Range

Name	Description	Property Type	Scripting	Category
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	String	.maxSelectionSize	Historical Range

Name	Description	Property Type	Scripting	Category
	selected date 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

Name	Description	Property Type	Scripting	Category
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	boolean	.allowPenManipulation	Behavior

Name	Description	Property Type	Scripting	Category
	turn on and off pens.			
Plot Background	The background color for all plots, unless they override it.	Color	.plotBackground	Appearance
Plot Orientation	The plot orientation for all plots.	int	.plotOrientation	Layout
Plot Outline	The color to use for the plot outline.	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	Uncategoriz

Name	Description	Property Type	Scripting	Category
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	Uncategoriz
Selection Highlight	For historical-mode date range. The focus highlight color for the selection box.	Color	.selectionHighlight	Historical Range
Show Density	For historical-mode date range. If true, a data density histogram will be shown	boolean	.showHistogram	Historical Range

Name	Description	Property Type	Scripting	Category
	in the date 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 Butto
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 Butto
Show Save Button?	If true, a small save button will be	boolean	.showSave	Utility Butto

Name	Description	Property Type	Scripting	Category
	displayed next to the chart.			
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	String	.startupRange	Historical Range

Name	Description	Property Type	Scripting	Category
	startup mode is Automatic, this will be the starting range of time available for selection.			
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,	int	.tagHistoryResolution	Tag History

Name	Description	Property Type	Scripting	Category
	which uses the width of the chart.			
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 Configuratio
Tick Density	For historical-mode date range. This is multiplied by	float	.tickDensity	Historical Range

Name	Description	Property Type	Scripting	Category
	the width to determine the current ideal tick unit.			
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.	Color	.todayIndicatorColor	Historical Range
Total Datapoints	The number of datapoints being displayed by the graph. (Read only. Usable in bindings and scripting.)	int	.datapoints	Uncategoriz

Name	Description	Property Type	Scripting	Category
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	boolean	tagHistoryValidateScanclass	Tag History

Name	Description	Property Type	Scripting	Category
	bad data for the time periods where the scanclasses did not execute.			
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	boolean	.xAxisAutoRange	Behavior

Name	Description	Property Type	Scripting	Category
	and end date.			
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	double	.xTraceNumberFormatThreshold	Appearance

Name	Description	Property Type	Scripting	Category
	Number Format will be used.			
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

See the [Easy Chart Scripting Functions](#) page for the full list of scripting functions available for this component.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

Refer to the [Easy Chart Customizer](#) and the [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](#)

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Easy Chart Customizer

Easy Chart Customizer

Pens

Axes

Subplots

Dynamic Groups

Tag History Pens

Name	Tag Path	Color	Preview
Sine0	[~]Sine/Sine0		
Sine1	[~]Sine/Sine1		
Sine2	[~]Sine/Sine2		

Database Pens

Name	Y Column	Table	Color	Preview
RV	Tank_Number	Tank_History		

Calculated Pens

Name	Function	Pen	Parameter	Color	Preview
sine0 Sum	RunningSum	Sine0			

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.

Pens Tab

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.

For more information, refer to the following sections:

- **Easy Chart - Pen Names and Groups**
- **Easy Chart - Calculated Pens**
- **Using the Easy Chart**

Property	Description
	Add pen (Browse for Tags).
	Add a pen manually.
	Edit pen.
	Delete pen.
Edit Pen Panel	Contains all the editable fields for the pen row selected.
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.

Database Pens Properties

✖

Edit Pen

General

Name

Realistic Value

Enabled

☒ true

Hidden

☐ false

User Selectable

☒ true

Axis

Default Axis

▼

Subplot

1

▲▼

Group Name

▼

Digital Offset

☐ false

Style

Color

Style

Area

▼

Dash Pattern

Line Weight

3

▲▼

Shape

▼

Fill Shape?

☒ true

Labels

☐ false

Preview

Data

Value Column

Realistic0

▼

Table Name

realistic_table

▼

Time Column

t_stamp

▼

Datasource

<Default>

▼

↺

Where Clause

Run Diagnostics

OK

Cancel

Property	Description
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

✕

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

Area

▼

Dash Pattern

Line Weight

1

▲▼

Shape

▼

Fill Shape?

☒ true

Labels

☐ false

Preview

Data

Function

Constant

▼

Driving Pen

Sine0

▼

Constant Value

200

▲▼

OK

Cancel

Property	Description
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).

Axes Tab

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 Form Override	Specifies a number format pattern to use for tick labels. Leave blank for automatic number formatting.

 Edit Axis



General

Name

Sine 4

Label

Sine 4

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

OK

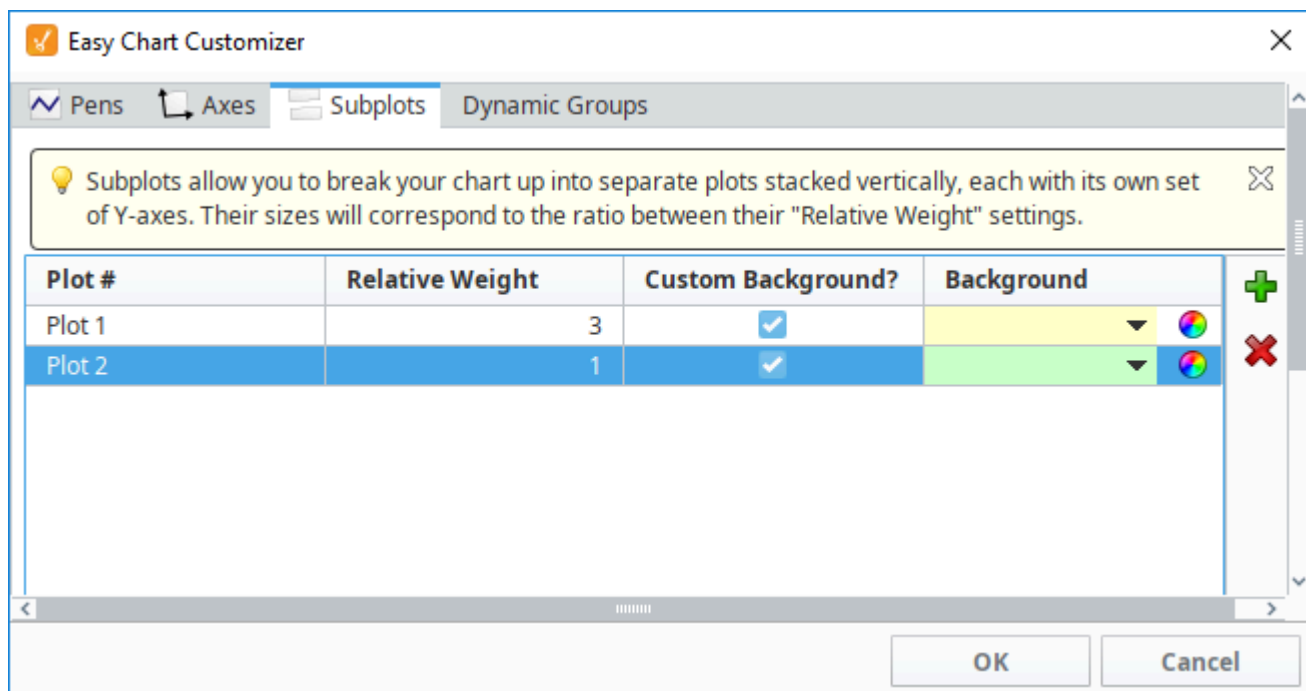
Cancel

For more information, refer to the [Easy Chart - Axes](#) page.

Subplots Tab

The Subplots 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.



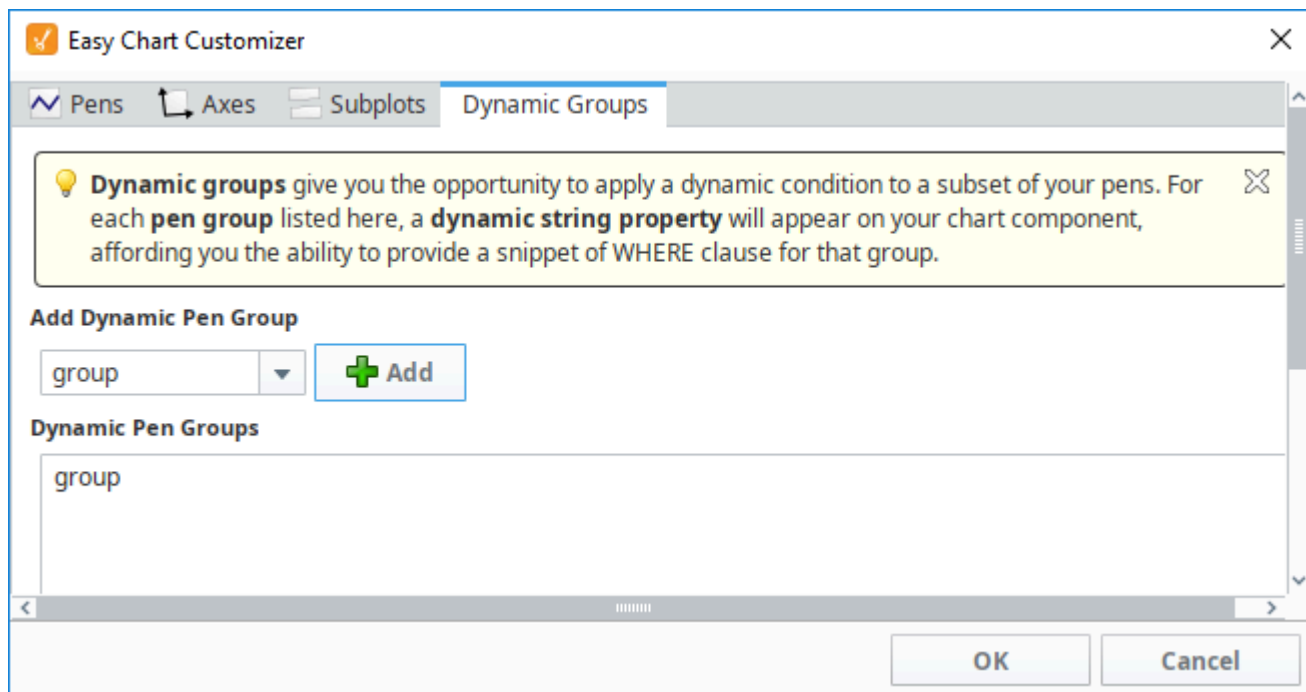
! IN THE PENS TAB

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](#).

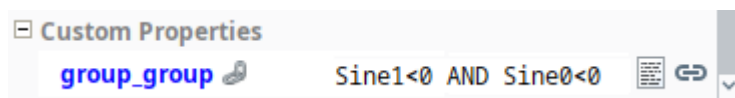
Dynamic Groups Tab

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.



Property Editor - Custom Properties - Where Clause for Dynamic Group Property

The dynamic string property "group_group" has a value of "Sine1<0 AND Sine0<0" which corresponds to the WHERE clause configured on the Dynamic Groups tab.



To learn more about Dynamic Groups, refer to the [Easy Chart](#) section.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Easy Chart - Scripting Functions

Scripting Functions

exportExcel(filename)

- Description
 - This function saves 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
- Scope
 - Client

print()

- Description
 - This function will print the chart.
- Parameters
 - Nothing
- Return
 - Nothing
- Scope

- Client

setMode(mode)

- 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
- Scope
 - Client

exportDatasets()

- 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.
- Scope
 - Client

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 behavior 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

configureChart

- 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
- Scope
 - Client

getXTraceLabel

- 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
- Scope
 - Client

onPowerTableRowsDropped

- 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 indices that were dragged, in the order they were selected.
 - Dataset **rowData** - A dataset containing the rows that were dragged.
- Return
 - Nothing
- Scope
 - Client

onTagsDropped

- 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
- Scope
 - Client

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)
```

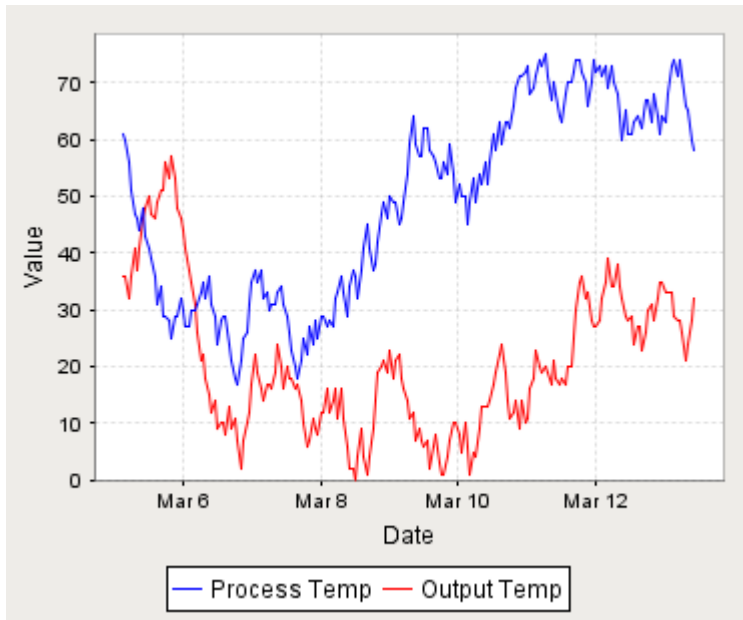
Example - Group Name

```
#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)
```

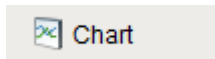
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Chart



Component Palette Icon:



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.

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 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 Binding. 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 Bindings](#).

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 [Chart Customizer](#)).

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected 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.	int	.cursorCode	Common
Data Quality	The data quality code for any tag	int	.dataQuality	Data

Name	Description	Property Type	Scripting	Category
	bindings on this component.			
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

Name	Description	Property Type	Scripting	Category
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

Name	Description	Property Type	Scripting	Category
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 1 subplot.	int	.subplotMode	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

See the [Chart Scripting Functions](#) page for the full list of scripting functions available for this component.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

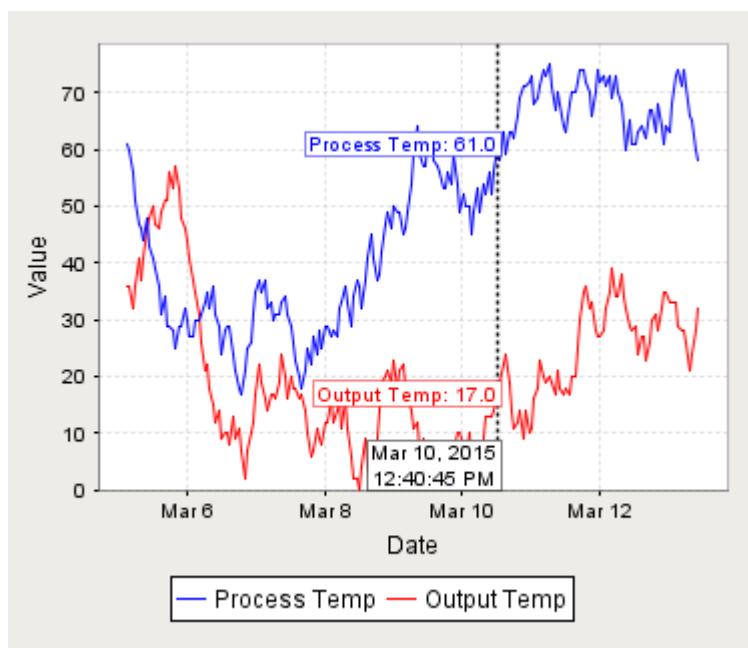
Customizers

The Chart component uses its own customizer called the [Chart Customizer](#). You 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 [Chart Customizer](#) section for property descriptions and examples of chart axis types.

- [Chart Customizer](#)
- [Component Customizers](#)
- [Understanding Component Customizers](#)

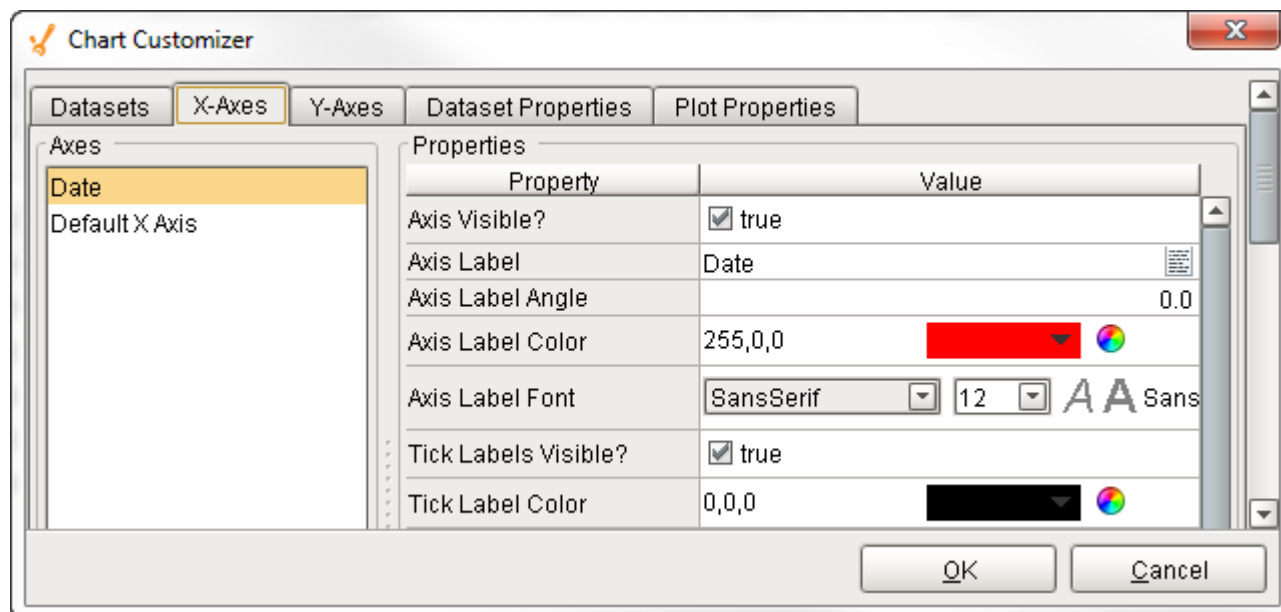
Example



This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Chart Customizer



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.

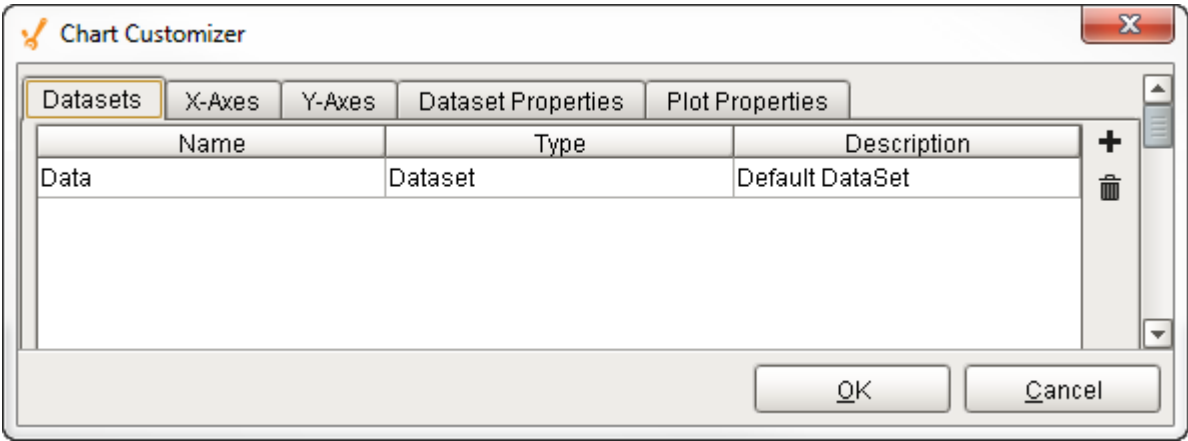
There are six types of axes to choose from when configuring a chart, each having its own list of properties: Number Axis, Date Axis, Category Axis, Logarithmic Axis, Elapsed Axis, and Symbols Axis. Most of the X and Y axes properties are used in the customizer, and some properties are specific to the 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.

Datasets Tab

The Dataset tab is where you set up, add, and remove datasets.



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.

X-Axes Tab

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

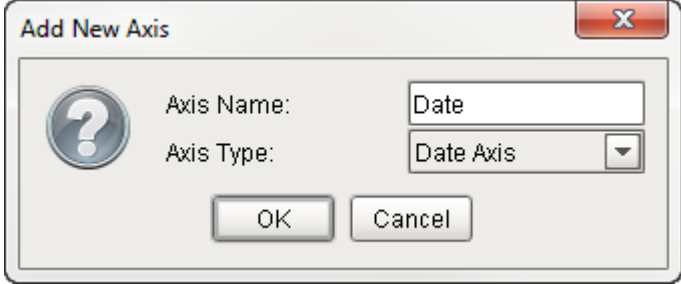
Default X Axis

Properties

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	Support 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. 	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

Property	Description	Support Axes Types
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

Property	Description	Support Axes Types
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

Property	Description	Support Axes Types
"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

Y-Axes Tab

The Y-Axes tab is where Y-Axis properties are configured. You can also add and delete Y axes here.

Chart Customizer

DatasetsX-AxesY-AxesDataset PropertiesPlot Properties

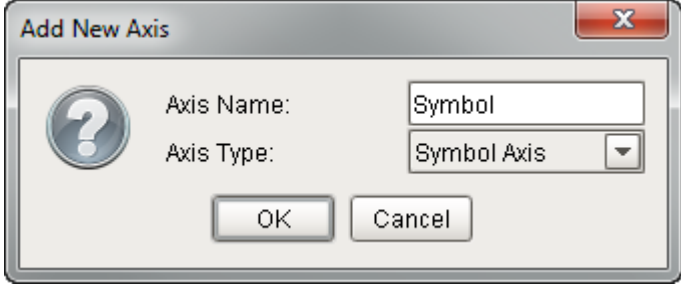
Axes

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	

OKCancel

Property	Description	Support Axes Types
	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. 	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

Property	Description	Support Axes Types
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 may 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

Property	Description	Support Axes Types
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

Property	Description	Support Axes Types
"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

Dataset Properties Tab

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.

Chart Customizer

DatasetsX-AxesY-AxesDataset PropertiesPlot Properties

Datasets

Data

Dataset Properties

X AxisDate

Y AxisDefault Y Axis

Subplot Number1

Enabled?☒

RendererXY Line/Shape Renderer

Dataset Renderer Properties

Property	Value
Series Colors	12345678
Type	Lines Only
Line Size	1.0
Dash Pattern	
Fill Shapes	☒ true
Shape Offset	0

OKCancel

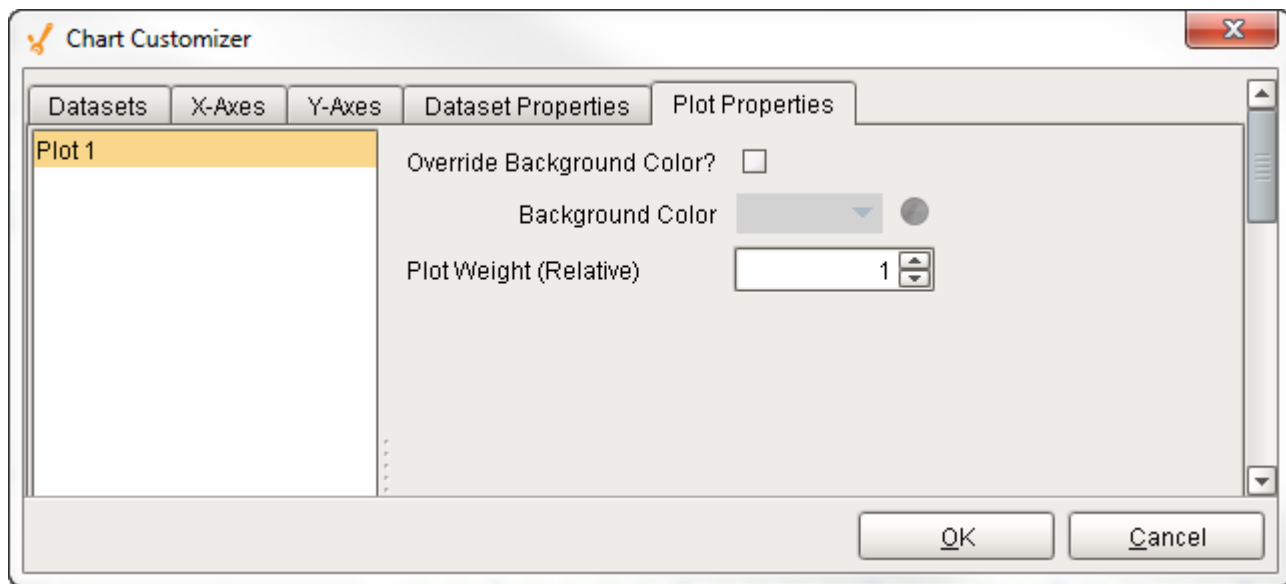
Property	Description	Support Axes Types
Dataset	Collection of data in tabular form. Data from the dataset drives the chart.	All
X Axis	Horizontal axis.	All
Y Axis	Vertical axis.	All
Subplot Number	Number of plot areas on one chart.	All
Enabled	If true, the chart is displayed with the selected renderer 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. Offset values and respective values are	Offset

Property	Description	Support Axes Types
	listed below. • 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
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

Property	Description	Support Axes Types
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

Plot Properties Tab

The Plot Properties tab allows you to break up the chart plot area into multiple distinct subplots.



Property	Description	Axes Types Supports
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

- [Chart](#)
- [Component Customizers](#)
- [Understanding Component Customizers](#)

Axis Type Examples

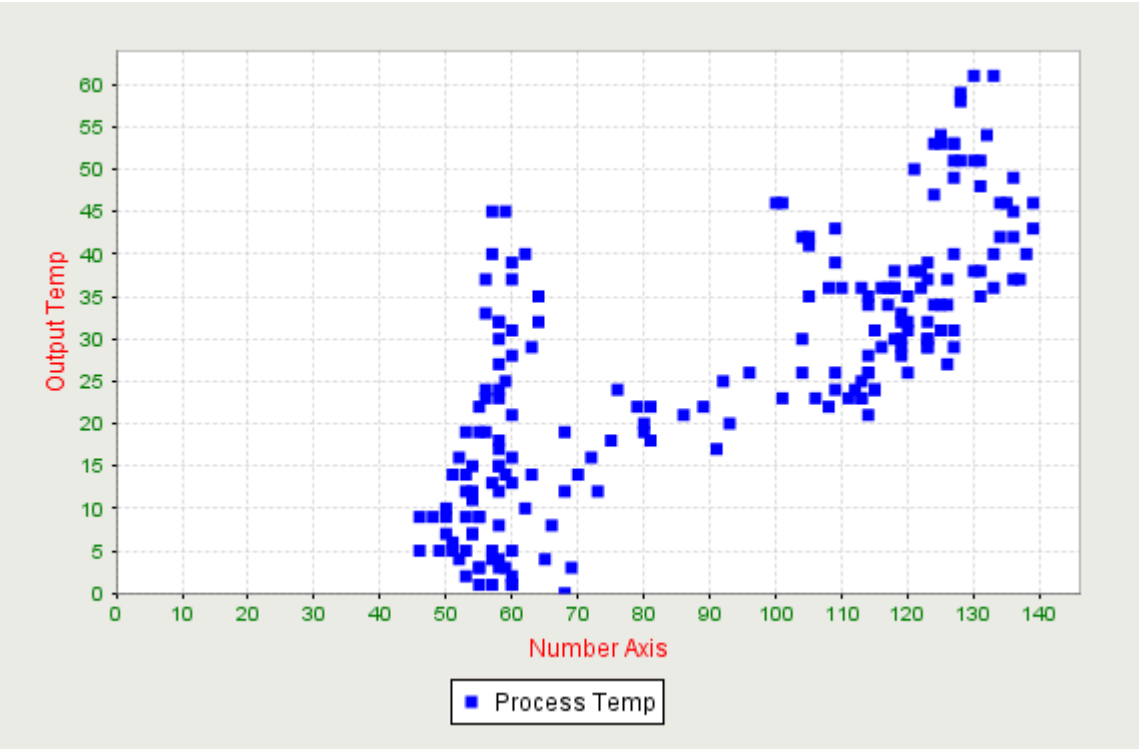
The Chart Customizer has six different axis types to choose from when configuring a chart, each with its own list of properties.

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	

Property Name	Value
Axes	Number
X Axis Label	Number Axis
Axis Label Color	Red
Tick Label Color	Green
Y-Axes Tab	

Property Name	Value
Axes	Default Y Axis
Y Axis Label	Output Temp
Axis Label Color	Red
Tick Label Color	Green
Dataset Properties Tab	

Property Name	Value
X Axis	Number
Y Axis	Default Y Axis
Renderer	XY Line/Shape Renderer
Type	Shapes Only

Dataset 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: ----

OK

Cancel

Date Axis Chart

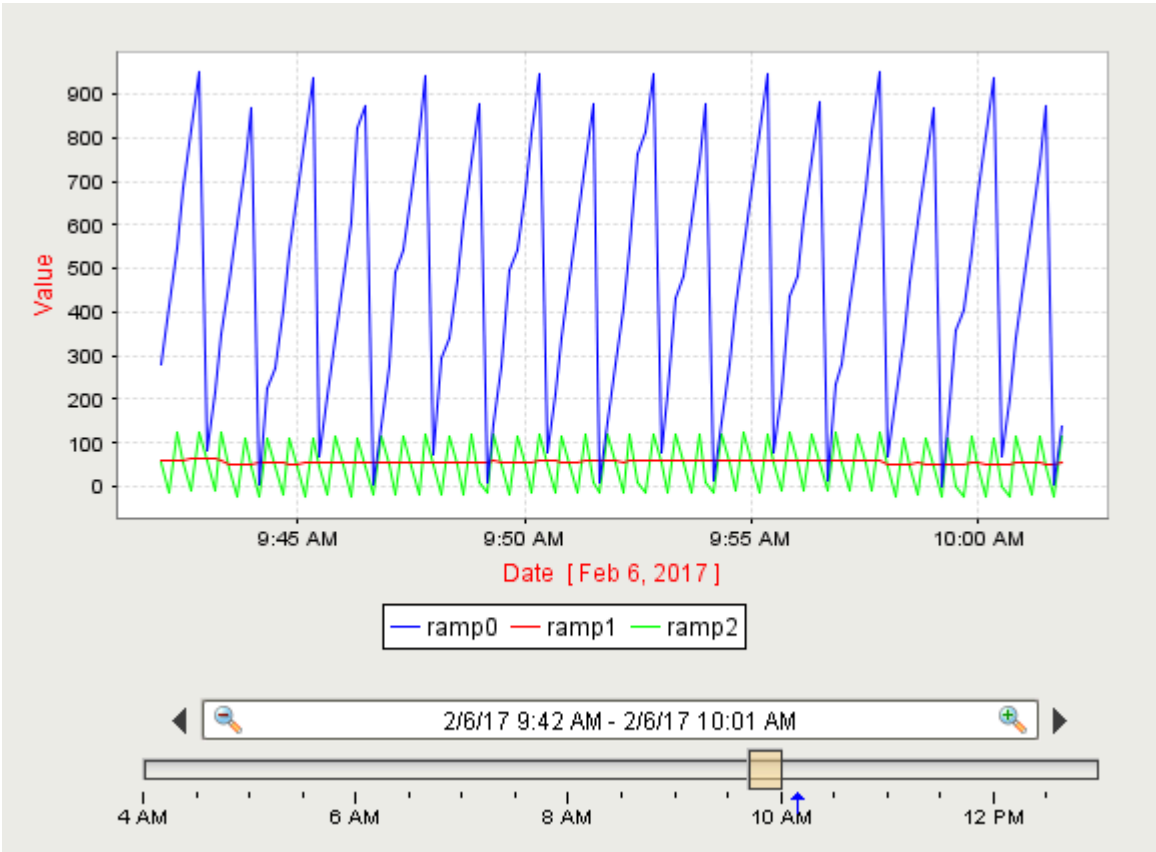


Chart Customizer Property Settings

Datasets Tab

Property Name	Value
Datasets	Data
X-Axes Tab	

Property Name	Value
Axes	Date
Axis Label	Date
Axis Label Color	Red

Y-Axes Tab

Property Name	Value
Axes	Default Y Axis
Axis Label	Value
Axis Label Color	Red

Dataset Properties Tab

Property Name	Value
Datasets	Data
X Axis	Date
Y Axis	Default Y Axis
Renderer	XY Line/Shape Renderer
Type	Lines Only

Dataset Property Dataset

Dataset Viewer

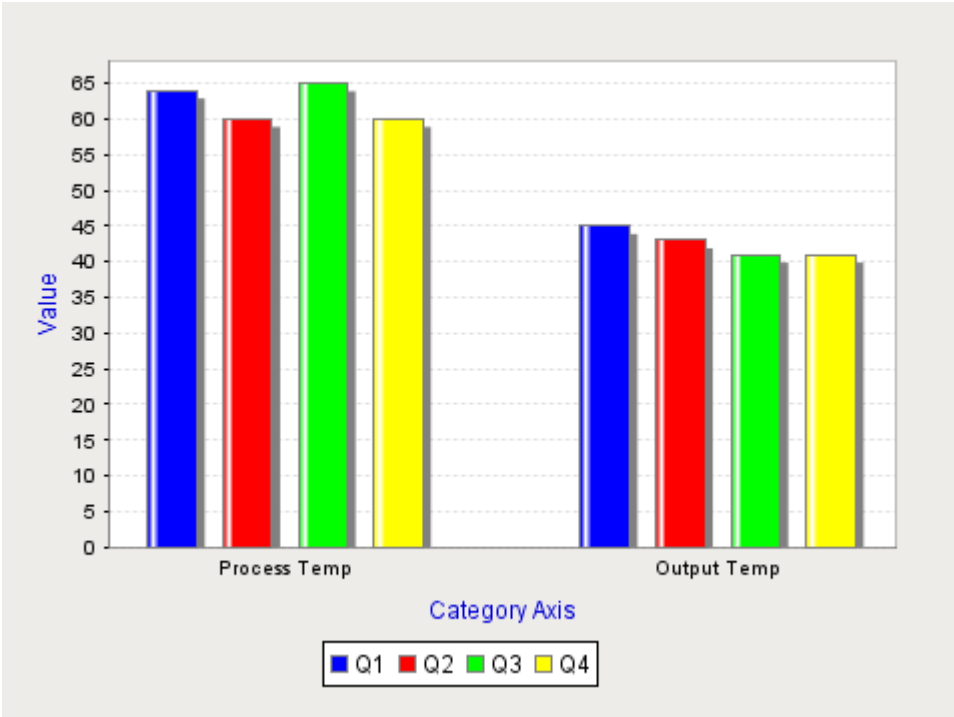
t_stamp	ramp0	ramp1	ramp2
02/03/2017 11:00:00	436.427	77.32	11.427
02/03/2017 11:00:10	571.093	78.32	146.093
02/03/2017 11:00:20	794.493	78.337	34.427
02/03/2017 11:00:30	837.787	78.34	12.787
02/03/2017 11:00:40	972.453	79.34	147.453
02/03/2017 11:00:50	105.8	79.35	80.8
02/03/2017 11:01:00	236.467	77.35	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

Column Name: ---- Column Type: ----

OK

Cancel

Category Axis Chart



Behavior

Property	Value
Chart Type	Category

Chart Customizer Property Settings

Datasets Tab

Property Name	Value
Datasets	Data
X-Axes Tab	

Property Name	Value
Axes	Category
Axis Label	Category Axis
Axis Label Color	Blue

Y-Axes Tab

Property Name	Value
Axes	Default Y Axis
Axis Label	Value
Axis Label Color	Blue

Dataset Properties Tab

Property Name	Value
Datasets	Data
X Axis	Category
Y Axis	Default Y Axis
Renderer	Category Bar Renderer
Type	Bar

Dataset Property Dataset

 Dataset Viewer

Month	Process Temp	Output Temp
Q1	64	45
Q2	60	43
Q3	65	41
Q4	60	41

Column Name: ---- Column Type: ----

OK

Cancel



Logarithmic Axis Chart

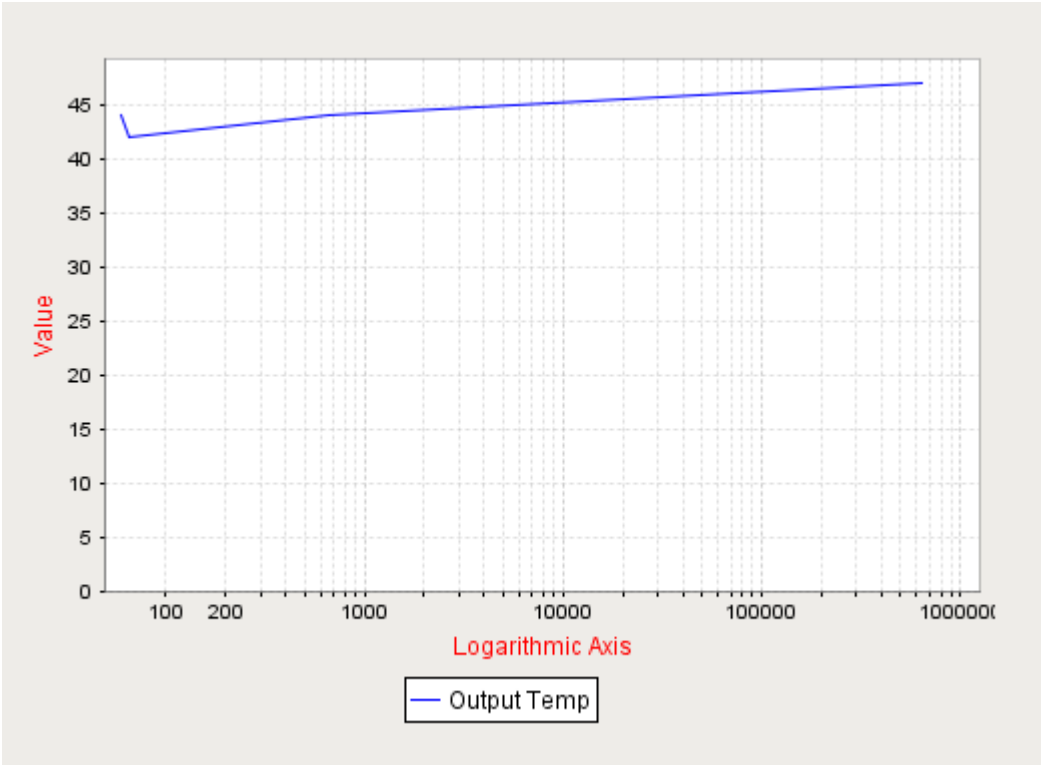


Chart Customizer Property Settings

Datasets Tab

Property Name	Value
Datasets	Data
X-Axes Tab	

Property Name	Value
Axes	Logarithmic
Axis Label	Logarithmic Axis
Axis Label Color	Red

Y-Axes Tab

Symbols Axis Chart

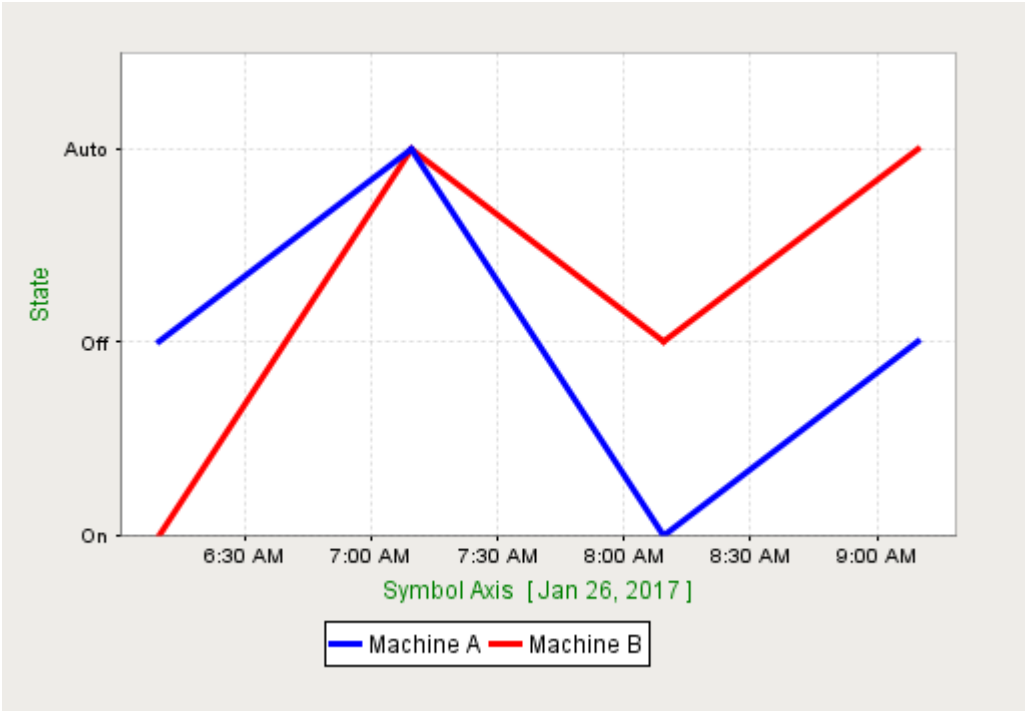


Chart Customizer Property Settings

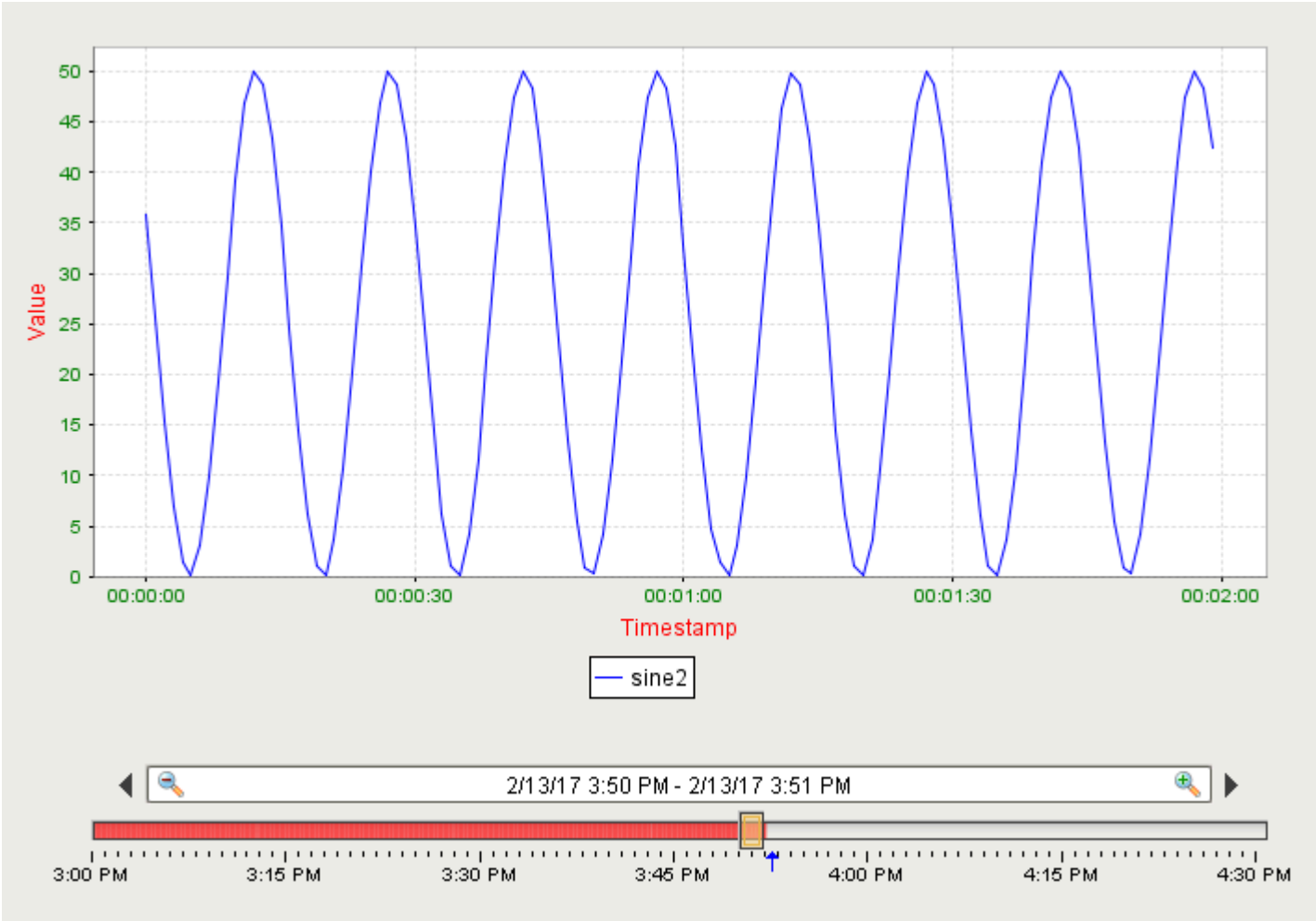
Datasets Tab

Property Name	Value
Datasets	Data
X-Axes Tab	

Property Name	Value
Axes	Default Axis
Axis Label	Symbol Axis
Axis Label Color	Green

Y-Axes Tab

Elapsed Time Axis Chart



Binding Type

- Database
- SQL Query

Chart Customizer Property Settings

Datasets Tab

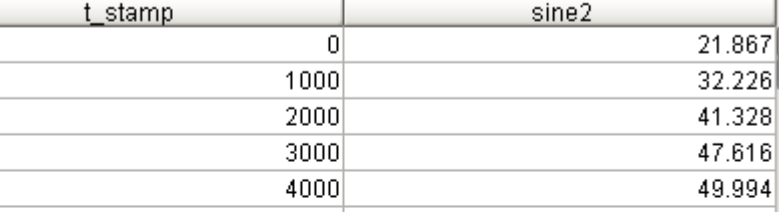
Property Name	Value
Dataset	Data
X-Axes Tab	

Property Name	Value
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	

Property Name	Value
Axes	Default Y Axis
Axis Label	Value
Axis Label Color	Red
Tick Label Color	Green
Dataset Properties Tab	

Property Name	Value
Datasets	Data
X Axis	Elapsed
Y Axis	Default Y Axis
Renderer	XY Line/Shape Renderer
Type	Lines Only

Dataset Property Dataset



Dataset Viewer

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

Column Name: ---- Column Type: ----

OK Cancel

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Chart Scripting Functions

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

configureChart

- 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
- Scope
 - Client

getXTraceLabel

- 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

- Scope

- Client

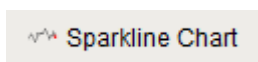
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Sparkline Chart



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 High and Range Low properties.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected 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.	int	.cursorCode	Common

Name	Description	Property Type	Scripting	Category
Data	The history data to draw in the sparkline chart.	Dataset	.data	Data
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	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.	Color	.desiredRangeColor	Appearance
First Marker Color	The color of the first value marker.	Color	.firstMarkerColor	Markers
First Marker Size	The size of the first value marker.	double	.firstMarkerSize	Markers

Name	Description	Property Type	Scripting	Category
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.	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.	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.	Color	.foreground	Appearance

Name	Description	Property Type	Scripting	Category
Line Width	The width of the sparkline.	float	.lineWidth	Appearance
Low Marker Color	The color of the low value marker.	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
Range High	A fixed value for the upper edge of the chart. If left blank (null), the upper range will be	Double	.rangeHi	Data

Name	Description	Property Type	Scripting	Category
	calculated automatically.			
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

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

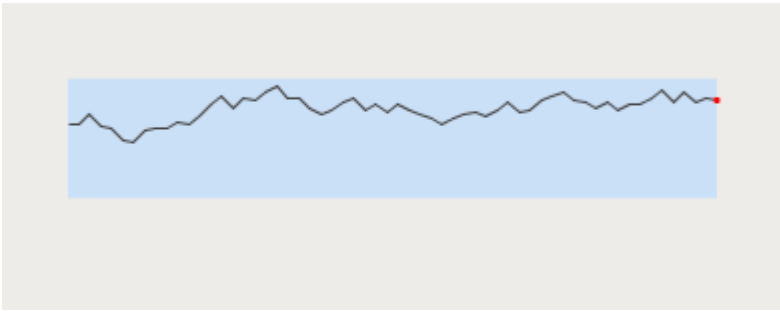
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

This component does not have any custom properties.

Examples

Sparkline Chart with Low and High Limits

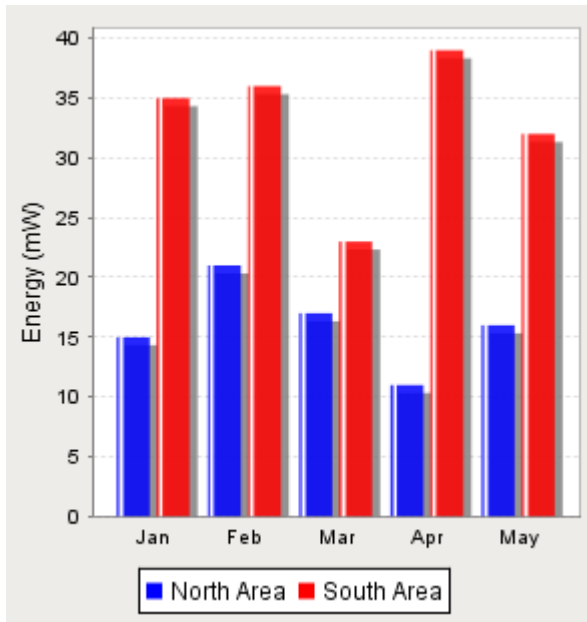


Property Name	Value
Range High	100
Range Low	0
Desired High	75
Desired Low	40

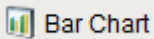
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Bar Chart



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.



TIP

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.	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. NOTE that 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.	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.	Color	.catAxisTickColor	Axes

Name	Description	Property Type	Scripting	Category
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.	int	.cursorCode	Common
Data	The data driving the chart.	Dataset	.data	Data
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Extract Order	Controls whether the first row defines the categories or the series.	int	.extractOrder	Data

Name	Description	Property Type	Scripting	Category
Foreground Transparency	The transparency of the pie (useful for 3D pies).	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
Series Colors	The sequence of colors used for series in the bar chart.	Color[]	.seriesColors	Appearance

Name	Description	Property Type	Scripting	Category
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.	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.	Color	.valAxisTickColor	Axes
Value Axis Tick Font	The font for the value axis' ticks.	Font	.valAxisTickFont	Axes

Name	Description	Property Type	Scripting	Category
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

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

getBarColor

- 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
- Scope
 - Client

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

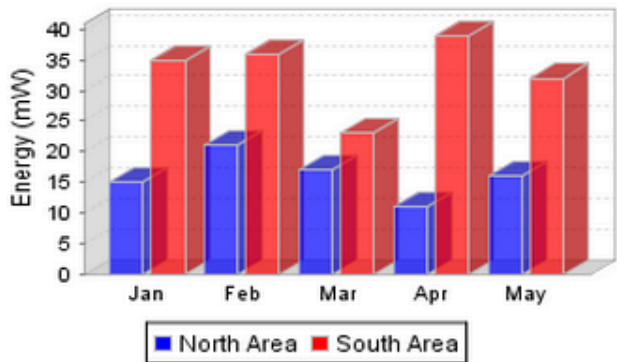
Examples

Extract Order Example

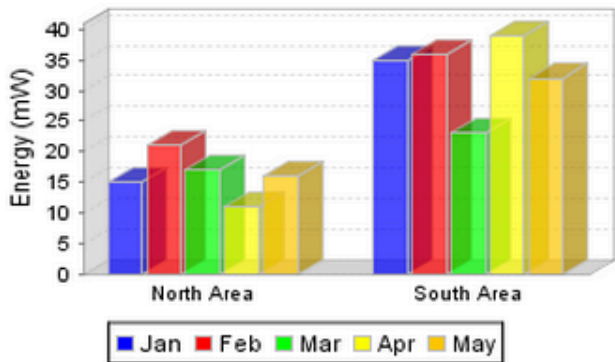
The following two charts demonstrate the effects of the extract order property on the given dataset

Label (String)	North Area (Integer)	South Area (integer)
Jan	15	35
Feb	21	36
Mar	17	23
Apr	11	39
May	16	32

Extract Order: By Row



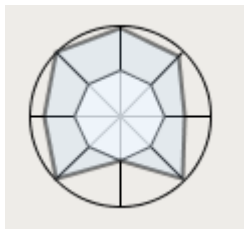
Extract Order: By Column



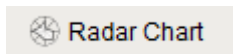
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Radar Chart



Component Palette Icon:



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.

Properties

Name	Description	Property Type	Scripting	Category
Actual Fill Color	Fill color for the actual polygon.	Color	.actualFillColor	Appearance
Actual Stroke Color	Stroke color for the actual polygon.	Color	.actualStrokeColor	Appearance
Actual Stroke Width	Stroke width for the actual polygon.	float	.actualStrokeWidth	Appearance
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that 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.	int	.cursorCode	Common
Data	Contains the datapoints for the radar plot. Each row represents a spoke	Dataset	.data	Data

Name	Description	Property Type	Scripting	Category
	and point on the polygon.			
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Desired Fill Color	Fill color for the desired polygon.	Color	.desiredFillColor	Appearance
Desired Stroke Color	Stroke color for the desired polygon.	Color	.desiredStrokeColor	Appearance
Desired Stroke Width	Stroke width for the desired polygon.	float	.desiredStrokeWidth	Appearance
Name	The name of this component.	String	.name	Common
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.	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

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

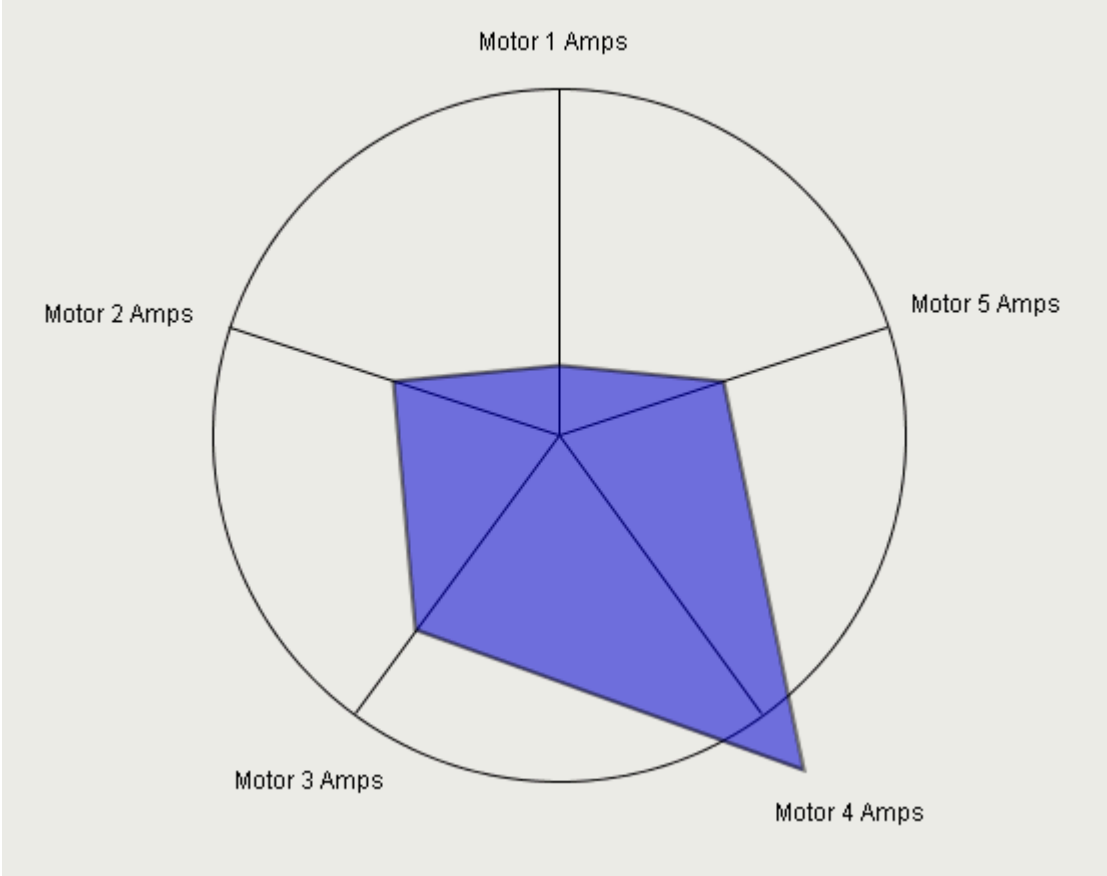
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples

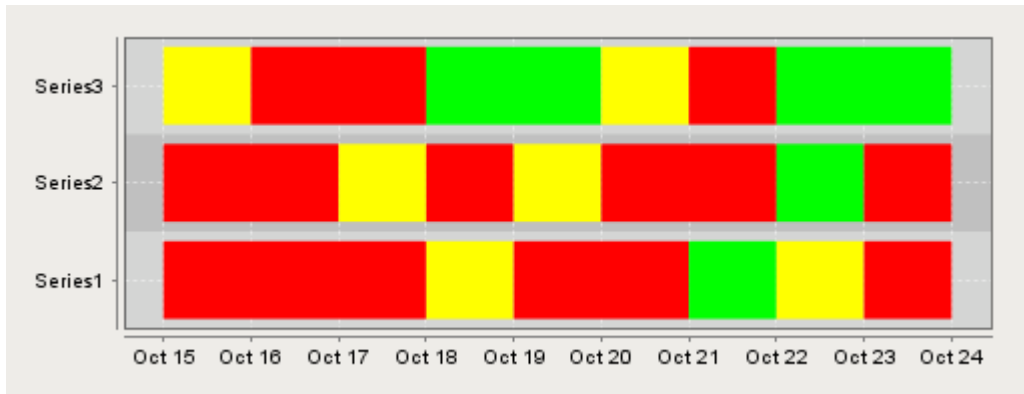
The following shows that the Motor 4 Amps are higher than the maximum allowed for Motor 4. In this example each motor has an independent min and max value for its amps. The radar chart allows an operator to quickly assess a group of independent variables to determine if anything is out of its specifications.



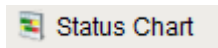
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Status Chart



Component Palette Icon:



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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that 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.	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	int	.dataFormat	Data

Name	Description	Property Type	Scripting	Category
	represents one series in the chart. In "tall" format, the first column is a timestamp, the second column is a series name.			
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Date Style	The style to display dates in. For international support.	int	.dateStyle	Appearance
Domain Axis Color	Color used on the domain axis.	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
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common

Name	Description	Property Type	Scripting	Category
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
Range Axis Color	Color used on the range axis.	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

Name	Description	Property Type	Scripting	Category
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.	Color	.titleColor	Appearance
Title Font	Font on the chart title.	Font	.titleFont	Appearance

Name	Description	Property Type	Scripting	Category
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

See the [Status Chart Scripting Functions](#) page for the full list of scripting functions available for this component.

Event Handlers

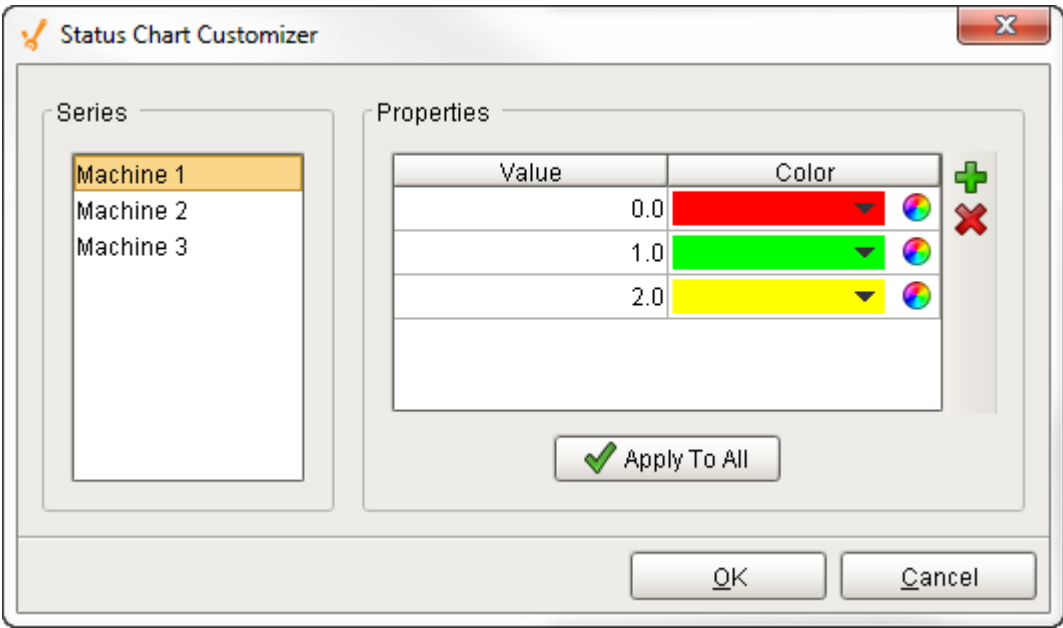
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

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

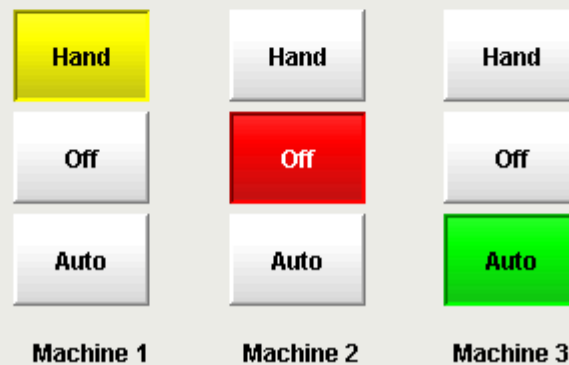
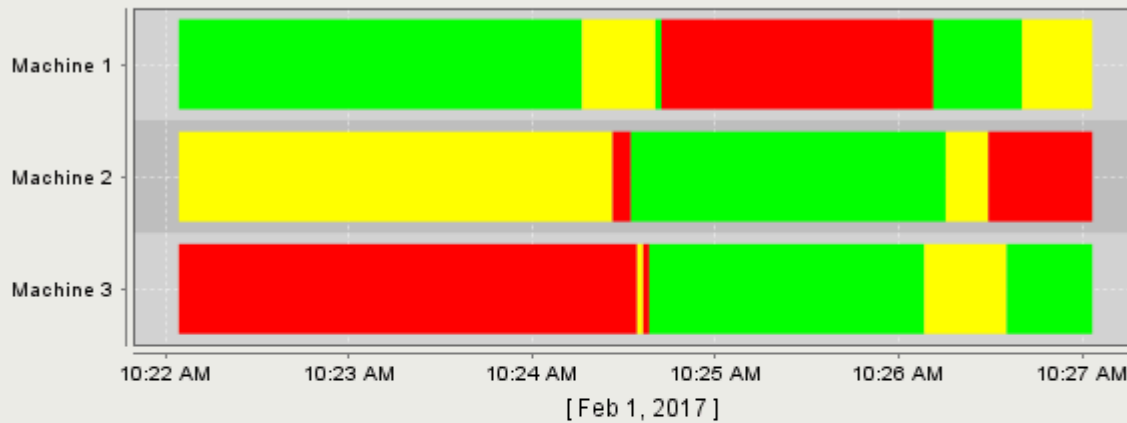
Property	Description
Series	Selectable list of all objects in the Series Data property. • Wide format: Each non-timestamp column. • Tall format: each unique value in the Name column.
Properties Table	The number-to-color mapping for the selected Series.
Value	A numeric value to match against.
Color	The color to display for the given value.
Apply To All	Set all of the Series mappings to the currently selected mapping.



Examples

This example uses the Status Chart to display the state of each of the three machines over the last 5 minutes. 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.

Machine Status



Series Data - Dataset Viewer

The Dataset Viewer window displays a table of series data for three machines. The table has four columns: 't_stamp', 'Machine 3', 'Machine 2', and 'Machine 1'. The data is organized into a grid with 10 rows of timestamps and 4 columns of machine data. The 'Machine 3' column contains values 0, 0, 0, 0, 0, 0, 0, 0, 0, 0. The 'Machine 2' column contains values 2, 2, 2, 2, 2, 2, 2, 2, 2, 2. The 'Machine 1' column contains values 1, 1, 1, 1, 1, 1, 1, 1, 1, 1. The 't_stamp' column contains timestamps from 02/01/2017 10:22:04 to 02/01/2017 10:22:13. The window also includes a 'Column Name' and 'Column Type' field, and 'OK' and 'Cancel' buttons.

t_stamp	Machine 3	Machine 2	Machine 1
02/01/2017 10:22:04	0	2	1
02/01/2017 10:22:05	0	2	1
02/01/2017 10:22:06	0	2	1
02/01/2017 10:22:07	0	2	1
02/01/2017 10:22:08	0	2	1
02/01/2017 10:22:09	0	2	1
02/01/2017 10:22:10	0	2	1
02/01/2017 10:22:11	0	2	1
02/01/2017 10:22:12	0	2	1
02/01/2017 10:22:13	0	2	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.

 Dataset Viewer

X

SeriesName	Value	Color
Series1	0	
Series1	1	
Series1	2	
Series2	0	
Series2	1	
Series2	2	
Series3	0	
Series3	1	
Series3	2	

Column Name: ---- Column Type: ----

OK

Cancel

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Status Chart Scripting Functions

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

getToolTip

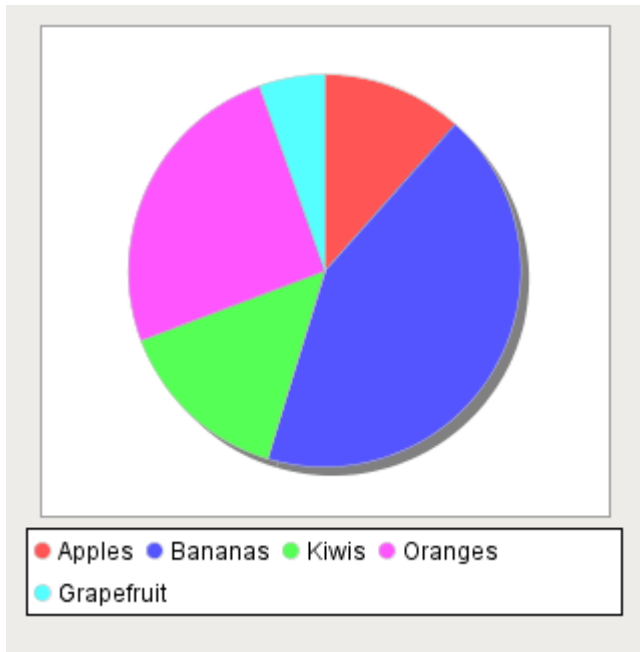
- Description
 - Return a formatted tool tip String
- Parameters
 - Component **self**- A reference to the component that is invoking this function.
 - int **seriesIndex**- The series index corresponding to the column in the series dataset.
 - int **selectedTimeStamp**- The time stamp corresponding to 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**
- Scope
 - Client

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Pie Chart



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](#) 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 tables show 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

New in 7.9.8

The 3D? Property on the the Pie Chart has been removed as of 7.9.8. Instead, the **Style** property should be use to configure a 3D Pie Chart. The depreciated property is listed further down this page.

Name	Description	Property Type	Scripting	Category
3D Depth Factor	The depth of a 3D pie as a factor of the chart height.	double	.depthFactor	Appearance
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Border	The border surrounding this component. NOTE that 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
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data	The data driving the chart.	Dataset	.data	Data
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data

Name	Description	Property Type	Scripting	Category
Enforce Circularity?	If true, the pie cannot be an oval, even if the overall chart is.	boolean	.circular	Appearance
Extract Order	Controls whether or not a pie plot views columns as pies, or rows.	int	.extractOrder	Data
Foreground Transparency	The transparency of the pie (useful for 3D pies).	double	.foregroundAlpha	Appearance
Label Font	The font for labels items, if there are labels.	Font	.labelFont	Appearance
Label Format	Formatting String. '0' is the wedge name, '1' is the value, '2' is the percent.	String	.labelFormat	Appearance
Labels?	Should labels be displayed near sections?	boolean	.labels	Appearance
Legend Font	The font for legend items, if there is a legend.	Font	.legendFont	Appearance
Legend?	Should there be an item legend	boolean	.legend	Appearance

Name	Description	Property Type	Scripting	Category
	below the chart?			
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 Colors	The colors to use for the pie wedge outlines.	Color[]	.outlineColors	Appearance
Outline Stroke	The width for the section outline stroke.	float	.outlineStroke	Appearance
Plot Background	The background color for all plots, unless they override it.	Color	.plotBackground	Appearance
Rotation	Draw the wedges clockwise or counter-clockwise from the starting angle?	int	.rotation	Appearance
Section Colors	The colors to use for the pie wedge fills.	Color[]	.sectionColors	Appearance

Name	Description	Property Type	Scripting	Category
Selected Wedge	The currently selected wedge. (Read only. Usable in bindings and scripting.)	String	.selectedData	Uncategorized
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	.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
Starting Angle	The start angle to draw the pie wedges.	int	.startAngle	Appearance
Style	Style of pie chart, standard, 3D, or ring.	int	.style	Appearance
Title Font	The font for the chart's title.	Font	.titleFont	Appearance

Name	Description	Property Type	Scripting	Category
Tooltip Format	Formatting String. '0' is the wedge name, '1' is the value, '2' is the percent.	String	.tooltipFormat	Appearance
Tooltips?	Should tooltips be displayed when the mouse hovers over sections?	boolean	.tooltips	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

The property below is deprecated, and should not be used.

Name	Description	Property Type	Scripting	Category
3D?	Enables 3D rendering. Deprecated. Use Style property instead.	boolean	.threeDimensional	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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)

Examples

Code Snippet

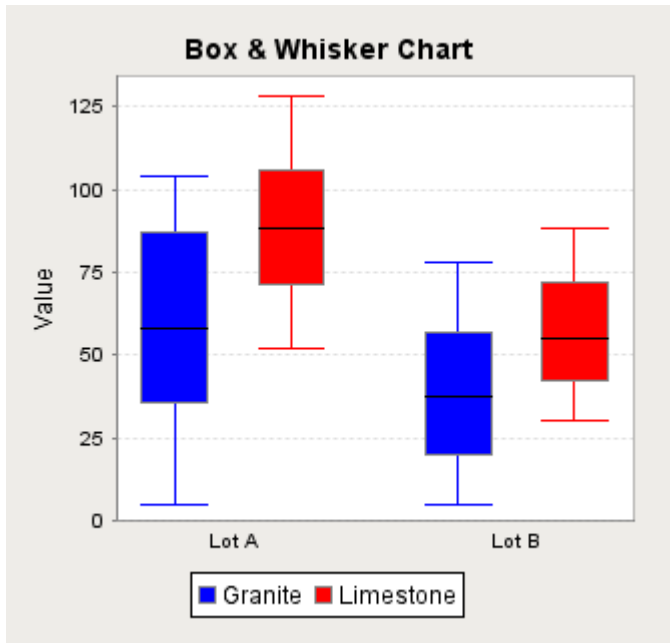
#The following code will print named and value of the selected wedge to the console.
#Alternatively, this can be used to write to a custom property of a table that is used to create the 'Where' clause of a SQL query that populates a table.

```
selectedWedge = event.source.selectedData  
print selectedWedge
```

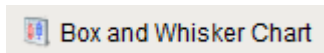
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Box and Whisker Chart



Component Palette Icon:



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.

Properties

Name	Description	Property Type	Scripting	Category
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Border	The border surrounding this component. NOTE that 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.	int	.cursorCode	Common
Data	The data driving the chart.	Dataset	.data	Data
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	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

Name	Description	Property Type	Scripting	Category
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
Series Colors	The colors to paint each box in a series.	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

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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

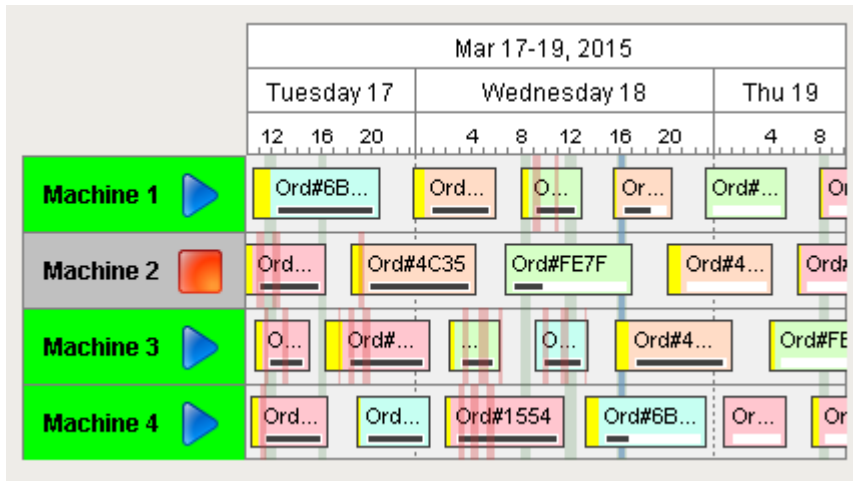
Customizers

This component does not have any custom properties.

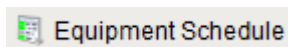
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Equipment Schedule



Component Palette Icon:



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 to 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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Border	The border surrounding this component. NOTE that 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.	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

Name	Description	Property Type	Scripting	Category
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.	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.	Color	.headerTextColor	Appearance
Items	The cells, or equipment items, to have their schedules displayed.	Dataset	.items	Data

Name	Description	Property Type	Scripting	Category
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.	Color	.progressBackground	Appearance
Progress Bar Border	The border color for the event progress bars.	Color	.progressBorder	Appearance
Progress Bar Fill	The color for 'done' portion the event progress bars.	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.	Color	.scheduleBackground	Appearance
Scheduled Events	The scheduled events for all configured items.	Dataset	.scheduledEvents	Data

Name	Description	Property Type	Scripting	Category
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

See the [Equipment Schedule Scripting Functions](#) page for the full list of scripting functions available for this component.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Equipment Schedule Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

onBackgroundDragged

- 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
- Scope
 - Client

onEventClicked

- 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
- Scope
 - Client

onEventDropped

- 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
- Scope
 - Client

onEventPopupTrigger

- 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
- Scope
 - Client

onEventResized

- 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 **oldEndDate** - 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
- Scope
 - Client

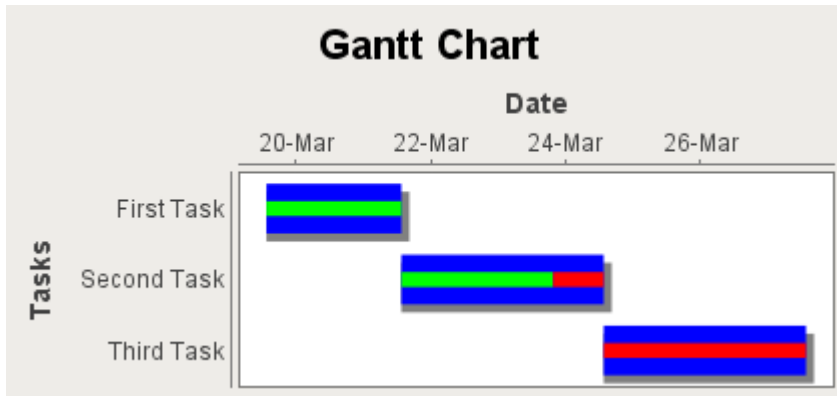
onPopupTrigger

- 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
- Scope
 - Client

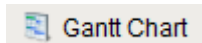
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Gantt Chart



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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Axis Font	The font for axis labels.	Font	.axisLabelFont	Appearance
Border	The border surrounding this component. NOTE that 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.	Color	.completeColor	Appearance
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data	The data driving the chart.	Dataset	.data	Data
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	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.	Color	.incompleteColor	Appearance

Name	Description	Property Type	Scripting	Category
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
Task Axis Title	A task label to display on the Axis Title.	String	.taskAxisTitle	Appearance
Task Color	The main color to draw tasks.	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

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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

This component does not have any custom properties.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Containers

The following components give you the ability to group and display components.

Container

All components are always inside of a container, except for the special Root Container of each window.

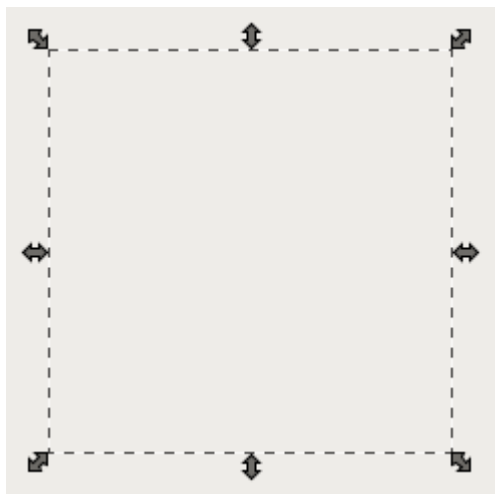
Template Repeater

Template Canvas

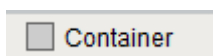
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Container



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.

 INFO

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.	Color	.background	Appearance
Border	The border surrounding this component. Note that 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.	int	.cursorCode	Commo
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data

Name	Description	Property Type	Scripting	Category
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
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

Name	Description	Property Type	Scripting	Category
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

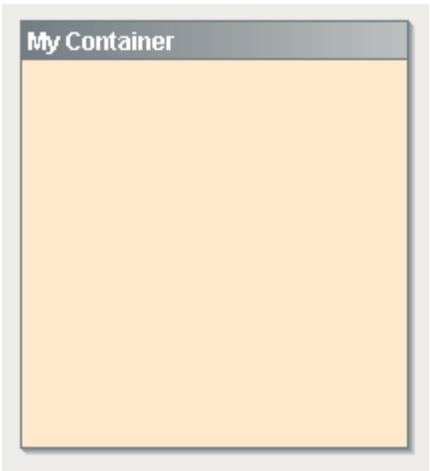
Extension Functions

This component does not have extension functions associated with it.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page.

Examples



Property Name	Value
Border	Titled
Background Color	255,232,204

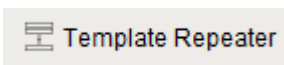
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Template Repeater



Component Palette Icon:



Description



INDUCTIVE
UNIVERSITY

Template Repeater

[Watch the video](#)

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.

! INFO

An Example of configuring the Template Repeater can be found on the [Template Repeater Component](#) page.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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

Name	Description	Property Type	Scripting	Category
Name	The name of this component.	String	.name	Common
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
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	boolean	.visible	Common

Name	Description	Property Type	Scripting	Category
	hidden.			

Scripting

See the [Template Repeater Scripting Functions](#) page for the full list of scripting functions available for this component.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page.

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
```

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Template Repeater Component



INDUCTIVE
UNIVERSITY

Template Repeater

[Watch the video](#)

Using the Template Repeater

The **Template Repeater** component lets you easily create multiple instances of a master template for display on the HMI. Each instance shown in the Template Repeater has the same look, feel, and functionality of the master template. The instances can be arranged vertically, horizontally, or in a "flow" layout, which can either be top-to-bottom or left-to-right. If there are too many instances to fit, a scrollbar is added to the display. The Template Repeater also gives you the ability to pass parameters to each instance of the template, making the templates dynamic.

The Template Repeater can create multiple template instances in two different ways, which will also affect how it passes parameters to those instances. The first is **Count** mode, which will allow you to specify how many times the Template Repeater will repeat a template. It will then use the index number of each template as a parameter that it will pass into the template. The second method is **Dataset** mode, where each row of a dataset will be a new template instance and each column will be a parameter that will be passed into the template. This is useful if you have multiple parameters that need to be passed into a template. See the [Template Repeater](#) page for a more detailed description of this property and how it works. We will go over both methods of using the repeater below.

Creating a Template

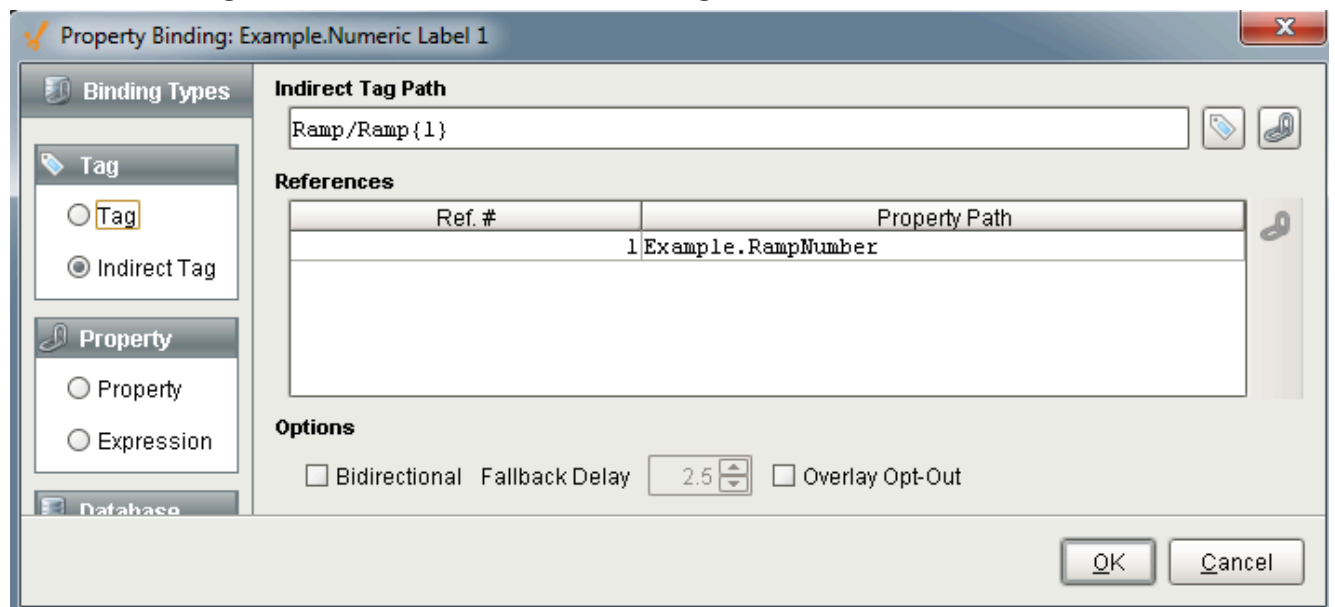
Before we use the Template Repeater, we need to first have a template that we need multiple copies of. I will be using the Ramp Tags in the Generic Simulator device built in to Ignition in my template. My template will have a label at the top with the name of the ramp that I come up with, as well as display the ramp number, and the ramp value. To do this our template needs to have two parameters. One called **RampNumber**, that will get passed the ramp number to be displayed on the template, and for use in an indirect tag binding to get the value. The second parameter will be a **RampName**, that will

be passed in a string name that is the name of the ramp that I made up. The steps for making this template are listed below, or you can skip ahead to the next section if you are familiar enough with making templates.

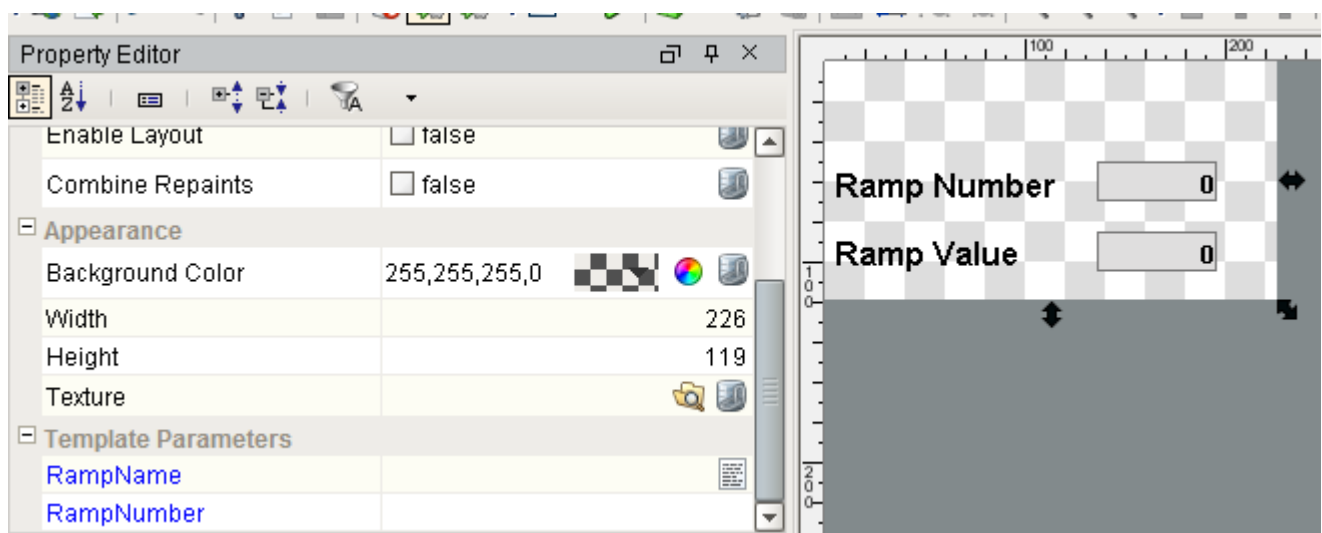
To Make a Template:

1. In the Templates section of our Project Browser, create a new template. I have named mine Example. See the [Templates](#) section for a more detailed description on what templates are and how to use them.
2. We are going to want to pass in a value to the template, so we need to create a Template Parameter.
 - i. Right click on the Root Container, and add a Template Parameter called **RampNumber**, and make it an Integer type.
 - ii. Add a second Template Parameter called **RampName**, and make it a String type.
3. Next, we want to add some components to our Template.
 - i. We will need two Numeric Labels and one Label. I also added two extra label components, and manually typed in some static text into their **Text** property. In one, I entered "Ramp Number" and in the other, "Ramp Value".
 - ii. Resize the Components so that they are easy enough to read. Place the blank Label component at the top of the Template, and place each Label that we wrote text into next to a Numeric Label.
4. Finally, we want to setup some bindings on our Template.
 - i. Using a [property binding](#), bind the **Text** property of the blank Label component at the top of the Template to the **RampName** Template Parameter.
 - ii. Using a property binding, bind the **Value** property of the Numeric Label next to the Label that says "Ramp Number" and bind it to the **RampNumber** Template Parameter.
 - iii. Using an [indirect Tag binding](#), bind the **Value** property of the Numeric Label next to the Label that says "Ramp Value" and bind it to a Ramp Tag that we have in our Tag browser, using the **RampNumber** Template Parameter as the indirect reference for the number in the

name of the Tag. It should look similar to the image below.



5. Our template is now finished and ready to use. See the image below for an idea of what mine looks like. Your should look similar. Note the two Template Parameters, as well as the placement of the components, including the label at the top that can't be seen because there is no text in it.

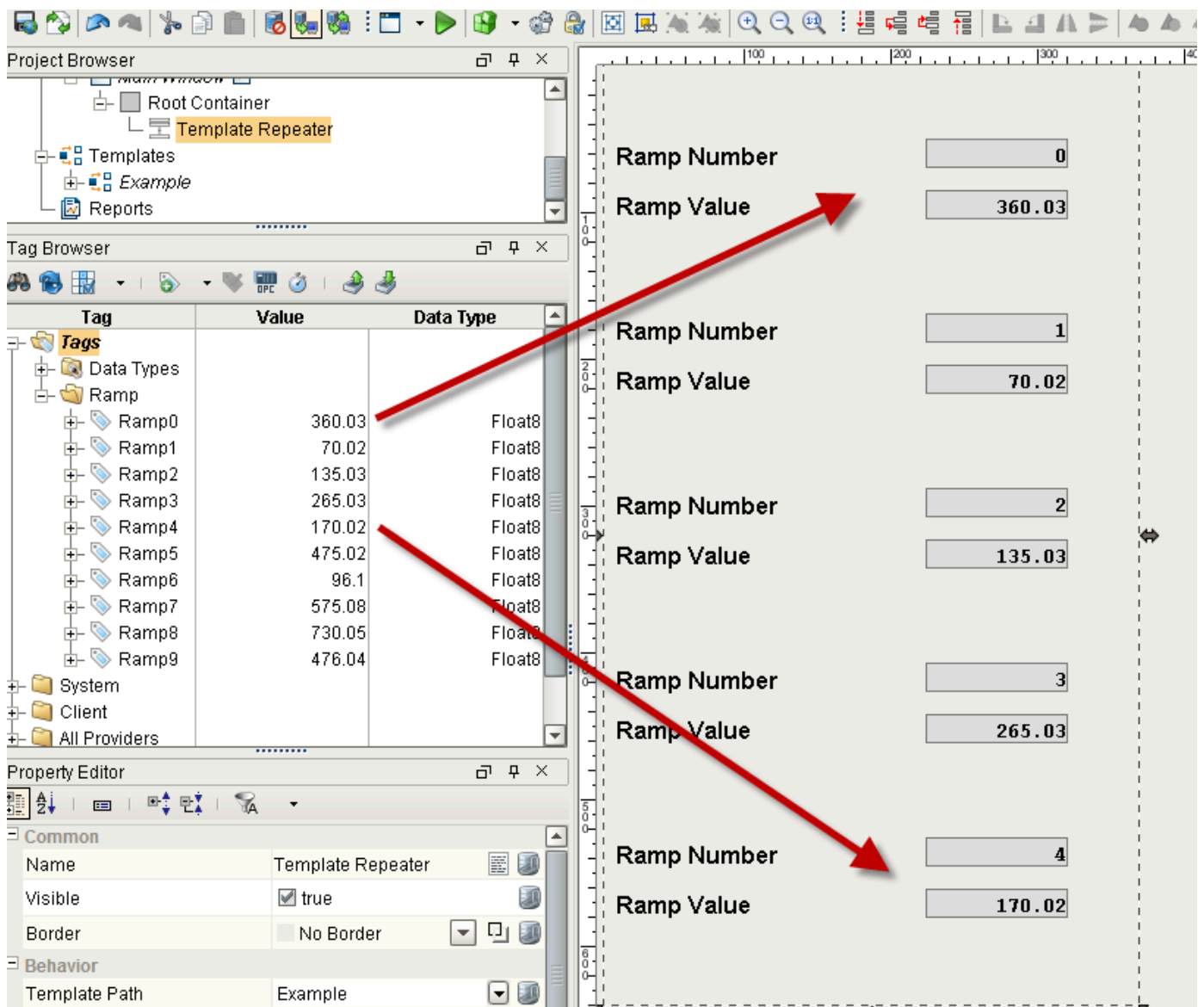


Using the Template Repeater with Count Mode

Now let's take a look at how to use the Template we just made in a Template Repeater.

1. On a new window, create a good size Template Repeater that is taller than it is long, which is located in the Containers section of the Component Palette.
2. With the Template Repeater selected, find the **Template Path** property, and find the name of the template that we just created in the drop down menu.
3. With the Template Repeater still selected, find the **Index Parameter Name** property. Here we can enter in the name of the property we want to use as an index.

- i. It should be located in the **Behavior** section of the Property Editor. If you can't see it, be sure to set the Filter Properties at the top of the Property Editor to 'All'.
 - ii. In the **Index Parameter Name**, we want to enter in '**RampNumber**', which was the name of one of our Template Parameters from our Template.
4. Ensure that the **Repeat Behavior** property of the Template Repeater is set to Count.
5. In the **Repeat Count** property of the Template Repeater, set the value to '5'.
6. Just like that, you should see your Template repeated 5 times within the Template Repeater. You should notice that our Ramp Number value in the Template corresponds to the index value of the Template, 0 through 4 because it is 0 based. You should also see that the value in our Ramp Value component corresponds to the value of one of our Ramp Tags.



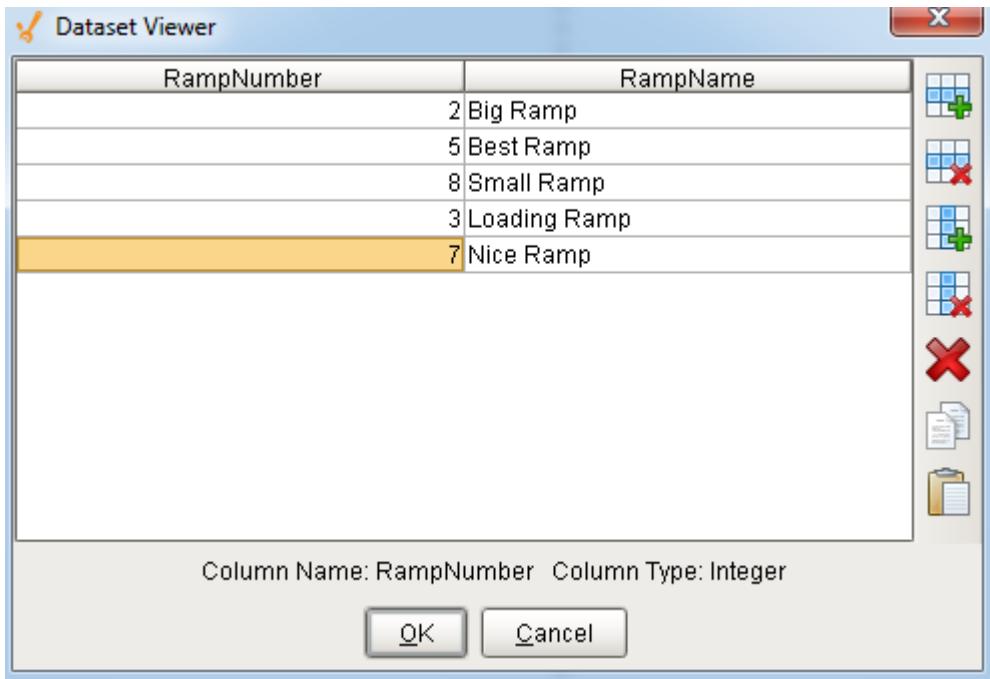
That is all there is to it! You can modify the Repeat Count to change how many times the Template gets repeated.

Using the Template Repeater with Dataset Mode

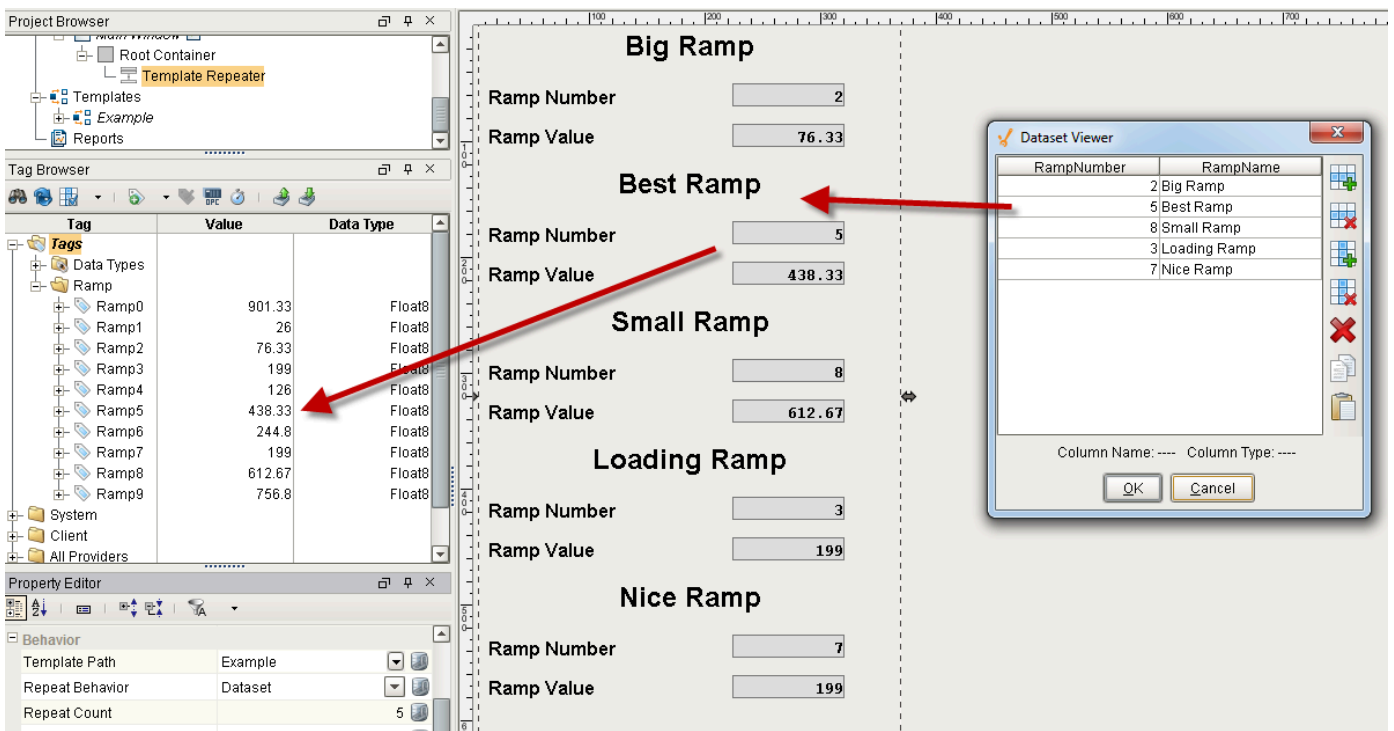
You may have noticed in our previous example that although we were able to successfully pass in the index parameter into our Template, we were not passing anything into the **RampName** Template Parameter and because of this, the blank label at the top of our Template remained blank. This is because **Count** mode only allows a single index parameter to be passed in. However, since we want to pass in two parameters, we can instead use **Dataset** mode. In addition, because we are going to be defining a dataset to pass in parameters, we don't have to use a 0 indexed parameter, and can instead use whatever values we want. We will use the same Template and Template Repeater from before.

1. On the Template Repeater, set the Repeat Behavior to Dataset. All of the Templates that were previously in there should disappear. Don't worry, they will come back!
2. Click the Dataset Viewer () button on the **Template Parameters** property of the Template Repeater. You should see a new popup that allows you to build a dataset.
3. First we need to add some columns by clicking the Add Column () button.
 - i. First, make a column called '**RampNumber**'. Make sure the name of the column exactly matches the name of the Template Parameter called **RampNumber**. It is case sensitive! Make this an integer type.
 - ii. Make another column called '**RampName**', again making sure the name matches exactly to our **RampName** Template Parameter. Make this a string type.
4. Now lets add some rows by clicking the Add Row () button. I have clicked it 5 times to add 5 rows.
 - i. We can add in some values for the **RampNumber** column.
 - a. Since we are defining all of the values, we don't have to start with a **RampNumber** of 0. In fact, we don't even need to have them be in sequence! We do need to make sure that it corresponds to one of our ramps though, so enter in a value 0 through 9 for the **RampNumber** column for each row. I have use 2, 5, 8, 3, 7 in that order.
 - ii. Next, we can add some values for the **RampName**.
 - a. This can be whatever we want, since we are going to be giving unique names to each of our Ramps. Feel free to call your Ramps whatever you like, or you can copy what I have done in the image below.

5. When finished, the dataset should look something like the image below.



6. Click the OK button to save your dataset. Notice how we now have our 5 templates back, and they are using the parameters that we are passing in from the dataset we just created. Notice how the Templates are receiving the parameters in the order that you made them in the dataset. We can also see that each Template instance corresponds to a specific ramp.



Dataset mode is great when you need to pass in multiple parameters, or if you have a single parameter that is not zero based. In addition to manually specifying values like we did in the example, you can instead setup a binding on the **Template Parameters** property to something like a database table. This allows you to easily modify the templates that are being displayed in the window simply by modifying the database table.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Template Repeater Scripting Functions

This page details the various component and extension functions available for Vision's Template Repeater component.

Component Functions

.getLoadedTemplates()

- 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
- Scope
 - Client

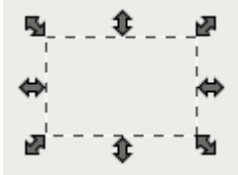
Extension Functions

This component does not have extension functions associated with it.

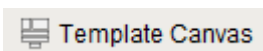
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Template Canvas



Component Palette Icon:



Description



INDUCTIVE
UNIVERSITY

Template Canvas

[Watch the video](#)

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	Type	Scripting	Category
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Layout Constraints	The overall layout constraints for the canvas.	String	.layoutConstraints	Behavior
Name	The name of this component.	String	.name	Common
Scroll Behavior	Controls wich direction(s) the canvas will scroll in.	int	.scrollBehavior	Behavior
Templates	A dataset containing a row per template to instantiate.	Dataset	.templates	Data
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

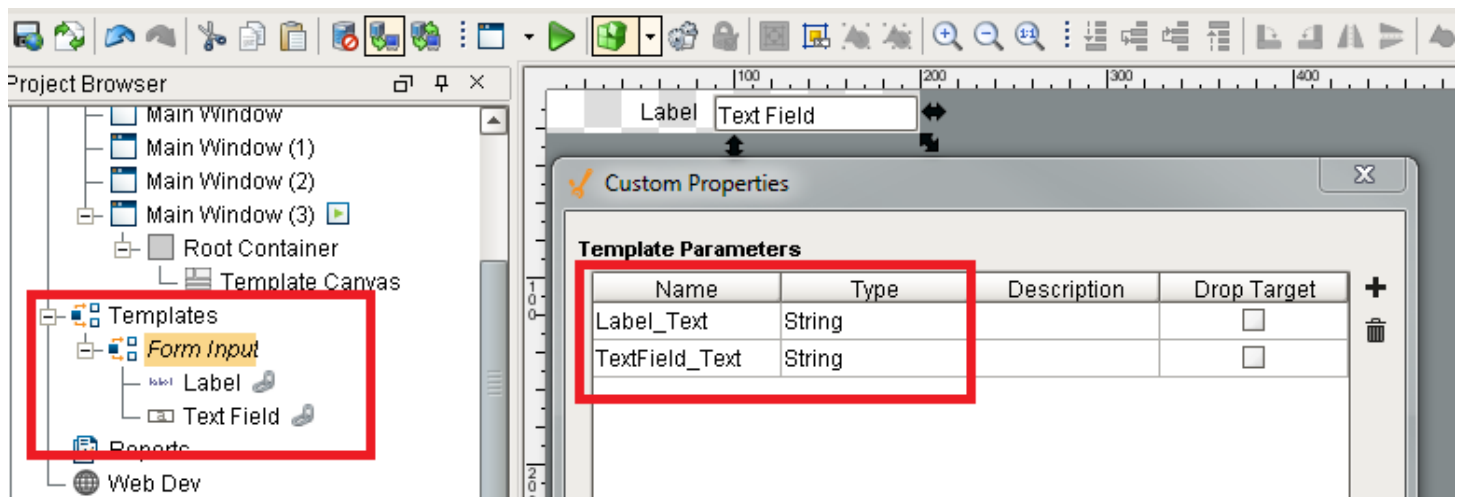
Template Canvas Walkthrough

The following example will create a form for users to input data.

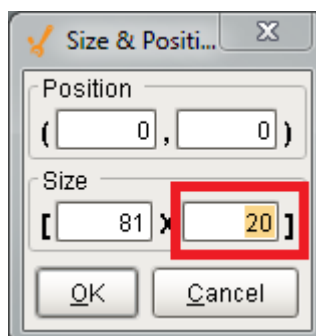
Step 1: Template Creation and Absolute Positioning

The example uses a template named Form Input. Form Input contains a Label component and Text Field component. Form Input also has two Template Parameters: Label_Text and TextField_Text

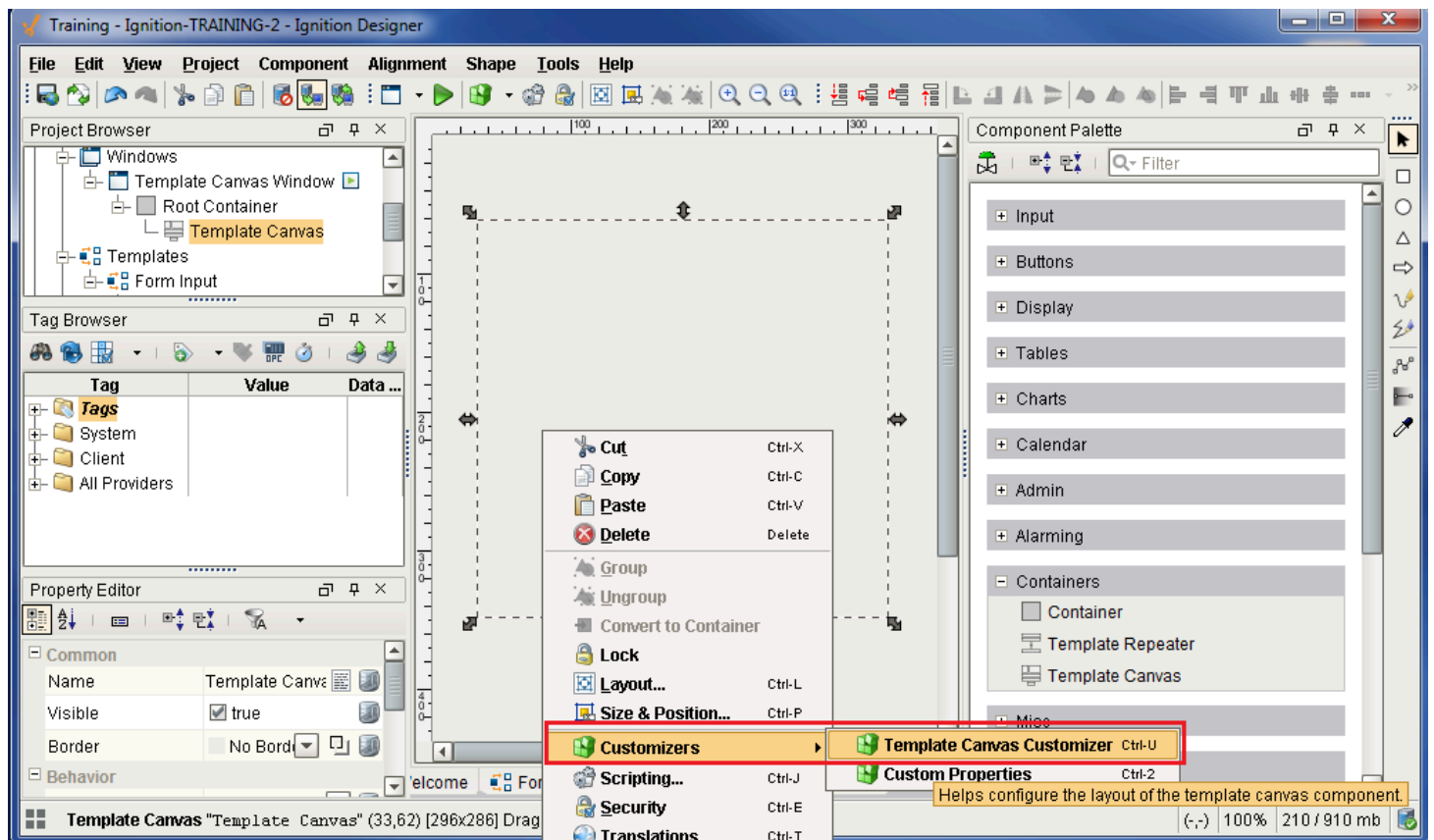
- The Text property on the Label component is bound to the Label_Text parameter
- The Text property on the Text Field component is bound to the TextField_Text parameter **bidirectionally**. This will allow the user to write to this property, and this makes reading the value later on easier.



Both Form Input and the components inside had their height set to 20 by right clicking on the components and selecting **Size and Position...** (Ctrl-P)



Once the template is in place, create a new window, place a **Template Canvas Component** down, and open the Template Canvas Customizer from either the right-click menu or Ctrl-U



Once open, the customizer allows us to specify the location of each template instance as well the value of any template parameters. Use the following values for the first instance of Form Input:

- Name: First Name
- Template: Form Input
- Position: Absolute, (0,0) [200x20]
- Parameters: Label_Text = First Name, TextField_Text = "" (blank)

Template Canvas Customizer

Instances

Add/Edit Instance

Name
First Name ✓

Template
Form Input ▼

☒ **Absolute Positioning**
0 0 200 20

☐ **Layout Positioning**
[Empty text field]

Parameters
Label_Text: First Name
TextField_Text: [Empty text field]

Add

Click the "Add" button to add an instance. Doing so will add an instance of Input Field. The instance will be visible in the preview section. Note that values for the components are still using the default values for the Label_Text and TextField_Text. This is intentional. The new values will appear once we close the customizer.

Take note of the yellow outline around the instance, and how First Name is highlighted at the top of the customizer. This means that the instance is selected, and the customizer is in edit mode. This allows use to make changes to the selected instance (First Name). To exit edit mode and add a new instance click on the "Cancel" button in the lower **left** of the window

 CAUTION

Clicking on the "Cancel" button in the lower right will cancel out of the customizer.

The screenshot shows the 'Template Canvas Customizer' window. On the left, the 'Instances' list contains 'First Name'. Below it, the 'Add/Edit Instance' section is active. The 'Name' field is 'First Name' with a green checkmark. The 'Template' dropdown is set to 'Form Input'. Under 'Absolute Positioning', the coordinates are 0, 0, 200, 20. The 'Parameters' section shows 'Label_Text' as 'First Name' and 'TextField_Text' as a blank field. At the bottom, 'Apply' and 'Cancel' buttons are visible. On the right, a preview of the instance shows a yellow box containing 'Label |Text Field'.

Once out of edit mode, start adding values for a new instance:

- Name: Last Name
- Template: Form Input
- Position: Absolute, (0,20) [200x20]
- Parameters: Label_Text = Last Name, TextField_Text = "" (blank)

Once entered, click the add button again.

The image shows a 'Template Canvas Customizer' dialog box. It is divided into several sections. On the left, under 'Instances', there is a list with 'First Name' and 'Last Name', where 'Last Name' is selected. Below this is an 'Add/Edit Instance' section with fields for 'Name' (set to 'Last Name'), 'Template' (set to 'Form Input'), and positioning options. 'Absolute Positioning' is selected, with coordinates (0, 20, 200, 20) shown. 'Layout Positioning' is also an option. At the bottom of this section are 'Parameters' for 'Label_Text' (set to 'Last Name') and 'TextField_Text'. 'Apply' and 'Cancel' buttons are at the bottom of the left panel. The right side of the dialog is a large canvas area showing two instances of a 'Label' and 'Text Field' widget, with the second instance highlighted in yellow. At the bottom right of the dialog are 'OK' and 'Cancel' buttons.

Instances	
First Name	
Last Name	

Add/Edit Instance

Name
Last Name ✓

Template
Form Input ▼

☒ **Absolute Positioning**
0 20 200 20

☐ **Layout Positioning**
[Empty Field]

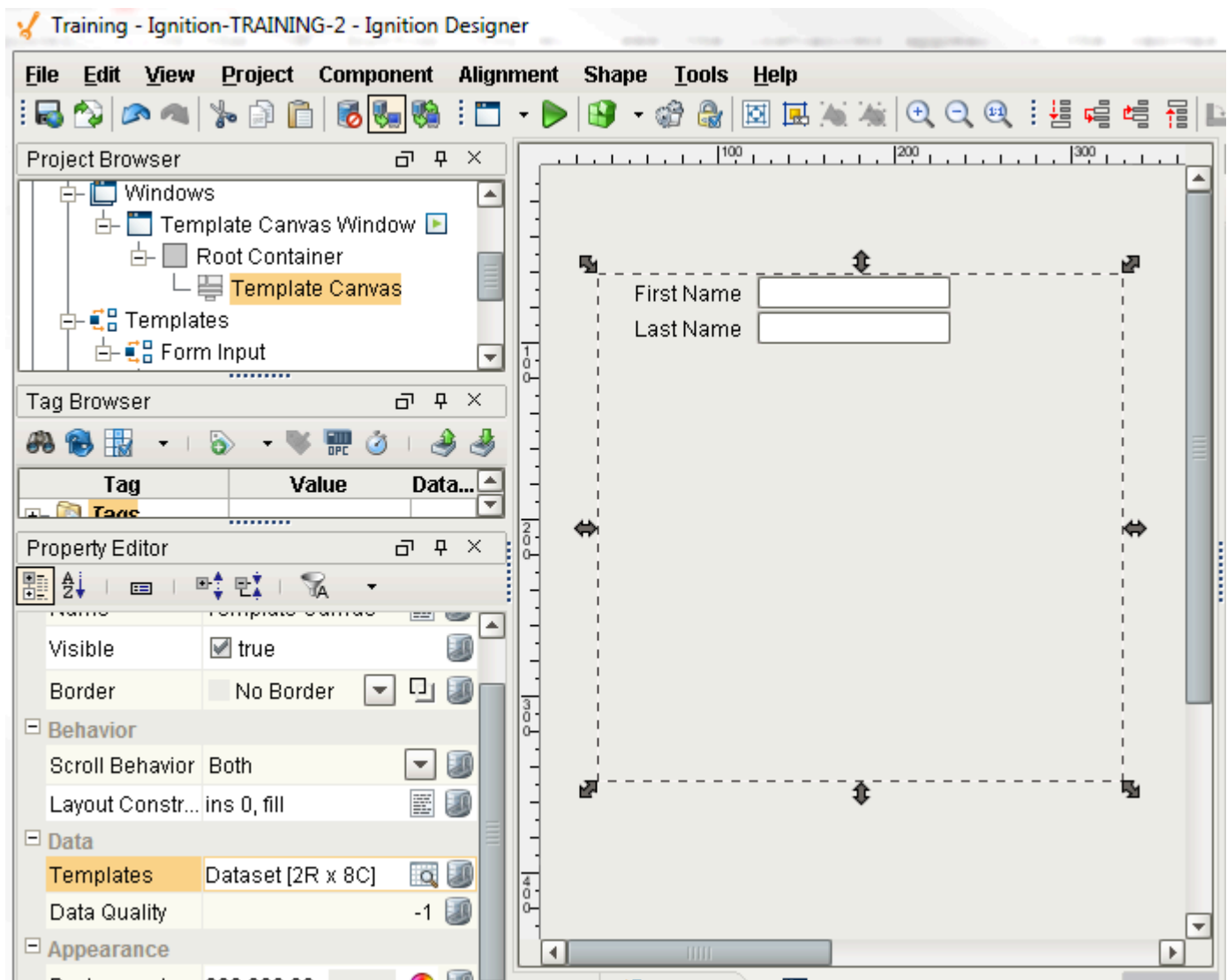
Parameters
Label_Text: Last Name
TextField_Text: [Empty Field]

Apply Cancel

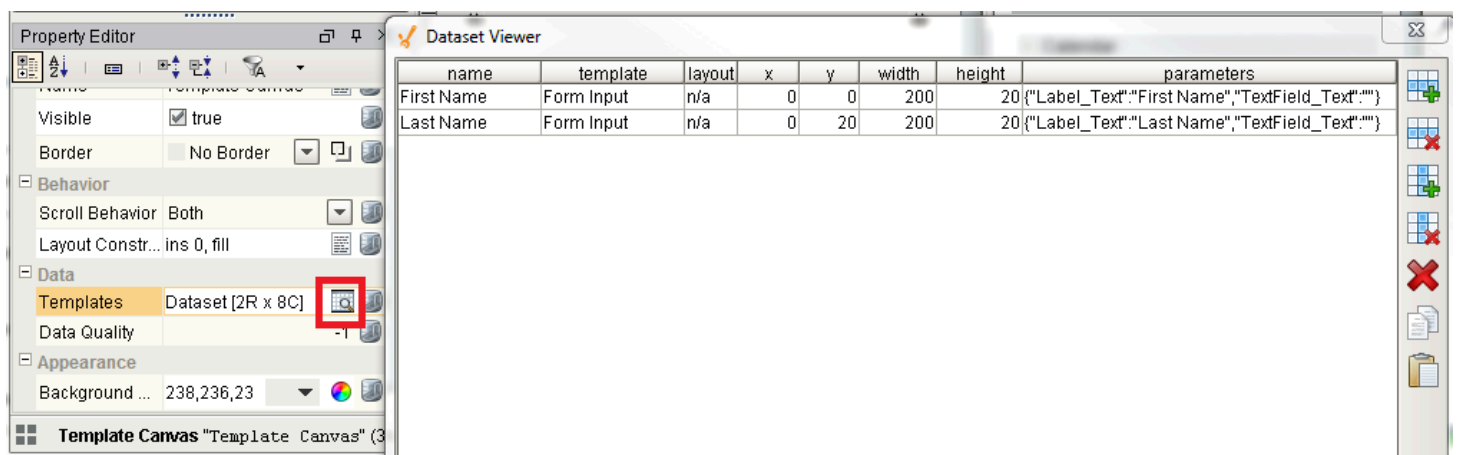
Label	Text Field
Label	Text Field

OK Cancel

Once both instances have been configured, click the "OK" button. You will see the instances appear in the canvas.



If a new instance needs to be added, it can be added through the customizer. However, the Template Canvas also has a Templates property. This property stores all of the data that was entered into the customizer, so new instances can be configured directly on the Templates property. View the dataset by clicking the Dataset Viewer button next to the Templates property.



Furthermore, template instance configurations could be stored in a database table, and the template canvas could fetch the data with a SQL Query binding on the Templates property.

Step 2: Layout Positioning (MiG Layout)

Instead of having to manually enter a size and position for each instance, we can make use of layout positioning to have the template canvas determine the best position for each instance, while also making suggestions as to where each instance is placed in relation to another. The layout positioning uses a grid-methodology to instance placement. Each instance, unless otherwise specified, is considered a single "cell" in the grid. We will tell the First Name instance to use the "wrap" parameter. This means the next cell in the grid should be placed on the next row.

Open the the Template Canvas Customizer again, and make the following modification to the First Name instance:

- Position: Layout Positioning
- Value: "wrap"

Click Apply, and the First Name instance will appear to overlap with the Last Name instance. This is because the grid only accounts for instances using the Layout Positioning.

The screenshot shows the 'Template Canvas Customizer' window. On the left, under the 'Instances' tab, a list contains 'First Name' (highlighted) and 'Last Name'. To the right of this list is a preview of the template canvas, which is a 2x2 grid. The top row contains 'Label' and 'Text Field', and the bottom row contains 'Label' and 'Text Field'. A red rectangle highlights the top row of the grid, and a yellow rectangle highlights the 'Text Field' in the top row. Below the instances list, the 'Add/Edit Instance' section is visible. It has a 'Name' field with 'First Name' and a green checkmark. The 'Template' dropdown is set to 'Form Input'. Under 'Absolute Positioning', there are four input fields, each containing '0'. The 'Layout Positioning' option is selected with a radio button. Below it, a text field contains the value 'wrap'. At the bottom, under 'Parameters', there are two fields: 'Label_Text' with the value 'First Name' and 'TextField_Text' which is empty. 'Apply' and 'Cancel' buttons are at the bottom.

Template Canvas Customizer

Instances

- First Name
- Last Name

Add/Edit Instance

Name
First Name ✓

Template
Form Input

☐ **Absolute Positioning**
0 0 0 0

☒ **Layout Positioning**
wrap

Parameters
Label_Text: First Name
TextField_Text:

Apply Cancel

The easy way to fix this is to configure Last Name with Layout Positioning as well. Make the following changes to Last Name:

- Position: Layout
- Value: "" (blank)

Template Canvas Customizer

Instances

First Name

Last Name

Add/Edit Instance

Name

Last Name

Template

Form Input

Absolute Positioning

02020020

Layout Positioning

Parameters

Label_Text

Last Name

TextField_Text

Apply

Cancel

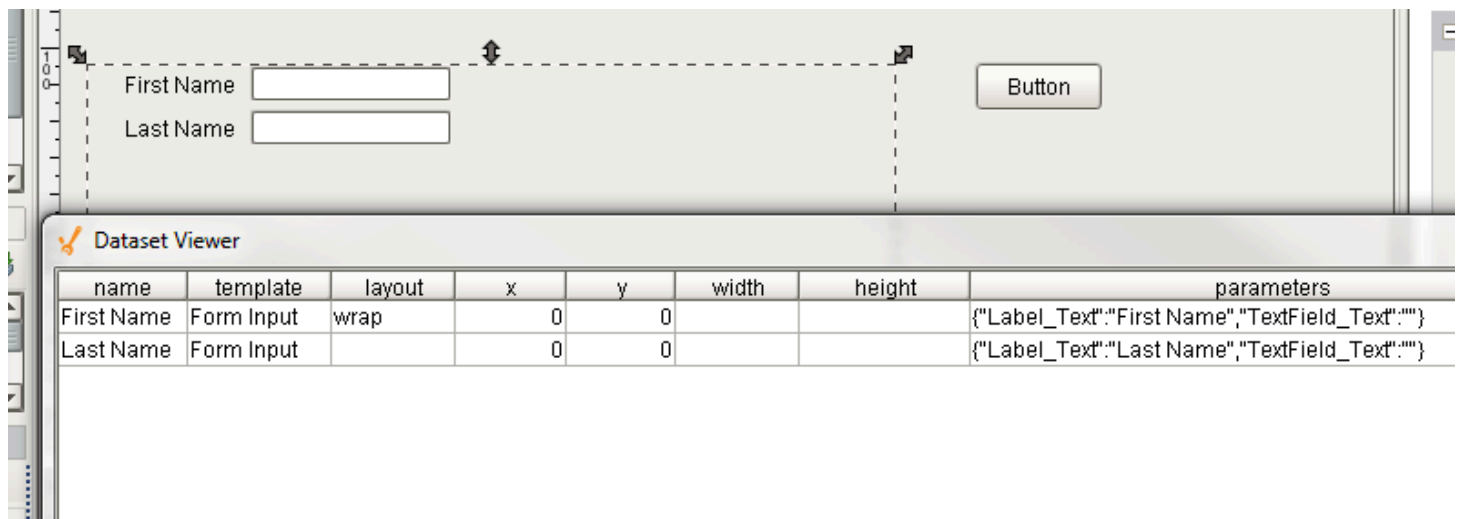
Label

Text Field

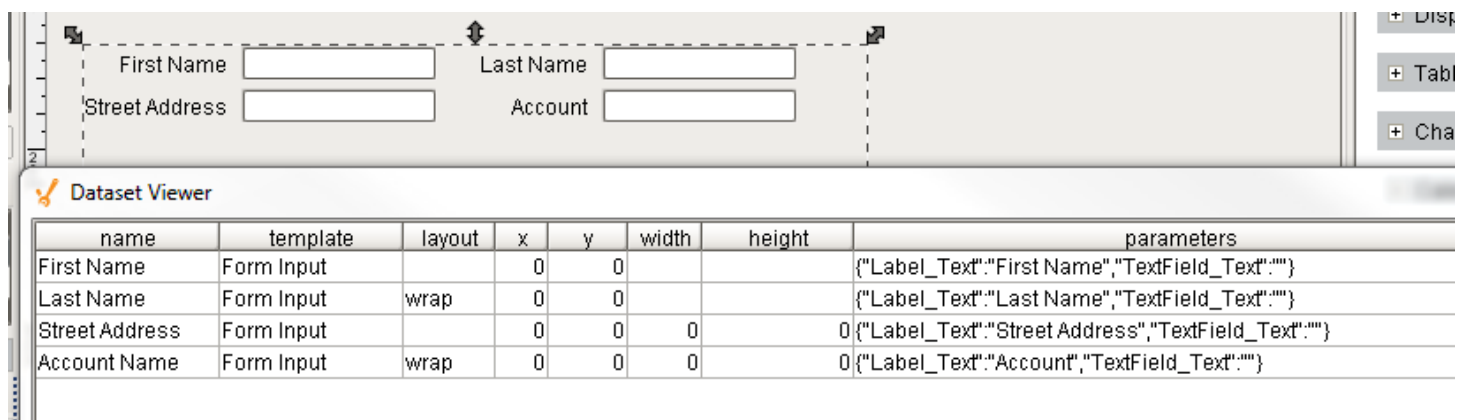
Label

Text Field

Once the changes have been applied, Click "OK". Check the Templates property again. Notice that the x, y, width, and height columns are no longer used, but the layout column for First Name now has a value.

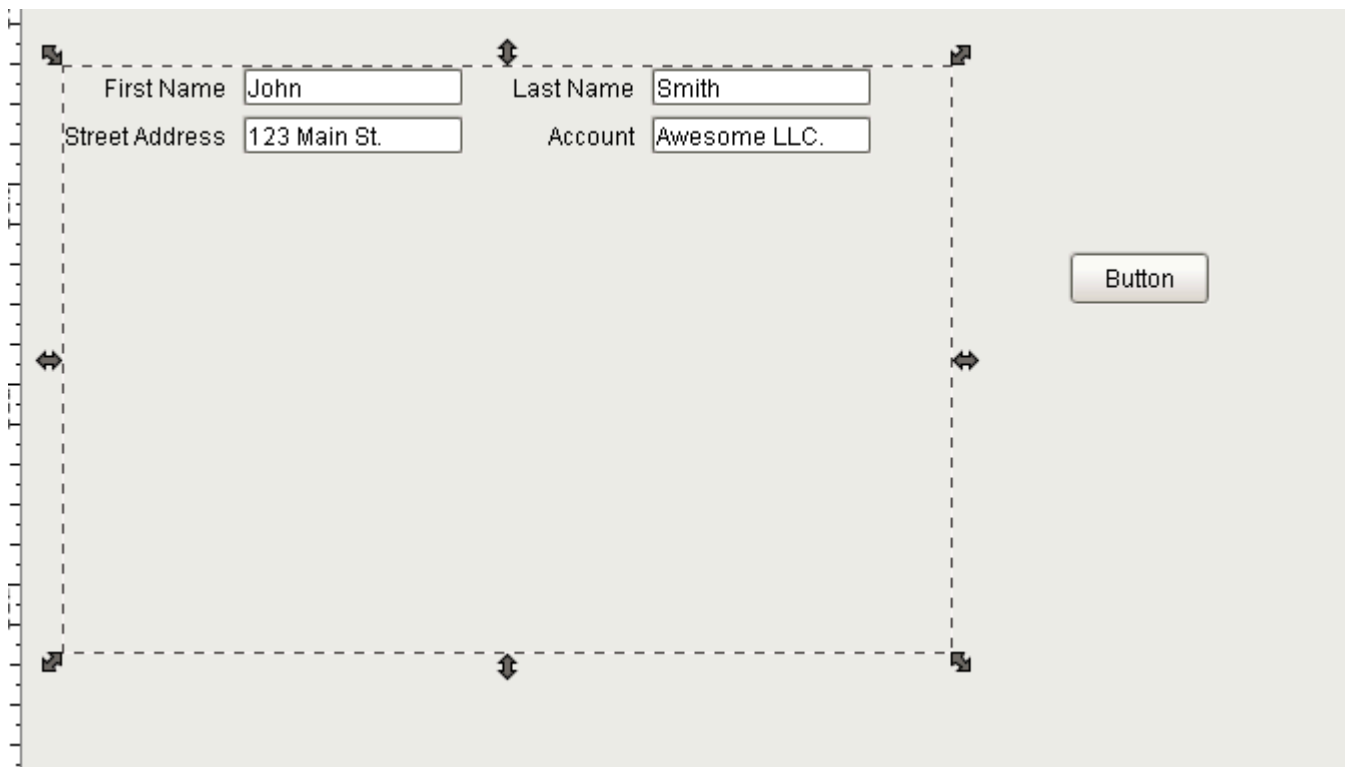


Like the previous example, new rows can be added directly to this dataset. Furthermore, the "wrap" value means the next template instance will begin on a new line. Add two new instances for Street Address and Account Name. Use either the customizer or simply add two new rows in the dataset viewer with the following values. Note that the layout value for First Name and Last Name have been changed since the previous image:

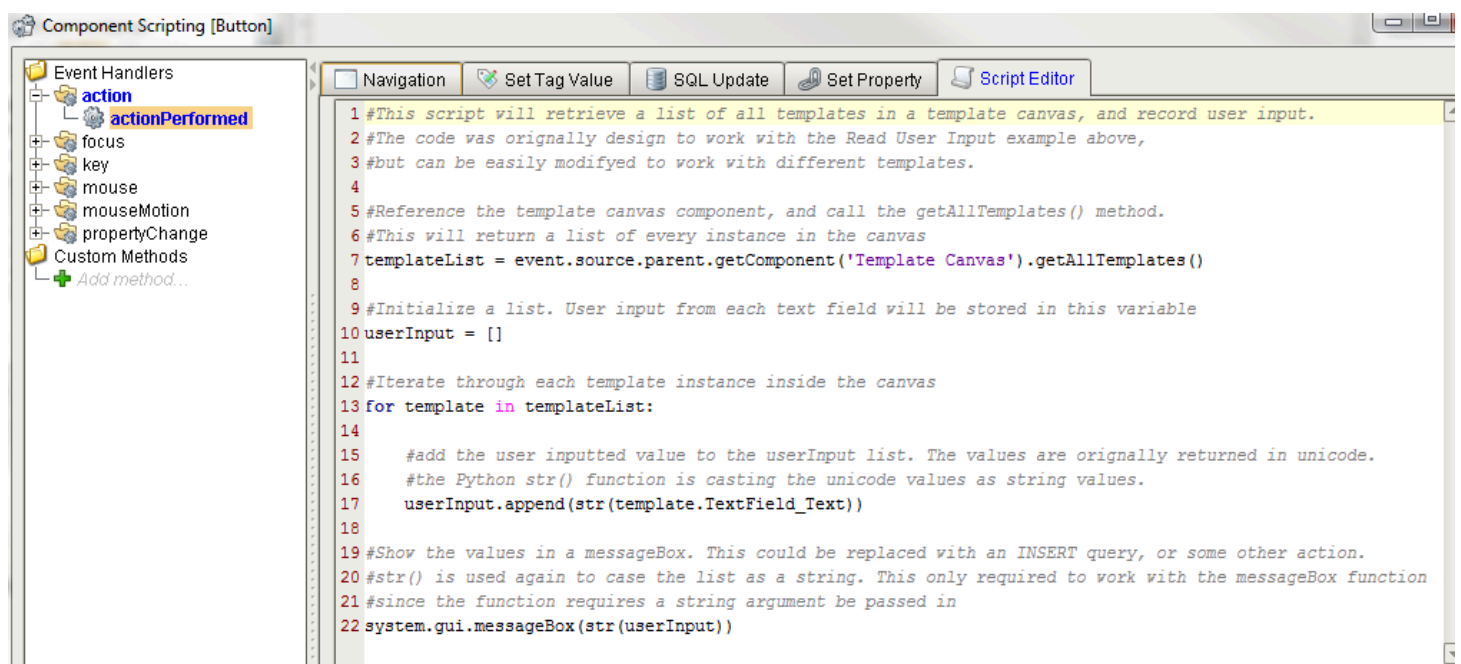


Step 3: Read User Input

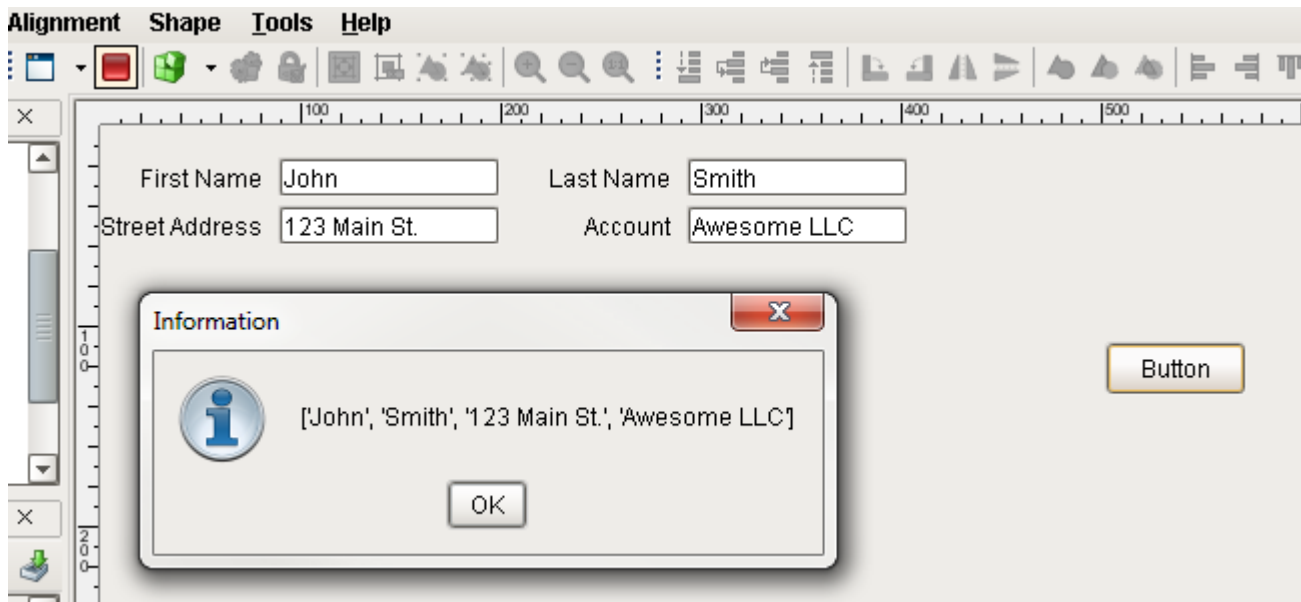
The last step is to read the user input. Put the Designer into preview mode and add some values for each text field component. Once finished, Switch the Designer back to design mode, and add a button component to the window (not the template canvas)



Add a script to the button component using the Code Snippet - **Read User Input Example** below in the Examples section. Place the code on the actionPerformed event of your button.



Click "OK" to close the Component Scripting window. Save the Project, put the Designer in preview mode, and click the button



Each value should appear in the message box. This example can easily be expanded to do something more meaningful with the input, like store to a database table.

Scripting

See the [Template Canvas Scripting Functions](#) page for the full list of scripting functions available for this component.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

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 Descriptions

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.
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 4 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 (ie: "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
```

```
#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 was originally design to work with the Read User Input example above, #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))
```

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Template Canvas Scripting Functions

This page details the various component and extension functions available for Vision's Template Canvas component.

Component Functions

.getAllTemplates()

- 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 it's definition's name. The names of each instance can be accessed with getInstanceName(). Individual components in each instance can be accessed with getComponent().
- Scope
 - Client

.getTemplate(name)

- 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
- Scope
 - Client

Extension Functions

initializeTemplate

- 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 invoking this function.
 - Vision Template **template** - The template. The name of the template in the dataset will be available as template.instanceName
- Return
 - Nothing
- Scope
 - Client

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Display

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

Label

The Label is one of the most versatile components. It can display text, images, or both.

Numeric Label

This component is a specialized label designed to display a number.

Multi-State Indicator

This component is a specialized label used to display a discrete state.

LED Display

The LED display is a stylized numeric or alphanumeric label.

Moving Analog Indicator

This component displays an analog value in context with other information about that value so that you can visua...

Image

The image component is a deceptively powerful component. While you can use other components, like the Label,...

Progress Bar

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

Cylindrical Tank

A component that looks like a 3D cylindrical tank, with some liquid inside.

Level Indicator

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

Linear Scale

Barcode

The barcode component displays some text as a barcode.

Meter

A meter display shows a value on a needle-gauge.

Compass

The compass is a component that displays up to three needles at once on a cardinal direction compass.

Thermometer

This component displays a temperature value depicted as a level in a mercury thermometer.

IP Camera Viewer

The IP camera viewing component displays a video stream from a network camera directly in one of your windows.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Label

Label

Component Palette Icon

Label

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=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 reusable 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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background Color	The background color of the label, if opaque is set to "true".	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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

Name	Description	Property Type	Scripting	Category
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	Description	Property Type	Scripting	Category
Name	The name of this component.	String	.name	Common
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

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

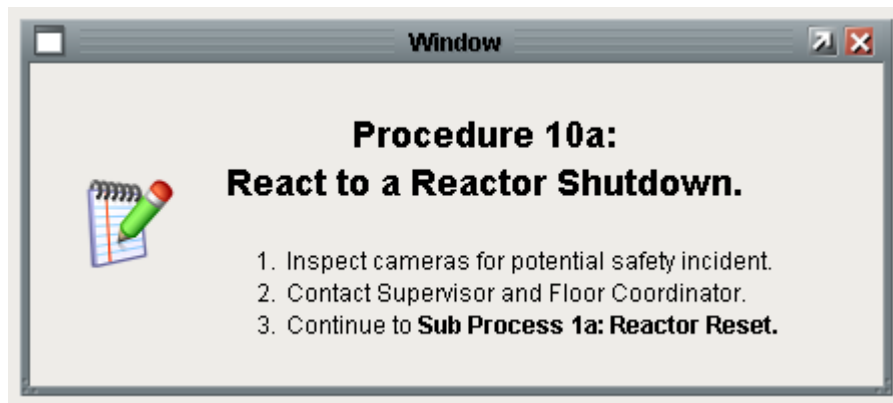
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Example

Stylized Label Inside a Popup Window



Properties

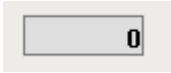
- Image Path: `Builtin/icons/48/edit.png`
- Text: Shown below:

```
<html><p><strong><center><h2>Procedure 10a: <br>
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>
```

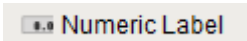
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Numeric Label



Component Palette Icon



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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Disabled Image Path	The relative path of the image to be displayed when this component is not enabled.	String	.disabledPath	Appearance

Name	Description	Property Type	Scripting	Category
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.	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

Name	Description	Property Type	Scripting	Category
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
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
Text	Full text of the label including the units. (Read only.	String	.text	Uncategorized

Name	Description	Property Type	Scripting	Category
	Usable in bindings and scripting.)			
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

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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples

Numeric Label with Red Background and Percent Sign

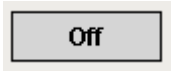


Property Name	Value
Units	%
Background Color	255,0,0

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Multi-State Indicator



Component Palette Icon



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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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

Name	Description	Property Type	Scripting	Category
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
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
State	The current state of the component.	int	.state	Data

Name	Description	Property Type	Scripting	Category
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

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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

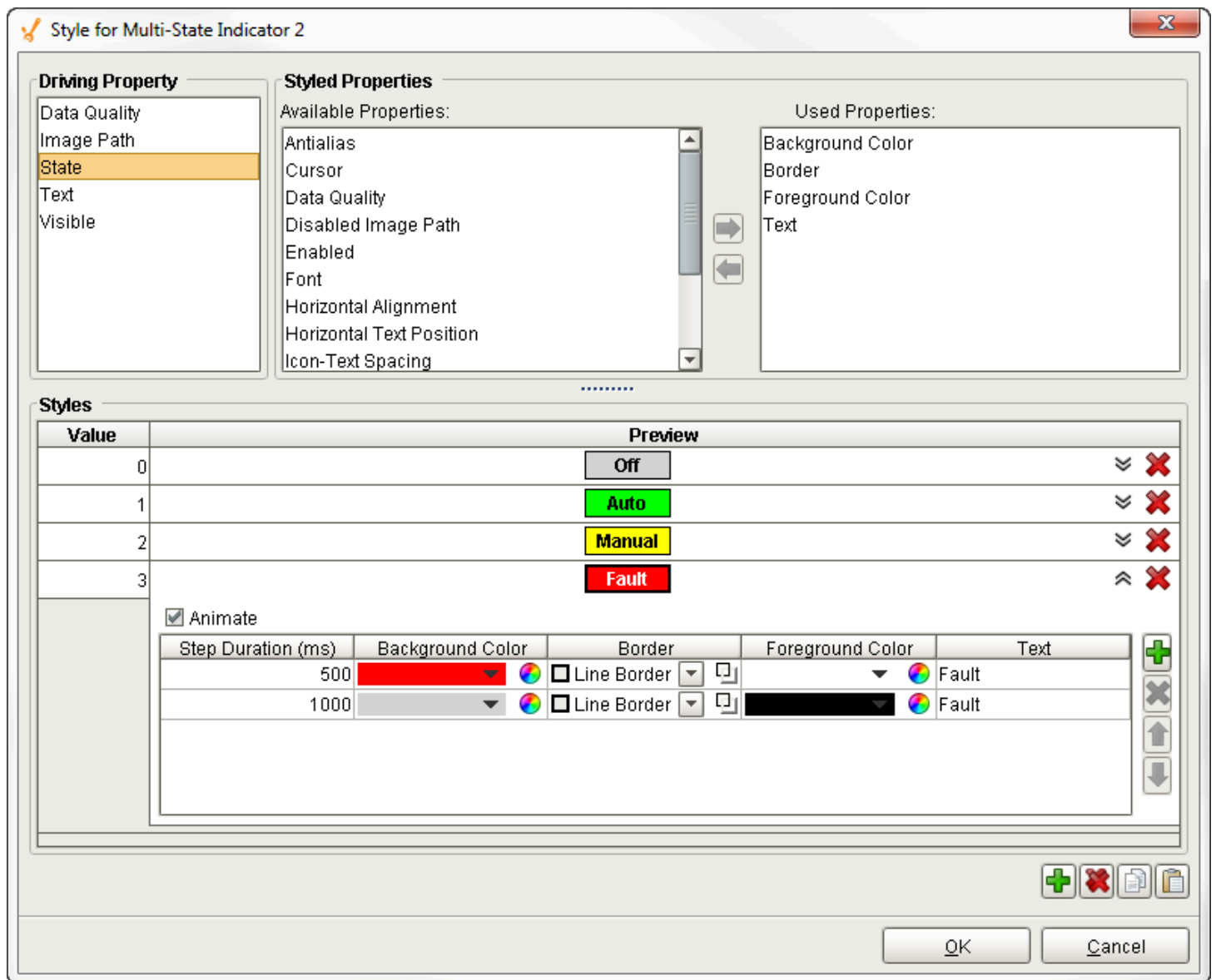
Customizers

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.

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 <i>not</i> 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.



- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples



Property Name	Value
Styles	As defined by the style customizer.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

LED Display



Component Palette Icon



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 property.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The color of the background.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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.	Color	.glyphForeground	Appearance
LED Unlit	The color of unlit LED segments.	Color	.glyphBackground	Appearance
Letter Gap	The percentage of the height to be used as an inter-character spacing.	float	.gap	Layout

Name	Description	Property Type	Scripting	Category
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
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

Name	Description	Property Type	Scripting	Category
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

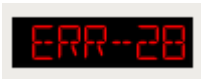
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples

Custom LED Component



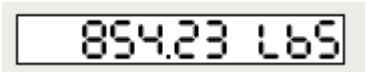
Property Name	Value
Mode	Alphanumeric
Text	ERR-28
Background Color	0,0,0
LED Lit	255,0,0
LED Unlit	0,0,0

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

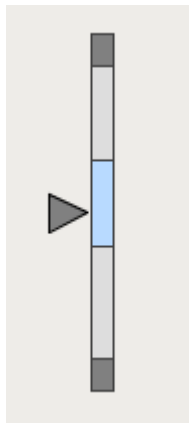


Property Name	Value
Style	5x7 Matrix
Background Color	64,64,64

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Moving Analog Indicator



Component Palette Icon

► Moving Analog Indicator

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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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.	Color	.desiredRangeColor	Appearance

Name	Description	Property Type	Scripting	Category
Hi Alarm Active	True when the process value is greater than the hi alarm threshold.	boolean	.hiAlarmActive	Data
Hi Interlock Active	True when the process value is greater than the hi interlock threshold.	boolean	.hiInterlockActive	Data
Hi-Hi Alarm Active	True when the process value is greater than the hi-hi alarm threshold.	boolean	.hihiAlarmActive	Data
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.	Color	.inactiveAlarmColor	Appearance
Interlock Color	The color of the interlock range.	Color	.interlockColor	Appearance

Name	Description	Property Type	Scripting	Category
Level 1 Alarm Color	The color of an active level 1 alarm (Hi-Hi or Lo-Lo)	Color	.level1AlarmColor	Appearance
Level 2 Alarm Color	The color of an active level 2 alarm (Hi or Lo).	Color	.level2AlarmColor	Appearance
Lo Alarm Active	True when the process value is less than the lo alarm threshold.	boolean	.loAlarmActive	Data
Lo Interlock Active	True when the process value is less than the lo interlock threshold.	boolean	.loInterlockActive	Data
Lo-Lo Alarm Active	True when the process value is less than the lo-lo alarm threshold.	boolean	.loloAlarmActive	Data
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

Name	Description	Property Type	Scripting	Category
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
Range Fill	The background color of the range strip.	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.	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.	Color	.setpointFill	Appearance

Name	Description	Property Type	Scripting	Category
Setpoint Stroke	The stroke color of the setpoint indicator.	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 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.	Color	.valueFill	Appearance
Value Indicator Stroke	The stroke color of the value indicator.	Color	.valueStroke	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

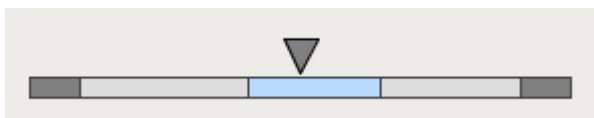
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

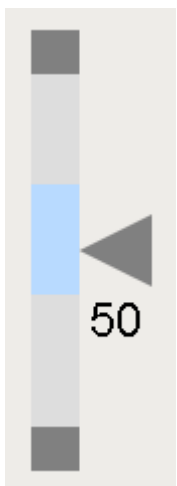
- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples

Moving Analog Indicator Expanded Horizontally



Stylized Moving Analog Indicator

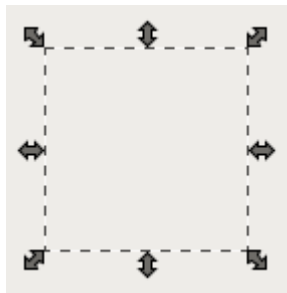


Property Name	Value
Show Value	True
Reverse Indicator	True
Stroke Width	0.0

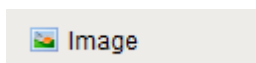
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Image



Component Palette Icon



The image component is a deceptively powerful component. While you can use other components, like the Label, to display images as well, this component gives you much more flexibility. In particular, this component has 4 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 button () 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. NOTE that the border is unaffected by rotation.	Border	.border	Common
Color Swap Filter	Swap a specific color to another.	boolean	.useColorSwap	Image Manipulation
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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

Name	Description	Property Type	Scripting	Category
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
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 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	int	.stretchWidth	Image Manipulation

Name	Description	Property Type	Scripting	Category
	If stretch mode is "% Bounds", this will be the percentage of the component's width.			
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.	Color	.swapToColor	Image Manipulation
Tint Color	If the Tint Filter is on, this is the color of the tint.	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	boolean	.useCache	Behavior

Name	Description	Property Type	Scripting	Category
	directly from the source.			
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.

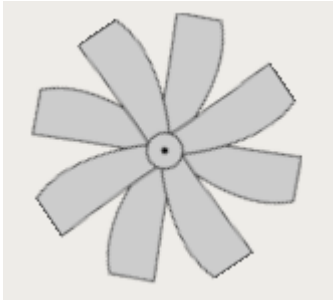
Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples

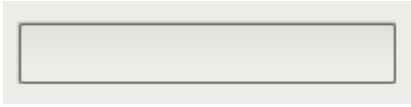


Property Name	Value
Image Path	Builtin/Air Flow/Air Flow 9.png

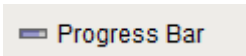
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Progress Bar



Component Palette Icon



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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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.	Color	.foreground	Appearance

Name	Description	Property Type	Scripting	Category
Horizontal?	If true, the progress bar will display horizontally, else it will display vertically.	boolean	.horizontal	Appearance
Indeterminate?	When true, the progressbar 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
Show Percentage?	If true, the progress bar will display its percentage.	boolean	.stringPainted	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance

Name	Description	Property Type	Scripting	Category
Value	The current state of the Progress Bar.	int	.value	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

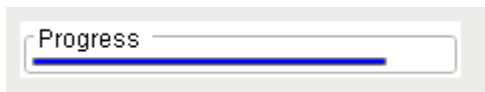
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples

Horizontal Blue Progress Bar



Property Name	Value
Border	Titled Border
Value	85
Foreground Color	0,0,255
Horizontal?	True

Wide Vertical Blue Progress Bar

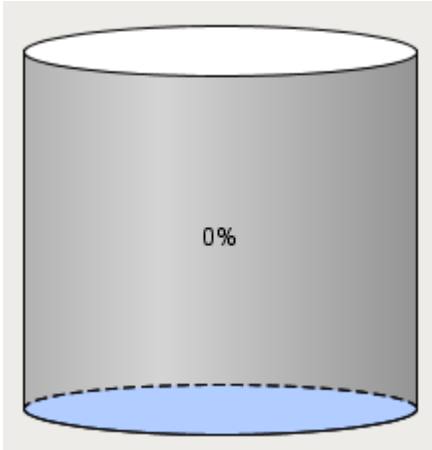


Property Name	Value
Border	Titled Border
Value	85
Foreground Color	0,0,255
Horizontal?	False

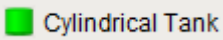
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Cylindrical Tank



Component Palette Icon



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
Anti Alias	Draw with antialias on? Makes text smoother.	boolean	.antiAlias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that 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.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Font	Font of text on this component.	Font	.font	Appearance
Font Color	The color of the value and/or percentage labels.	Color	.fontColor	Appearance
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance
Liquid Color	Color of the filled tank section.	Color	.liquidColor	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up	String	.toolTipText	Common

Name	Description	Property Type	Scripting	Category
	on mouseover of this component.			
Name	The name of this component.	String	.name	Common
Percent Format	Format string used for the percentage.	String	.percentFormat	Appearance
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.	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

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.

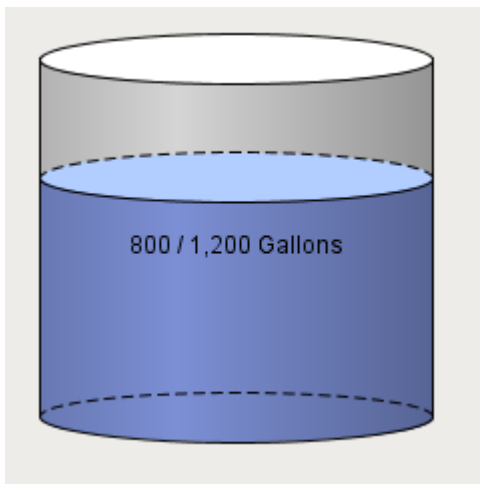
Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples

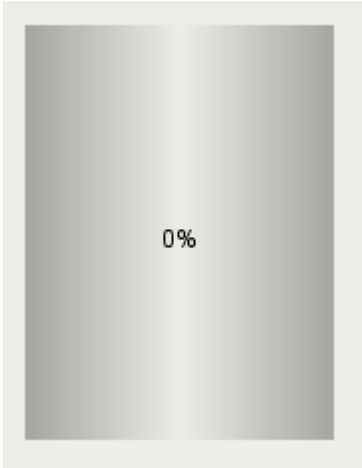


Property Name	Value
Value	800
Show Value	True
Show Percentage	False

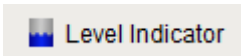
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Level Indicator



Component Palette Icon



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
Anti Alias	Draw with antialias on? Makes text smoother.	boolean	.antiAlias	Appearance
Background Color	The color of the background.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that 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.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Filled Color	Set the color of filled portion.	Color	.foreground	Appearance
Font	Font of text on this component.	Font	.font	Appearance
Font Color	The foreground color of the component.	Color	.fontColor	Appearance
Gradient	Draw level as 3D gradient?.	boolean	.gradient	Appearance
Liquid Waves	Draw liquid 'waves'?	boolean	.waves	Appearance

Name	Description	Property Type	Scripting	Category
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 way the level "grows" for an increase in value.	int	.orientation	Appearance
Percent Format	Format string used for the percentage.	String	.percentFormat	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
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

Name	Description	Property Type	Scripting	Category
Wave Length	The length of each 'wave'.	int	.waveLength	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

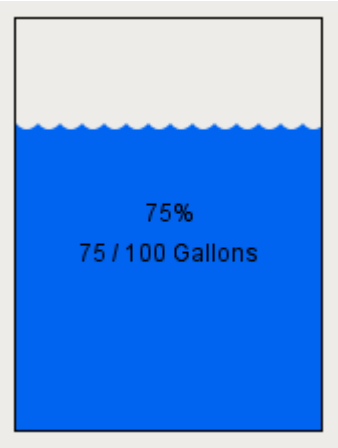
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples

Level Indicator

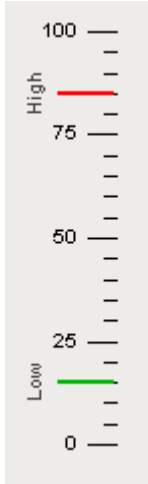


Property Name	Value
Border	Line Border
Value	75
Units	Gallons
Show Value	True
Gradient	False
Filled Color	0,100,240

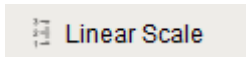
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Linear Scale



Component Palette Icon



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.

Note that 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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Fine Tick Color	The line color for fine ticks.	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	.fineTickSpan	Data

Name	Description	Property Type	Scripting	Category
Fine Tick Thickness	The line thickness for fine ticks, in pixels.	float	.fineTickStroke	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.	Color	.majorTickLabelColor	Appearance
Label Font	The font used for drawing tick labels.	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.	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	double	.majorTickSpan	Data

Name	Description	Property Type	Scripting	Category
	minor and fine tick spans.			
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.	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

Name	Description	Property Type	Scripting	Category
Mirror	Mirror the scale so it paints against the opposite edge.	boolean	.mirror	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
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

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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

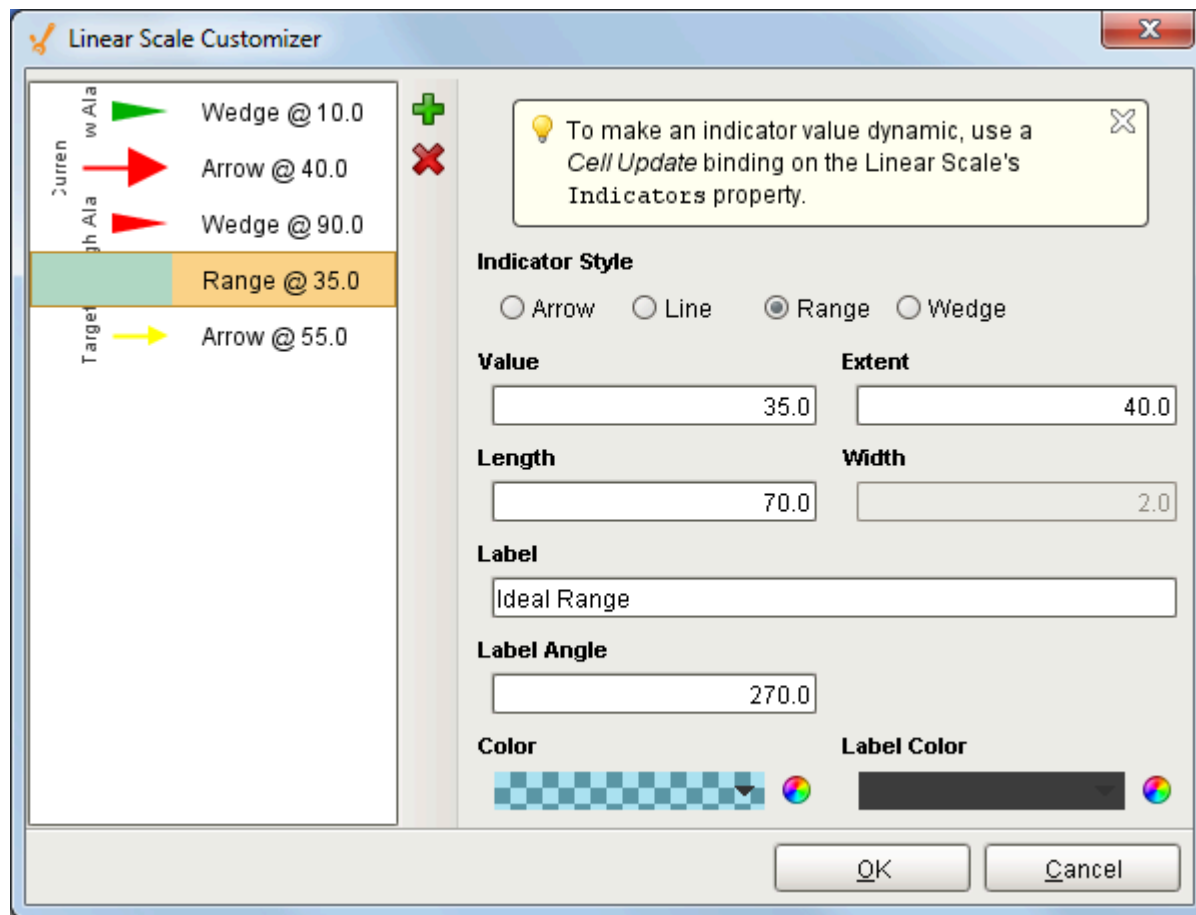
Customizers

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.

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.

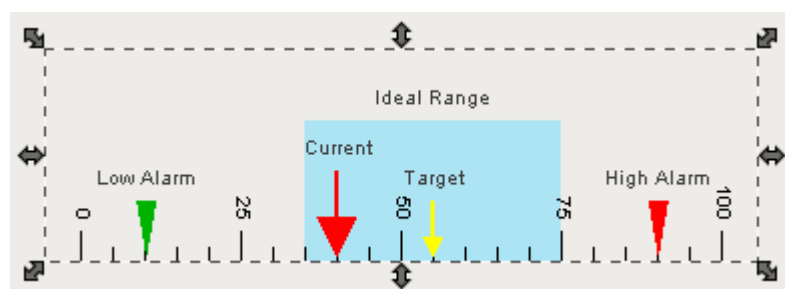


- Vision Component Customizers

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



Properties

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

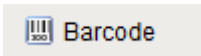
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Barcode



Component Palette Icon



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

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component.	Color	.background	Appearance
Barcode Background	The background color of the actual barcode.	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. NOTE that 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
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance

Name	Description	Property Type	Scripting	Category
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
Rotation	The angle of rotation in degrees.	int	.angleDegrees	Appearance
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

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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

This component does not have any custom properties

Examples

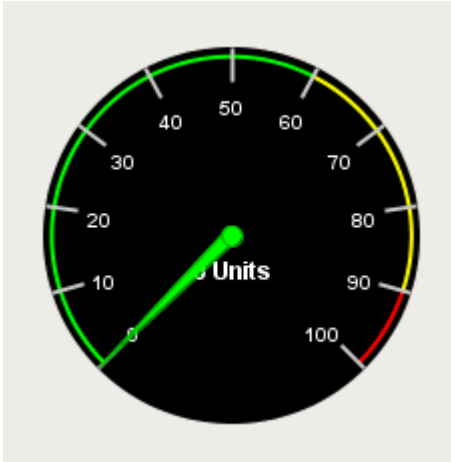


Property Name	Value
Code	123456789
Barcode Format	Extended Code 39 (narrow)
Show Text?	True

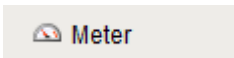
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Meter



Component Palette Icon



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 2 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. NOTE that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Dial Background	The background color of the dial face.	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.	Color	.interval1Background	Intervals
Interval 1 High	The upper bound of this interval.	double	.interval1High	Intervals

Name	Description	Property Type	Scripting	Category
Interval 1 Low	The lower bound of this interval.	double	.interval1Low	Intervals
Interval 1 Outline	The color to paint the arc of this interval.	Color	.interval1Outline	Intervals
Interval 2 Background	The color to fill the wedge of this interval.	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.	Color	.interval2Outline	Intervals
Interval 3 Background	The color to fill the wedge of this interval.	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.	Color	.interval3Outline	Intervals
Interval 4 Background	The color to fill the wedge of this interval.	Color	.interval4Background	Intervals

Name	Description	Property Type	Scripting	Category
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.	Color	.interval4Outline	Intervals
Interval 5 Background	The color to fill the wedge of this interval.	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.	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	Description	Property Type	Scripting	Category
Name	The name of this component.	String	.name	Common
Needle Color	The color of the meter's needle.	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.	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
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	String	.tickLabelFormat	Appearance

Name	Description	Property Type	Scripting	Category
	labels.			
Tick Label Color	The color of the tick labels.	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.	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

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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples

Updated Fonts



Property Name	Value
Value	35
Unit	m/s
Value Label Font	Vijaya, Bold, 15
Tick Label Font	Vijaya, Bold, 15

Chord Meter with Modified Value Intervals

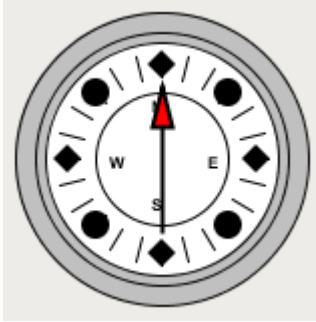


Property Name	Value
Value	35
Reverse Range?	True
Units	'None'
Arc Width	10
Meter Angle Extent	220
Meter Angle	0
Dial Shape	Chord
Interval 1 Low	40
Interval 2 High	60
Interval 3 Low	70

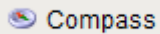
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Compass



Component Palette Icon



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 9 different styles. Use the "Disabled" style to turn off any needle.

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Center Color	The center color of the compass.	Color	.centerColor	Appearance
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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
Rose Color	The background color of the rose.	Color	.roseColor	Appearance
Rose Highlight	The highlight color of the rose.	Color	.roseHighlightColor	Appearance

Name	Description	Property Type	Scripting	Category
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.	Color	.value1Color	Appearance
Value 1 Needle	The needle type for this value.	int	.value1Needle	Data
Value 1 Outline	The outline color for value 1s needle.	Color	.value1OutlineColor	Appearance
Value 2	Value 2 for the compass.	double	.value2	Data
Value 2 Color	The main color for Value 2's needle.	Color	.value2Color	Appearance
Value 2 Needle	The needle type for this value.	int	.value2Needle	Data
Value 2 Outline	The outline color for value 2s needle.	Color	.value2OutlineColor	Appearance
Value 3	Value 3 for the compass.	double	.value3	Data
Value 3 Color	The main color for Value 3's needle.	Color	.value3Color	Appearance
Value 3 Needle	The needle type for this value.	int	.value3Needle	Data

Name	Description	Property Type	Scripting	Category
Value 3 Outline	The outline color for value 3s needle.	Color	.value3OutlineColor	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.

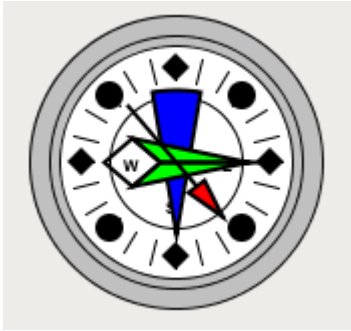
Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples

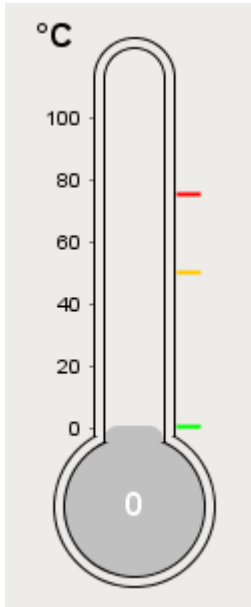


Property Name	Value
Value 1	140
Value 1 Needle	Arrow
Value 2	90
Value 2 Needle	Long
Value 3	180
Value 3 Needle	Plum

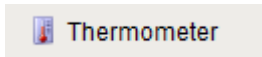
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Thermometer



Component Palette Icon



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.	Color	.axisColor	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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.	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

Name	Description	Property Type	Scripting	Category
Interval 2 Color	The color of this interval.	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.	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.	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

Name	Description	Property Type	Scripting	Category
Styles	Contains the component's styles	Dataset	.styles	Appearance
Thermometer Color	The color of the outline of the thermometer.	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.	Color	.valueColor	Appearance
Value Label Font	The font to use for the current value label.	Font	.valueFont	Appearance

Name	Description	Property Type	Scripting	Category
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.

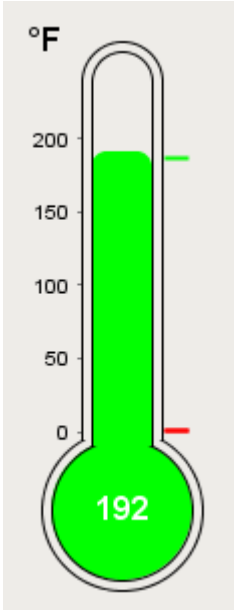
Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples

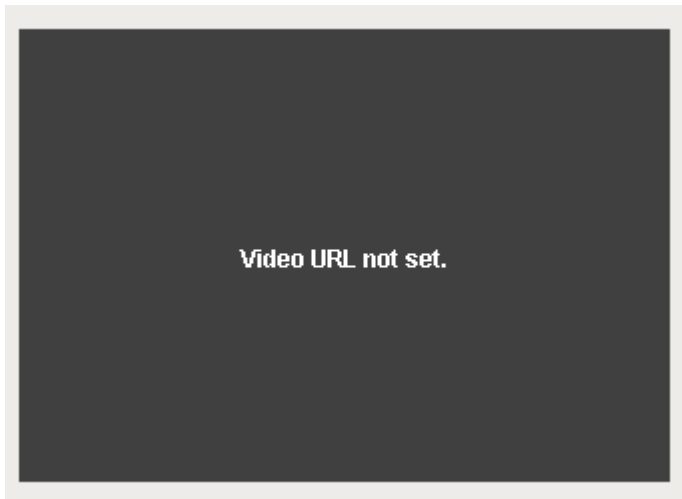


Property Name	Value
Units	Fahrenheit
Value	192
Use Range Color	True
Interval 1 Low	187
Interval 1 High	212
Interval 2 Low	0
Interval 2 High	187
Interval 3 Low	0
Interval 3 High	187

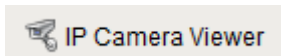
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

IP Camera Viewer



Component Palette Icon



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.

① 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.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected 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.	int	.cursorCode	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

Name	Description	Property Type	Scripting	Category
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

Name	Description	Property Type	Scripting	Category
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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples

Axis 2100 (MJPEG)

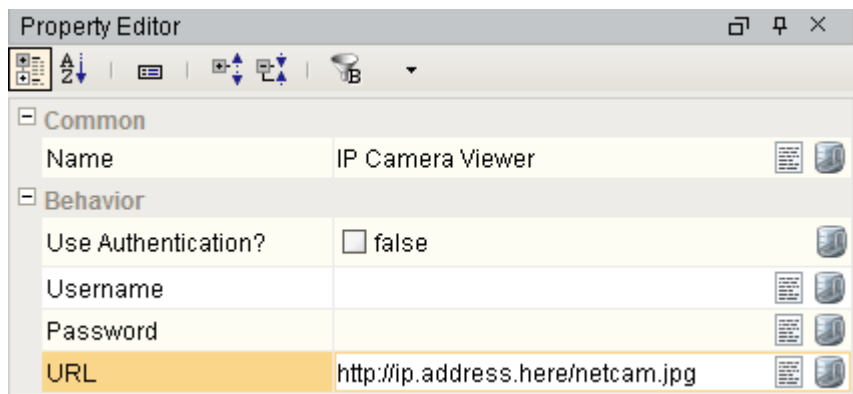
```
http://ip.address.here/axis-cgi/mjpg/video.cgi?resolution=640x480
```

Panasonic BL-C10A (MJPEG)

```
http://ip.address.here/nphMotionJpeg?Resolution=640x480&Quality=Standard
```

StarDot Netcam (JPEG stills)

```
http://ip.address.here/netcam.jpg
```



This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Input

The following components allow users to enter or select data.

Text Field

The Text Field component is used for input of any single-line text. This component will accept any alpha-numeric ...

Numeric Text Field

The Numeric Text Field is similar to the standard Text Field, except that it is specialized for use with numbers.

Spinner

The spinner component represents a value that is part of a series of values, such as numbers and dates.

Dropdown List

Formatted Text Field

This specialized text field is used for alphanumeric text input that must match some specific pattern or needs to b...

Password Field

A password field is like a text field that doesn't display the text that is being edited. You may alter the echo charac...

Text Area

Suitable for multi-line text display and editing. Will scroll vertically on demand. Will scroll horizontally if line wrap ...

Slider

The slider component lets the user drag an indicator along a scale to choose a value in a range. The slider can be...

Language Selector

The Language Selector component gives an easy way to set the user's locale to control display of dates, times, n...

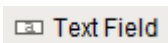
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Text Field



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 [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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background	The background color of the text box (when editable).	Color	.editableBackground	Appearance
Border	The border surrounding this component. NOTE that 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.	int	.cursorCode	Common
Data Quality	The data quality code for any tag	int	.dataQuality	Data

Name	Description	Property Type	Scripting	Category
	bindings on this component.			
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

Name	Description	Property Type	Scripting	Category
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
Protected Mode?	If true, users will need to double-click in the field in order to edit the text.	boolean	.protectedMode	Behavior
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

Name	Description	Property Type	Scripting	Category
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

Scripting

Scripting Functions

getSelectedText()

- 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.
- Scope
 - Client

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- Vision 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	Tabbed Border with Title
Font	Dialog, Bold, 12
Horizontal Alignment	Center

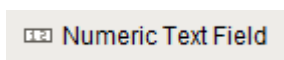
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Numeric Text Field



Component Palette Icon



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. 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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background	The background color of the text box (when editable).	Color	.editableBackground	Appearance
Border	The border surrounding this component. NOTE that 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.	int	.cursorCode	Common
Data Quality	The data quality code for any tag	int	.dataQuality	Data

Name	Description	Property Type	Scripting	Category
	bindings on this component.			
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

Name	Description	Property Type	Scripting	Category
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	String	.outOfBoundsMessage	Behavior

Name	Description	Property Type	Scripting	Category
	display if input is out-of-bounds.			
Protected Mode?	If true, users will need to double-click in the field in order to edit the value.	boolean	.protectedMode	Behavior
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-	boolean	.useBounds	Behavior

Name	Description	Property Type	Scripting	Category
	bounds", user-entered values will be silently modified to be in-bounds.			
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

Name	Description	Property Type	Scripting	Category
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

Scripting Functions

getSelectedText()

- 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.
- Scope
Client

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page.

Customizers

- [Vision 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

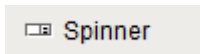
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Spinner



Component Palette Icon



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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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

Name	Description	Property Type	Scripting	Category
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
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

Name	Description	Property Type	Scripting	Category
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

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

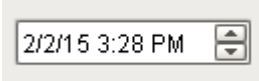
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples

Date Spinner

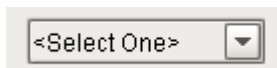


Property Name	Value
Spinner Mode	Date

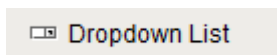
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Dropdown List



Component Palette Icon



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 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	Scripting	Category
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. Note that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data	The data which fills up the combo box. Either a 1 or 2 column DataSet, with the first column being the value, and the second being the display	Dataset	.data	Data
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data

Name	Description	Property Type	Scripting	Category
Dropdown Display Mode	Changes the dropdown's display.	int	.mode	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
Hide Table Columns?	A comma separated list of columns to hide from the dropdown table, eg. 0,2 (only used in table mode).	String	.hideTableColumns	Appearance
Horizontal Alignment	Determines the alignment of the contents along the X axis.	int	.horizontalAlignment	Layout
Max Row Count	The number of rows to display in the dropdown list before displaying a scrollbar.	int	.maximumRowCount	Appearance
Max Table Height	The maximum height allowed for the dropdown	int	.maxTableHeight	Appearance

Name	Description	Property Type	Scripting	Category
	table. (Only used in table mode).			
Max Table Width	The maximum width allowed for the dropdown table. (Only used in table mode).	int	.maxTableWidth	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
No Selection Label	The label to display when nothing is selected.	String	.noSelectionLabel	Behavior
No Selection String	The string value when nothing is selected.	String	.noSelectionString	Behavior
No Selection Value	The value when nothing is selected.	int	.noSelectionValue	Behavior
Selected Index	The index of the selected item. (Read only. Usable in bindings and scripting.)	int	.selectedIndex	Uncategorized

Name	Description	Property Type	Scripting	Category
Selected Label	The currently selected label.	String	.selectedLabel	Data
Selected String Value	The currently selected value, if the value column is a string.	String	.selectedStringValue	Data
Selected Value	The currently selected value.	Integer	.selectedValue	Data
Selection Background	The background color of a selected cell in the dropdown list.	Color	.selectionBackground	Appearance
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 even user-editable.	int	.selectionMode	Behavior
Show Table Header?	Selects whether or not the dropdown table header is displayed. (only used in table mode)	boolean	.showTableHeader	Appearance
Styles	Contains the component's	Dataset	.styles	Appearance

Name	Description	Property Type	Scripting	Category
	styles.			
Vertical Alignment	Determines the alignment of the contents along the Y axis.	int	.verticalAlignment	Layout
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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples

Setting up Datasets

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

Scenario 1: One column with a set of string values

Column
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

Column 1	Column 2
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

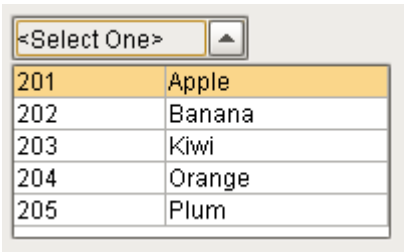
Column 1	Column 2
APL	Apples
ORN	Oranges
BAN	Bananas

- 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

Display Multiple Columns in Dropdown

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
```



Property Name	Value
Dropdown Display Mode	Table
Show Table Header	False

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Dropdown List Example



INDUCTIVE
UNIVERSITY

Dropdown

[Watch the video](#)

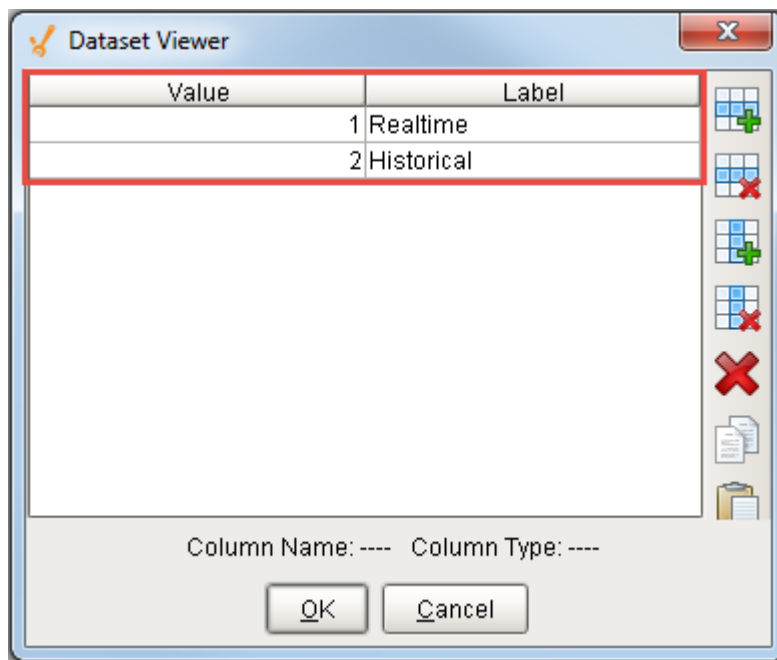
Dropdown lists are used when you want to select a single item from a list of options. The Dropdown component is under the Input section of the component palette. Simply drag it on to your window. The most important property of a Dropdown component is the Data property. It is a Dataset that contains one or more rows of data. Each of the rows are different options that you see on the component. Select the Dropdown component and click on the Dataset Viewer icon for the **Data** property to manually add some options.

Data		
Data	Dataset [2R x 3C]	
Selected Value		1 
Selected String Value	Realtime	
Selected Label	Realtime	
Data Quality		192 

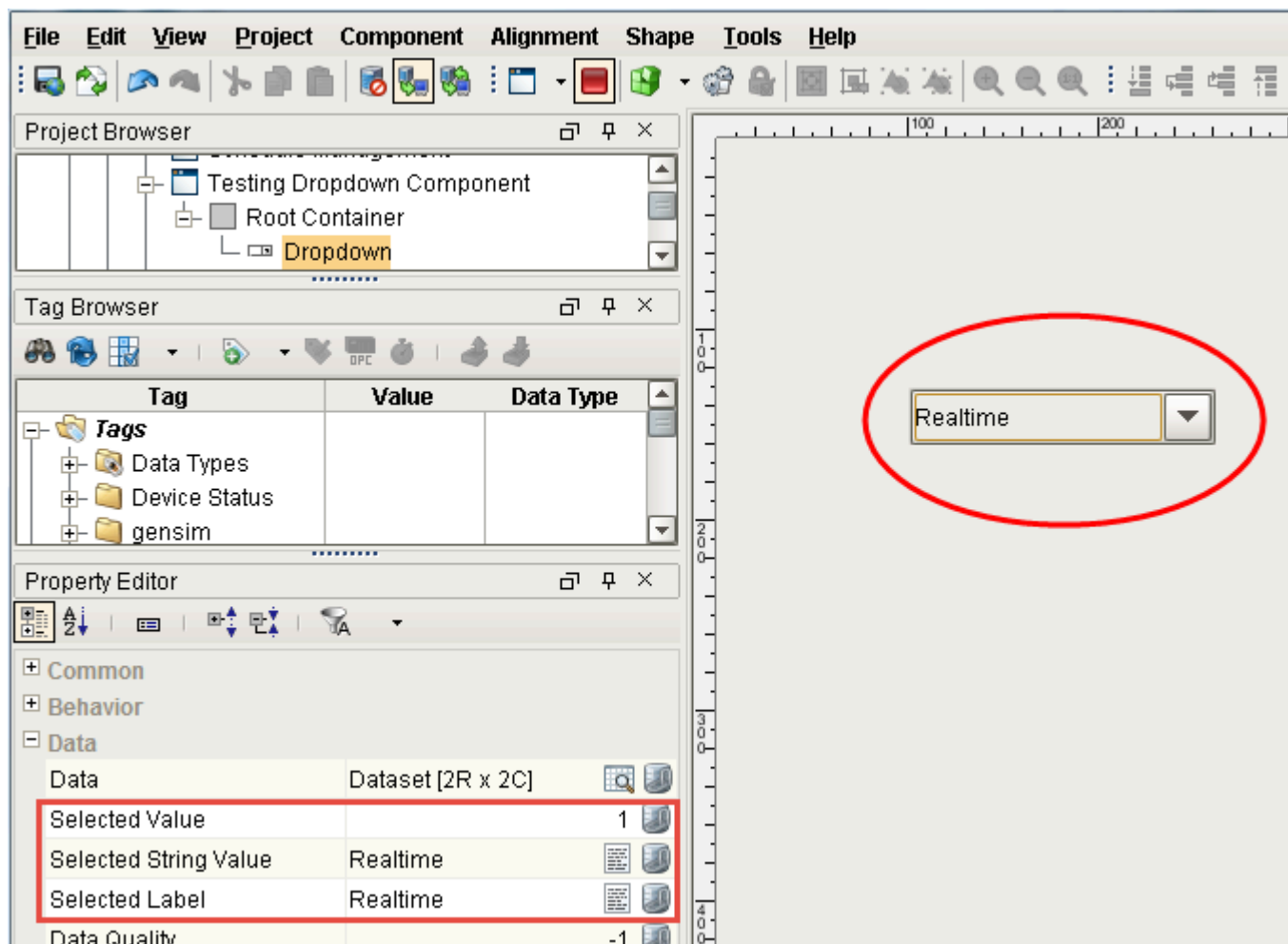
There are 3 ways you can use the **Data** property's dataset: a number/label pair, a single label column, and a code/label pair. Which mode is used depends on what columns are in the **Data** property's dataset, and will determine the values of the **Selected Value**, **Selected String Value** and **Selected Label** properties. Any additional columns that are added to the dataset will not affect these properties.

In the first mode, the first column is an integer (often an id) and the second column is a string (often a label). The first column is the **Value** column which is invisible to the user and is usually used in binding. The second column is the **Label** column and is visible to the user in the dropdown list.

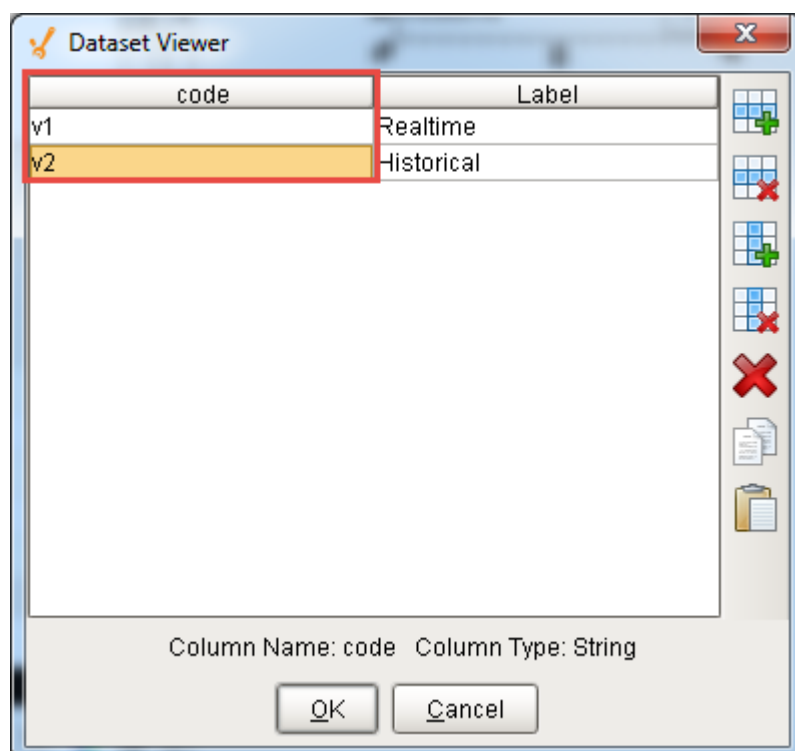
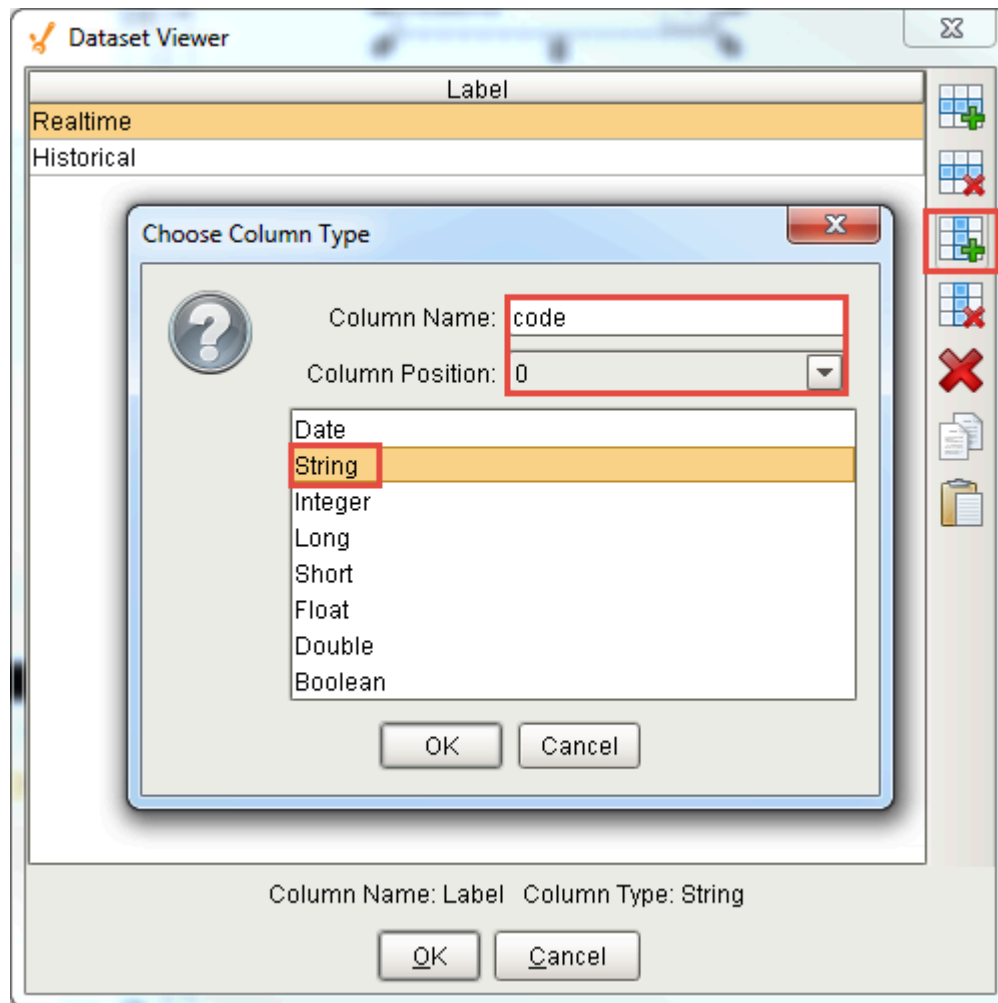
In this example, the dataset has 2 rows. Under the **Value** column, you see the integers '1' and '2' respectively. Under the **Label** column, you see the '**Realtime**' and '**Historical**' options.



In **Preview Mode**, select from the list of dropdown options. Notice, you can only see the options in the **Label** column. Select the '**Realtime**' option. You can see under the Data property section, the **Selected Value** is '1' which is the first (integer) column in the dataset and is not visible to users. The **Selected String Value** and the **Selected Label** are the same and show the '**Realtime**' option which is the second (string) column in the dataset.

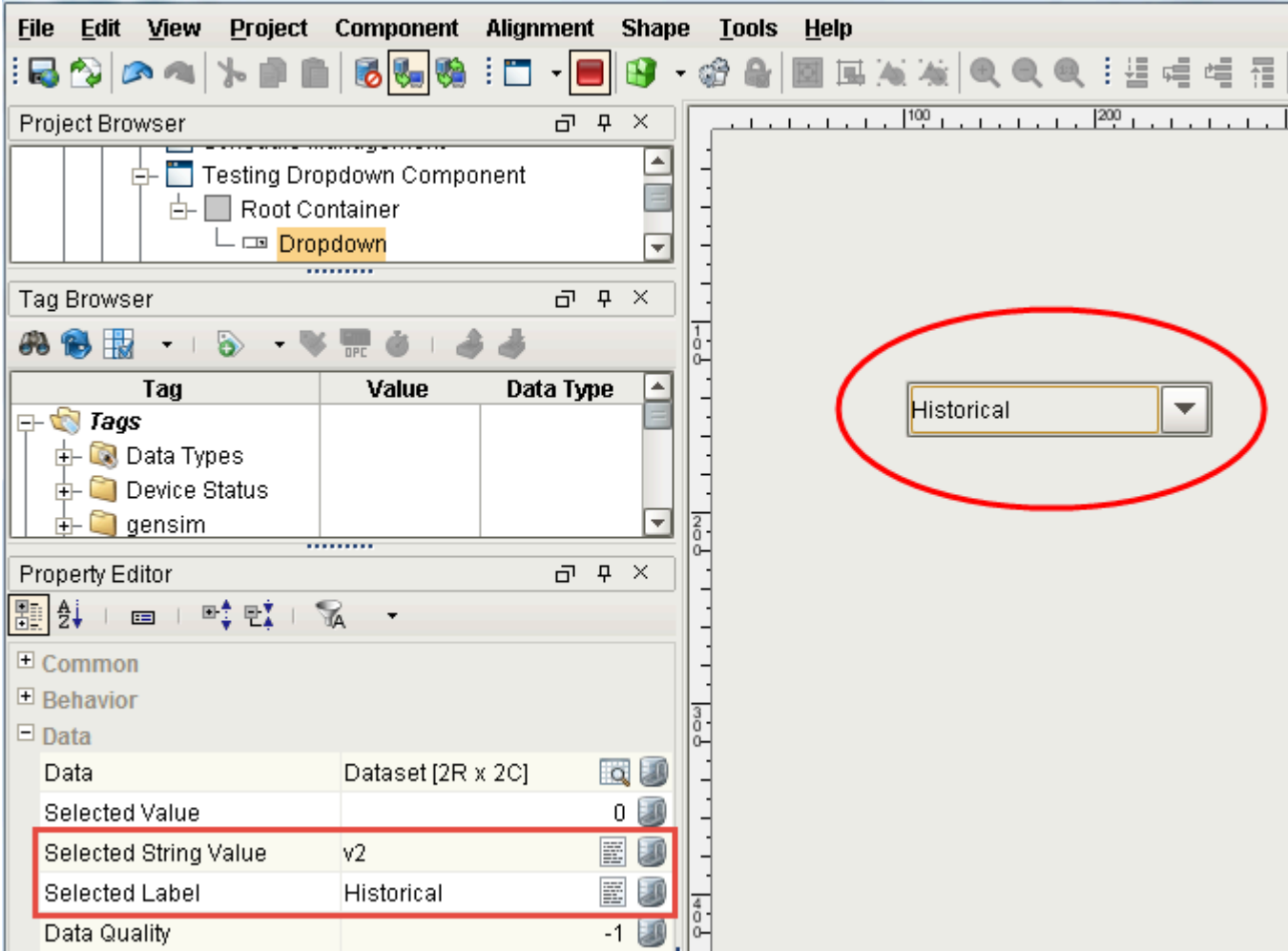


The second mode you can use is a string/string combination which are 2 columns that are both strings. Remove the **Value** column from our example by selecting a cell in that column and clicking on the vertical x icon. Add another column by clicking on the vertical plus icon. Call the new column 'code,' set the column position to '0' and make it a 'String.' Both columns are now strings. Under the 'code' column enter 'v1' and 'v2.' Press **OK**.

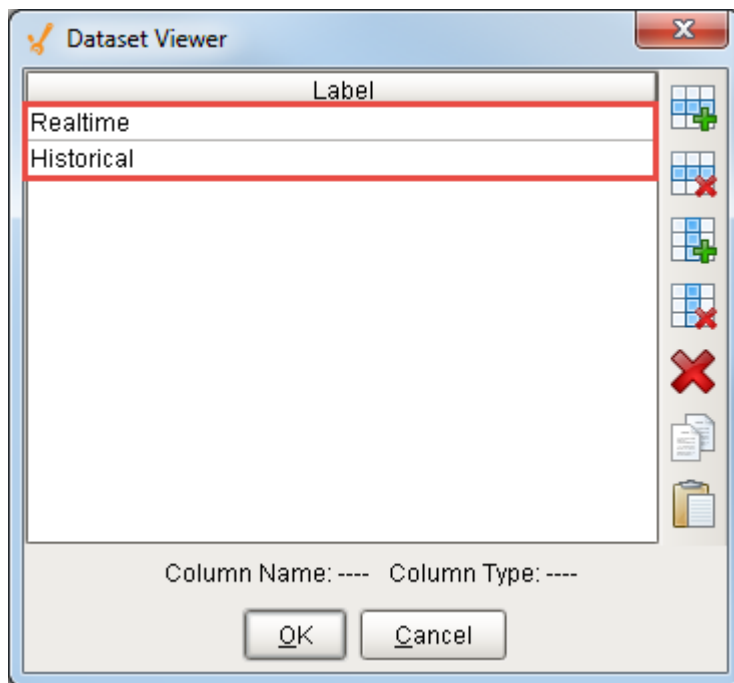


Because we added the '**code**' column in position '**0**,' the users will see the same 2 options in the dropdown list: '**Realtime**' and '**Historical**.' If you see '**v1**' and '**v2**' instead, that means your '**code**' column is not the first column.

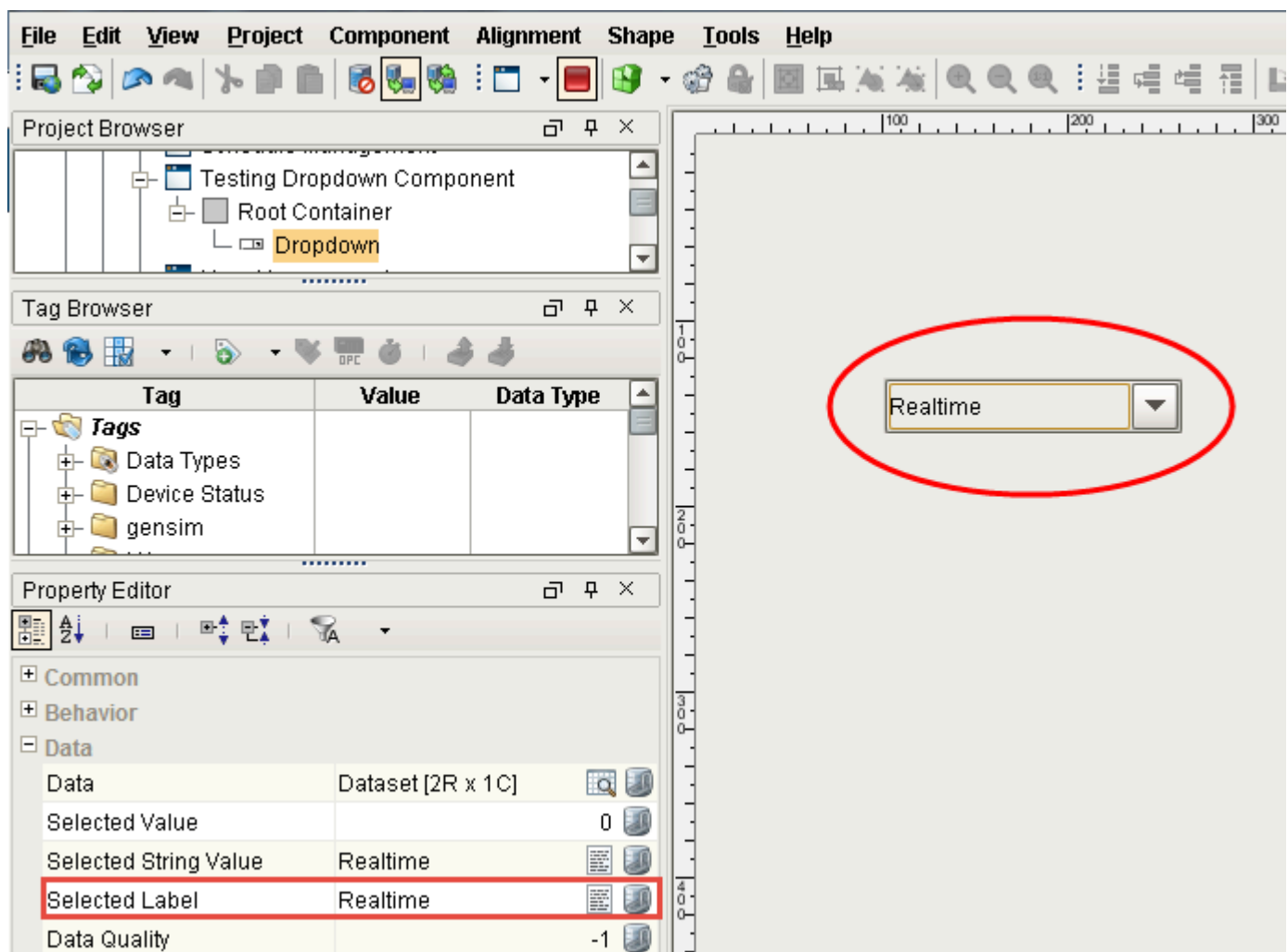
In **Preview Mode**, click on '**Historical**' and you can see the **Selected String Value** is '**v2**' because the value is in the first column and the **Selected Label** is '**Historical**.'



The third mode you can use is simply one string column. Remove the '**code**' column and keep the '**Label**' column the same. Again, you are going to see the same 2 options in the dataset viewer: '**Realtime**' and '**Historical**.' Note, that this only applies if there is exactly one column in this dataset.

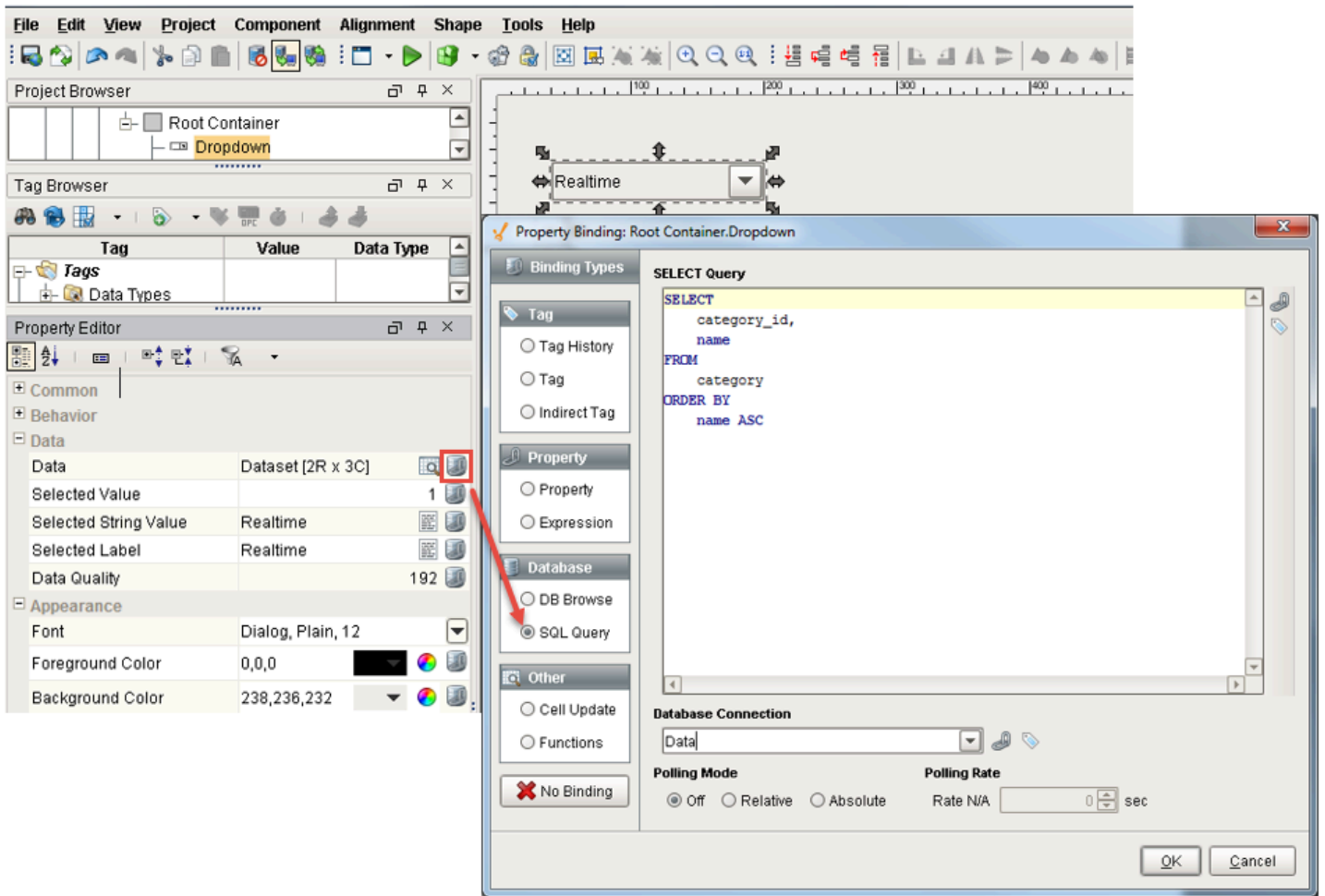


In **Preview Mode**, select the '**Realtime**' option. You will see the same value in the **Selected Label** and **Selected String Value** properties since there is only the one column.

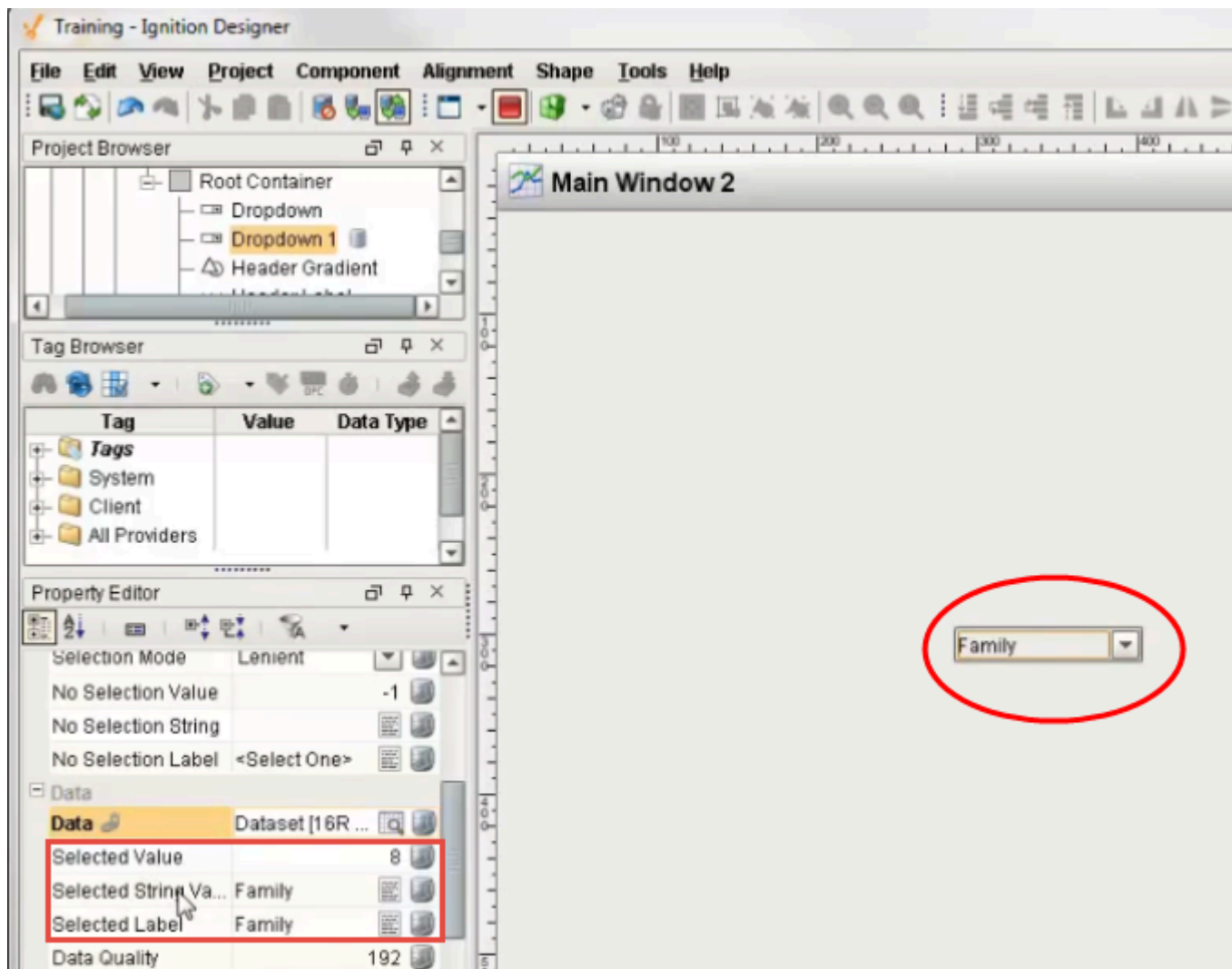


Now, you can set these dropdown options manually or bind the Data property. In this example, you can take a Dropdown List on the window and bind the Data property using a SQL query.

Click on the binding icon for the Data property. Select the SQL Query Binding Type and type a query that brings back an ID and Name from one of your tables in the database.

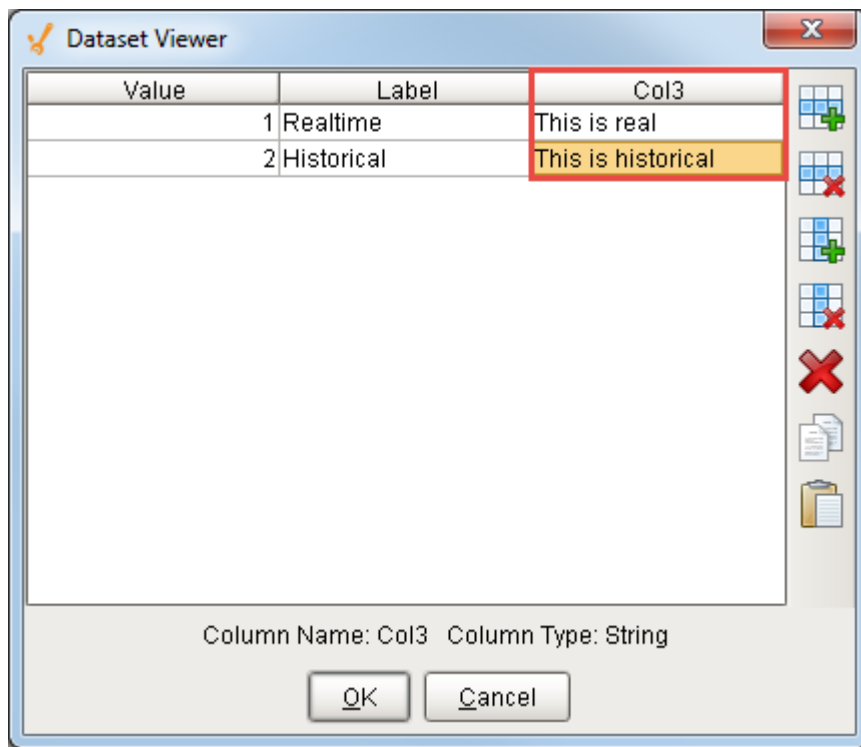


Click **OK** to see all the options that came back from the database. In **Preview Mode**, select any one of these options and you can see the **Selected Value**, **Selected String Value**, and **Selected Label** as shown in the following example.

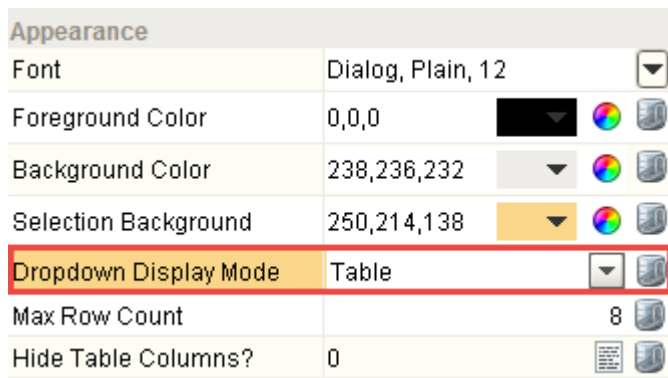


Another feature of the Dropdown component is you can show more than one column to a user. Drag another Dropdown component to your window. Go to the Dataset viewer and add some options manually. Instead of having only one or two columns you can add as many as you want. The first column needs to be either an integer or a string. Any additional columns will show up in the dropdown.

Open the Dataset View, under **Value**, enter '1' and '2.' Under **Label**, enter '**Realtime**' and '**Historical**.' Add another column and call it '**Col3**' and make it a string and enter '**This is real**' and '**This is historical**.'



Press **OK**. Go to the **Property Editor > Appearance** and set the **Dropdown Display Mode** to '**Table**' to see both columns.



Now you can see **Label** and **Col3** and select between those different rows. It's a nice way to show more information in the Dropdown list. The **Selected Value**, **Selected String Value**, and **Selected Label** properties will behave the same as previous examples ignoring any columns beyond the second.

File Edit View Project Component Alignment Shape Tools Help

Project Browser

- Testing Dropdown Component
 - Root Container
 - Dropdown
 - Dropdown 1

Tag Browser

Tag	Value	Data Type
Tags		
Data Types		
Device Status		
gensim		

Property Editor

Common

Behavior

Data

Data	Dataset [2R x 3C]	
Selected Value		1
Selected String Value	Reatime	
Selected Label	Reatime	
Data Quality		-1

Appearance

Font	Dialog, Plain, 12	
Foreground Color	0,0,0	
Background Color	238,236,232	
Selection Background	250,214,138	
Dropdown Display Mode	Table	
Max Row Count		8
Hide Table Columns?	0	

Reatime

Label	Col3
Reatime	This is real
Historical	This is historical

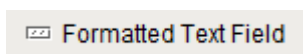
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Formatted Text Field



Component Palette Icon



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
Antialias	Draw with antialias on? Makes text smoother	boolean	.antialias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that 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.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data

Name	Description	Property Type	Scripting	Category
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.	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
Overwrites Text	Overwrites text while typing	boolean	.overwriteMode	Behavior
Reg Ex Pattern	Regular Expression Validation Pattern	String	.validationPattern	Behavior

Name	Description	Property Type	Scripting	Category
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	Selects regular expression or mask-driven field validation.	int	.validationMode	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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

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

Phone Number Format

(800) 266-7798

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

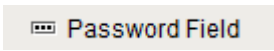
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Password Field



Component Palette Icon



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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. Note that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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

Name	Description	Property Type	Scripting	Category
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
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

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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Example

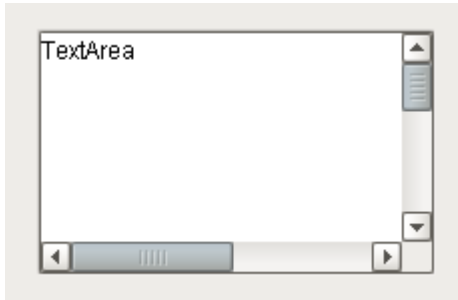
Password Field with Question Marks as the Echo Character

Property Name	Value
Echo Character	?

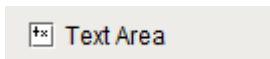
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Text Area



Component Palette Icon



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

New in 7.9.5 Some of the following properties are new.

Name	Description	Property Type	Scripting	Category
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that 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.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Defer Updates	When true, the 'text' property will not fire updates	boolean	.deferUpdates	Behavior

Name	Description	Property Type	Scripting	Category
	while typing. It will wait for the component to lose focus.			
Editable	Controls whether or not the user can edit the text within this text area.	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
Reject Updates During Edit	If true, this field will not accept	boolean	.rejectUpdatesDuringEdit	Behavior

Name	Description	Property Type	Scripting	Category
	updates from external sources (like DB bindings) while the user is editing the field.			
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

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

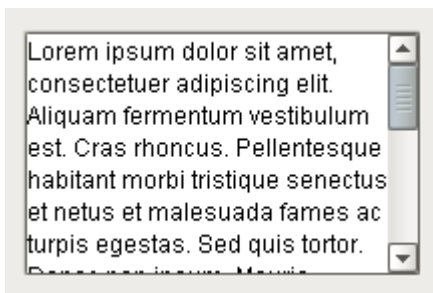
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples

Word Wrap and No Scroll Bars until Needed

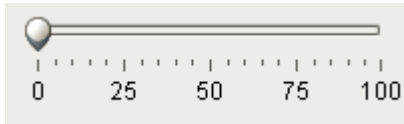


Property Name	Value
Line Wrap	True
Text	514 Characters
Rows	0
Columns	0

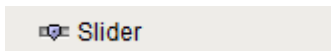
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Slider



Component Palette Icon



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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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.	Color	.foreground	Appearance

Name	Description	Property Type	Scripting	Category
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

Name	Description	Property Type	Scripting	Category
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
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

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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Examples

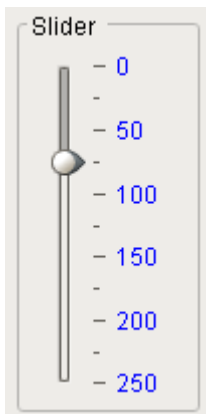
Code Snippet

#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 Type	Value
Maximum Value	250
Minor Tick Spacing	25
Foreground Color	0,0,255
Major Tick Spacing	50

Horizontal Slider without Tickmarks

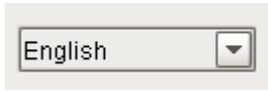


Property Type	Value
Paint Ticks?	False
Minor Tick Spacing	0
Major Tick Spacing	100

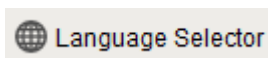
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Language Selector



Component Palette Icon



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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that 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.	String	.selectedLocale	Uncategorized

Name	Description	Property Type	Scripting	Category
	Usable in bindings and scripting.)			
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.

Event Handlers

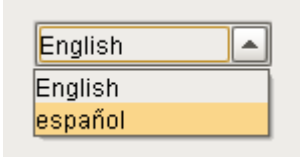
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

This component does not have any customizers.

Example

Select Between Languages



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

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Misc

The following components give you various ways to create or animate displays.

Paintable Canvas

The Paintable Canvas component is a component that can be custom painted using Jython scripting.

Line

The Line component displays a straight line. It can run north-south, east-west, or diagonally.

Pipe Segment

The Pipe Segment component displays a quasi-3D pipe.

Pipe Joint

The Pipe Joint displays a fancy joint component two join two pipe segments together.

Sound Player

The Sound Player component is an invisible component that facilitates audio playback in the client.

Timer

The Timer component is an invisible button that can be used to create repeated events in a window.

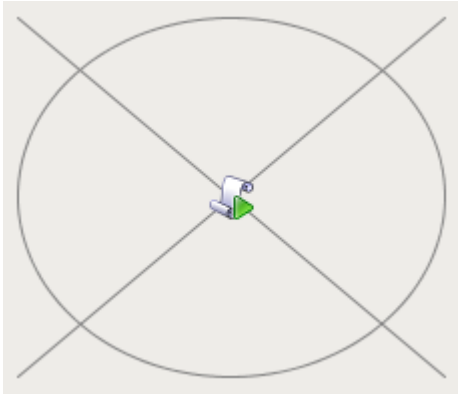
Signal Generator

The Signal Generator is similar to the Timer component, but its value isn't simply a counter.

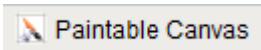
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Paintable Canvas



Component Palette Icon:



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 Canvas. 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.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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.	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

Name	Description	Property Type	Scripting	Category
Styles	Contains the component's styles.	Dataset	.styles	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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

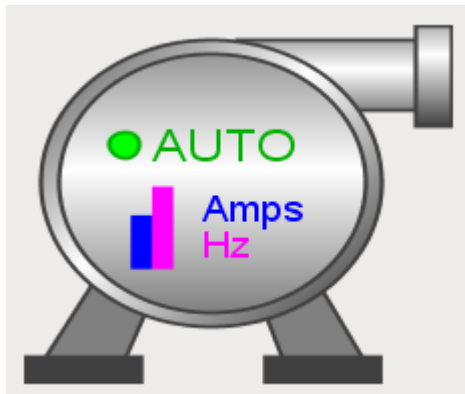
Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

Example

There are no examples associated with this component. However examples are available in the component itself.

The component comes prescribed to render the following pump:



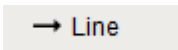
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Line



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 [Reporting Line](#).

Properties

Name	Description	Property Type	Scripting	Category
Anti Alias	Draw with antialias on? Makes text smoother.	boolean	.antiAlias	Appearance
Color	Set the color of the line.	Color	.foreground	Appearance
Cursor	The mouse cursor to use when hovering over this component.	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
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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.	int	.lineStyle	Appearance

Name	Description	Property Type	Scripting	Category
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
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

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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

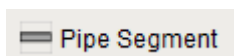
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Pipe Segment



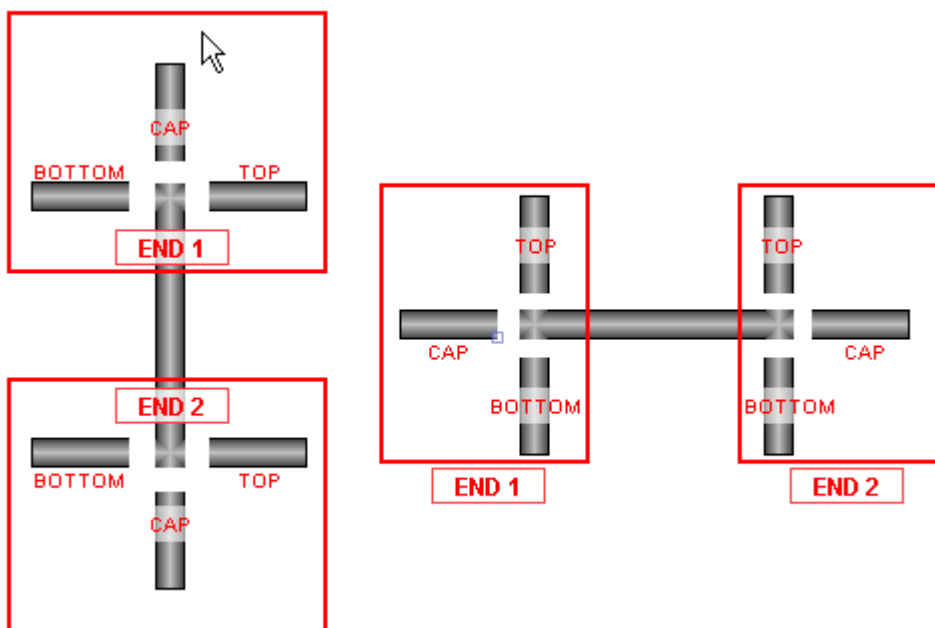
Component Palette Icon:



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 can be a bit confusing to a new user. It is done via 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 should make the naming conventions more clear:



Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Center Fill	The center of the fill gradient.	Color	.mainColor	Appearance
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Edge Fill	The edge of the fill gradient.	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

Name	Description	Property Type	Scripting	Category
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.	Color	.outlineColor	Appearance
Styles	Contains the component's styles.	Dataset	.styles	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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Pipe Joint



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 2,3, or 4-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. NOTE that the border is unaffected by rotation.	Border	.border	Common
Bottom?	Joint has an outlet at the bottom?	boolean	.bottom	Appearance
Center Fill	The center of the fill gradient.	Color	.mainColor	Appearance
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Edge Fill	The edge of the fill gradient.	Color	.secondaryColor	Appearance
Left?	Joint 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.	String	.name	Common
Outline Color	The color of the outline border.	Color	.outlineColor	Appearance

Name	Description	Property Type	Scripting	Category
Right?	Joint has an outlet at the right?	boolean	.right	Appearance
Styles	Contains the component's styles	Dataset	.styles	Appearance
Top?	Joint has an outlet at the top?	boolean	.top	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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)
- [Style Customizer](#)

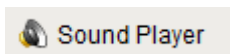
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Sound Player



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. Note that the sound clip needs to be a *.wav file, and that the clip becomes embedded within the window that the sound player is on - clients do not need access to a share *.wav file.

NOTE

Due to how the Mobile Module functions, the sound player component does not work with projects launched using the Mobile Module.

Properties

Name	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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
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

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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

- [Vision Component Customizers](#)

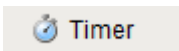
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Timer



Component Palette Icon:



Description

The Timer component 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 is 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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

This component does not have any custom properties.

Example

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"
```

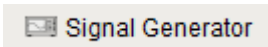
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Signal Generator



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 `Period` property, which is in milliseconds. You configure the resolution by setting the `Values/Period` property.

For example, if you choose a sine wave signal with a period of 2000 milliseconds and 10 values/period, 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

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page

Customizers

This component does not have any custom properties.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Tables

The following components give you various types of tables for displaying values.

📁 **Table**

📁 **Power Table**

📁 **List**

📁 **Tree View**

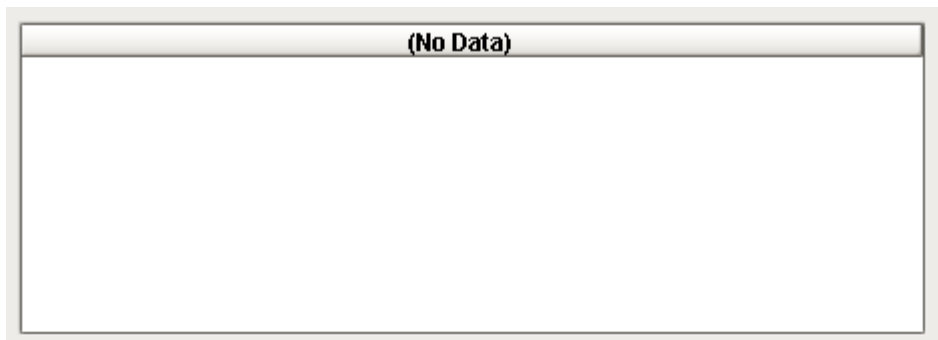
📁 **Comments Panel**

📁 **Tag Browse Tree**

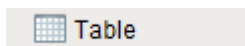
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Table



Component Palette Icon:



Description



INDUCTIVE
UNIVERSITY

Table

[Watch the video](#)

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 be 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 that's inside a dataset, 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 then respond to the resulting `cellEdited` event with an event handler and persist the data.

Exporting to HTML

You can export the table to an HTML file that retains the table's formatting. To do this, use a script like this:

Python Scripting

```
# Get a reference to the table and call the exportHTML() function  
table = event.source.parent.getComponent("Table")  
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:

Python Scripting

```
# Get a reference to the table and call the exportCSV() finction  
table = event.source.parent.getComponent("Table")  
table.exportCSV("MyTable.csv", 1)
```

Printing

Printing a table is a snap! Simply use the table's built in print function like this:

Python Scripting

```
# Get a reference to the table and call the print() finction  
table = event.source.parent.getComponent("Table")  
table.print()
```

Properties

Name	Description	Property Type	Scripting	Category
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Auto-Resize Mode	Determines how the table resizes the columns.	int	.autoResizeMode	Behavior
Background Color	The background color of the component.	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. NOTE that the border is unaffected by rotation.	Border	.border	Common
Column Attributes Data	The dataset describing the column attributes.	Dataset	.columnAttributesData	Data

Name	Description	Property Type	Scripting	Category
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.	int	.cursorCode	Common
Data	The data for this table.	Dataset	.data	Data
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	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

Name	Description	Property Type	Scripting	Category
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance
Grid Line Color	The color used to draw grid lines.	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.	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 data is filled in.	int	.initialRowSelection	Behavior
Name	The name of this component.	String	.name	Common

Name	Description	Property Type	Scripting	Category
Odd Row Background	The color which odd rows will be colored if background mode is 'Alternating'.	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
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	boolean	.rowSelectionAllowed	Behavior

Name	Description	Property Type	Scripting	Category
	Allowed flag to determine whether not whole-rows, whole-columns, or both (single-cells) are selectable.			
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.	Color	.selectionBackground	Appearance
Selection Foreground	The foreground color of a selected cell.	Color	.selectionForeground	Appearance

Name	Description	Property Type	Scripting	Category
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

Scripting

See the [Table Scripting Functions](#) page for the full list of scripting functions available for this component.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page.

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.

- [Table Customizer](#)
- [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)
```

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

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 of 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, and more.

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.

Property	Description
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

Property	Description
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.

Examples

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

 Dataset Viewer

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	Maintenance	
Ingan	Nickel Dot	Production Run	
Alingap	Rinse 2	Engineering Run	
TBD	Reactor D	Idle	1

Column Name: ---- Column Type: ----

OK

Cancel



This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Table Scripting Functions

This page details the various component and extension functions available for Vision's Table component.

Component Functions

.addRow(newRow)

- 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
- Scope
 - Client

.deleteRow(rowIndex)

- Description
 - Deletes a row from the table's dataset.
- Parameters
 - int **rowIndex** - The index of the row to delete.
- Return
 - Nothing
- Scope
 - Client

.exportCSV(filename, showHeaders)

- 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.
- Scope
 - Client

.exportHTML(filename, headerName, cellSpacing)

- Description
 - Prompts the user to save the table's data as an HTML file.
- Parameters
 - String **filename** - A suggested filename for the user. For example: "table_data.html"
 - String **headerName** - A name for the HTML header.
 - Int *cellSpacing* - A table width in pixels.
- Return
 - String - The path to the saved file, or null if the operation was cancelled.
- Scope
 - Client

.getDataAsHTML(title, width)

- 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.
- Scope
 - Client

.getRowsInViewOrder()

- 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
- Scope
 - Client

.getSelectedColumn()

- Description
 - Returns the index of the currently selected column, or -1 if none is selected.
- Parameters
 - none
- Return
 - int
- Scope
 - Client

.getSelectedColumnCount()

- Description
 - Returns the number of columns that are currently selected.
- Parameters
 - none
- Return
 - int
- Scope
 - Client

.getSelectedRow()

- Description
 - Returns the index of the currently selected row, or -1 if none is selected.
- Parameters

- none
- Return
 - int
- Scope
 - Client

.getSelectedRows()

- Description
 - Returns a list of the indexes of the selected row, or none if none is selected.
- Parameters
 - none
- Return
 - List, None
- Scope
 - Client

.getSelectedRowCount()

- Description
 - Returns the number of rows that are currently selected.
- Parameters
 - none
- Return
 - int
- Scope
 - Client

.isCellSelected(row, column)

- 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
- Scope
 - Client

.isColumnSelected(column)

- Description
 - Tests whether the given column is currently selected or not.
- Parameters
 - int **column**- The column to test.
- Return
 - boolean
- Scope
 - Client

.isRowSelected(row)

- Description
 - Tests whether the given row is currently selected or not.
- Parameters
 - int **row** - The row to test.
- Return
 - boolean
- Scope
 - Client

.print(fitWidth, headerFormat, footerFormat, showDialog, landscape)

- 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.
- Scope
 - Client

.setColumnLabel(column, label)

- 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 header label.
 - String **label** - The new header label.
- Return
 - nothing
- Scope
 - Client

.setColumnSelectionInterval(index0, index1)

- 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.
- Scope
 - Client

.setColumnWidth(column, width)

- 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
- Scope
 - Client

.setRowSelectionInterval(index0, index1)

- 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.
- Scope
 - Client

.setSelectedColumn(column)

- Description
 - Sets the given column to be the selected column.
- Parameters
 - int **column** - Column to select.
- Return
 - nothing
- Scope
 - Client

.setSelectedRow(row)

- Description
 - Sets the given row to be the selected row.
- Parameters
 - int **row** - Row to select.
- Return
 - nothing
- Scope
 - Client

.setValue(row, column, value)

- 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
- Scope
 - Client

.sortByColumn(columnName [, asc])

- 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
- Scope
 - Client

.sortOriginal()

- 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
- Scope
 - Client

.updateRow(rowIndex, changes)

- 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
- Scope
 - Client

Extension Functions

getBackgroundAt

- Description
 - Called for each cell, returns the appropriate background color. Do not block, sleep, or execute any I/O; called on painting thread.
- Keyword Args
 - 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
- Scope
 - Client

getForegroundAt

- Description

- Called for each cell, returns the appropriate foreground (text) color. Do not block, sleep, or execute any I/O; called on painting thread.
- Keyword Args
 - 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
- Scope
 - Client

getDisplayTextAt

- 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.
- Keyword Args
 - 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
- Scope
 - Client

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Power Table

Int Column  1	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 DEEED4F		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

Description



INDUCTIVE
UNIVERSITY

Power Table

[Watch the video](#)

The power table is a much more customizable version of the table component, and has many more features. The power table contains 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.

 CAUTION

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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
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.	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	Dataset	.columnAttributesData	Appearance

Name	Description	Property Type	Scripting	Category
	initialized until the customizer is opened and the OK button is pressed.			
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-	String	.defaultColumnView	Appearance

Name	Description	Property Type	Scripting	Category
	selected ordering.			
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.	int	.cursorCode	Common
Data	The data for this table.	Dataset	.data	Data
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Edit Click Count	The number of clicks required to start editing a cell.	int	.clickCountToStart	Behavior

Name	Description	Property Type	Scripting	Category
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
Grid Line Color	The color used to draw grid lines.	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	int	.propertiesLoading	Uncategorized

Name	Description	Property Type	Scripting	Category
	loaded. (Read only. Usable in bindings and scripting.)			
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	int	.selectedColumn	Data

Name	Description	Property Type	Scripting	Category
	column, or -1 if none.			
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.	Color	.selectionBackground	Appearance
Selection Foreground	The default foreground color of selected cells.	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

Name	Description	Property Type	Scripting	Category
TestData	Toggle this property to True to fill in the table's data with random data. It will immediately reset itself to False but leave the data.	boolean	.test	Misc
View Dataset	A read-only copy of the data as it appears on screen in the table.	Dataset	.viewDataset	Data
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

See the [Power Table Scripting Functions](#) page for the full list of scripting functions available for this component.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page.

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).

- [Power Table Customizer](#)
- [Component Customizers](#)
- [Understanding Component Customizers](#)

Example

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.
```


This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

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, yyyy h:mm a	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

Description

The **Power Table** offers the same functionality as the classic **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, select a number format and date format style, and more.

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.

- [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 Example

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	D31CAA2	Toyota	4 Runner	2001
2017-01-08	☐	\$ 199	737F701F	Ford	Escape	2010

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

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Power Table Scripting Functions

This page details the various component and extension functions available for Vision's Power Table component.

Component Functions

.getSelectedColumns()

- 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.
- Scope
 - Client

.getSelectedRows()

- 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.
- Scope
 - Client

.print([fitWidth] [, headerFormat] [, footerFormat] [, showDialog] [, landscape])

- 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.
- Scope
 - Client

.setColumnWidth(column, width)

- Description

- Used to set a column's width at runtime.
- Parameters
 - int **column** - Column to adjust.
 - int **width** - Width in pixels.
- Return
 - Nothing
- Scope
 - Client

Extension Functions

configureCell

- 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 overridden 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
- Scope
 - Client

configureEditor

- Description
 - Provides a chance 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 attribute will be ignored.
- 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
- Scope
 - Client

configureHeaderStyle

- Description
 - Provides a chance to configure the style of each column header. Return a dictionary of name-value pairs with the designed attributes. Available 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
- Scope
 - Client

initialize

- 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
- Scope
 - Client

isCellEditable

- 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
- Scope
 - Client

onCellEdited

- 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.
- Caution:** 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
- Scope
 - Client

onMousePress

- 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
- Scope
 - Client

onDoubleClick

- 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
- Scope
 - Client

onPopupTrigger

- 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
- Scope
 - Client

onRowsDropped

- 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
- Scope

- Client

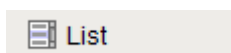
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

List



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
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common
Data	The data for the list. If multiple columns exist, the first will be used.	Dataset	.data	Data
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common

Name	Description	Property Type	Scripting	Category
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	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
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

Name	Description	Property Type	Scripting	Category
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).	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

Scripting

See the [List Scripting Functions](#) page for the full list of scripting functions available for this component.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page.

Customizers

- [Component Customizers](#)
- [Style Customizer](#)

Example

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
```

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

List Scripting Functions

This page details the various component and extension functions available for Vision's List component.

Component Functions

.addSelectionInterval(start, end)

- Description
 - Adds the options at indexes start through end (inclusive) to the selected options.
- Parameters
 - int **start** - The first index (starting at 0) to add to the selection.
 - int **end** - The last index (starting at 0) to add to the selection.
- Return
 - Nothing
- Scope
 - Client

.clearSelection()

- Description
 - Clears the current selection, making nothing selected.
- Parameters
 - Nothing
- Return
 - Nothing
- Scope
 - Client

.getSelectedIndices()

- 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
- Scope
 - Client

.getSelectedValue()

- Description
 - Returns the currently selected value, or None if the selection is empty.
- Parameters
 - Nothing
- Return
 - Object
- Scope
 - Client

.getSelectedValues()

- Description
 - Returns a list of the currently selected values. Returns an empty list if the selection is empty.
- Parameters
 - Nothing
- Return
 - Object[]
- Scope
 - Client

.isSelectedIndex(index)

- Description
 - Checks whether or not the given index is currently selected.
- Parameters

- int index
- Return
 - boolean
- Scope
 - Client

.isEmpty()

- Description
 - Checks to see if anything is selected in the list or not.
- Parameters
 - Nothing
- Return
 - boolean
- Scope
 - Client

.setValue(value)

- Description
 - Sets the currently selected value to the argument, if found in the list.
- Parameters
 - Object **value**
- Return
 - Nothing
- Scope
 - Client

Extension Functions

This component does not have extension functions associated with it.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Tree View



Component Palette Icon:



Description

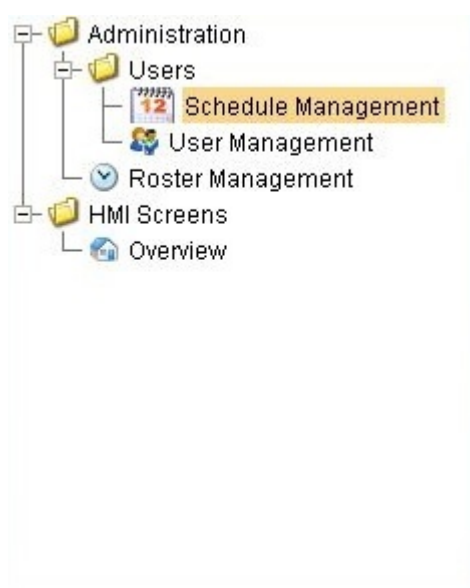
The Tree View component can display any tree hierarchy. It is configured by filling in a dataset. Each row in the dataset will become a node in the tree. Each node has a path, for example, "West Area/Process/Valve1" that determines its location in the tree. The Separation Character property (by default it is 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. This component recognizes the following column titles. The references to optional column titles means that a dataset does not need to have them present in the dataset for the tree to render and function.

- Path - the path determines the node's location. Broken up into a list by splitting on the separation character.

- Text - the text of the node while not selected.
- Icon - a path to an icon for the node. Use the value: "default" to use the tree automatic folder/leaf icons. [optional]
- Background - 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]
- Foreground - a string representation of the unselected foreground color. [optional]
- Tooltip - if not empty, will be used 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]
- SelectedText - the text of the node while selected.[optional]
- SelectedIcon - a path to an icon for the node while selected. Use the value: "default" to use the tree automatic folder/leaf icons.[optional]
- SelectedBackground - a string representation of the selected foreground color.[optional]
- SelectedForeground - a string representation of the selected foreground color.[optional]
- SelectedTooltip - if not empty, will be used as the tooltip for the node while selected.[optional]
- SelectedBorder - a string that will be coerced into a Border for the node while selected. May be empty.[optional]

Below is an example configuration of the treeview's items property. Notice how not all of the fields listed above are used, because there are certain properties that are not necessary to build our treeview. Notice how I have chosen a larger version of the same images for the Selected Icon, 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
HMI Screens	Overview	Builtin/icons/16/home.png	color(255, 255, 255, 255)	color(0, 0, 255)
Administration/Users	User Management	Builtin/icons/16/users3.png	color(255, 255, 255, 255)	color(0, 0, 255)
Administration/Users	Schedule Management	Builtin/icons/16/calendar.png	color(255, 255, 255, 255)	color(0, 0, 255)
Administration	Roster Management	Builtin/icons/16/clock.png	color(255, 255, 255, 255)	color(0, 0, 255)

Properties

Name	Description	Property Type	Scripting	Category
Antialias	Draw with antialias on? Makes text smoother.	boolean	.antialias	Appearance
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.	Color	.background	Appearance

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border	.border	Common
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
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.	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	Color	.defaultForeground	Appearance

Name	Description	Property Type	Scripting	Category
	a node if no foreground is set.			
Default Node Selected Background	The default selected background of a node if no background is set.	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.	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

Name	Description	Property Type	Scripting	Category
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
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	int	.selectionMode	Behavior

Name	Description	Property Type	Scripting	Category
	regions does the tree allow.			
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

Scripting

See the [Tree View Scripting Functions](#) page for the full list of scripting functions available for this component.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page.

Customizers

The tree view customizer allows for easy custom manipulation of the tree view components underlying formatting.

- [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)
```


This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Tree View Example



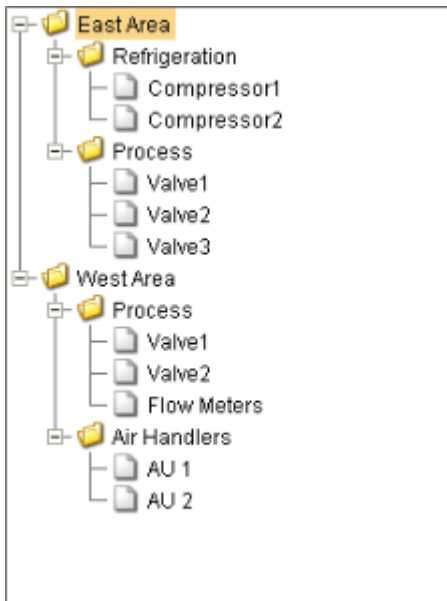
INDUCTIVE
UNIVERSITY

Tree View

[Watch the video](#)

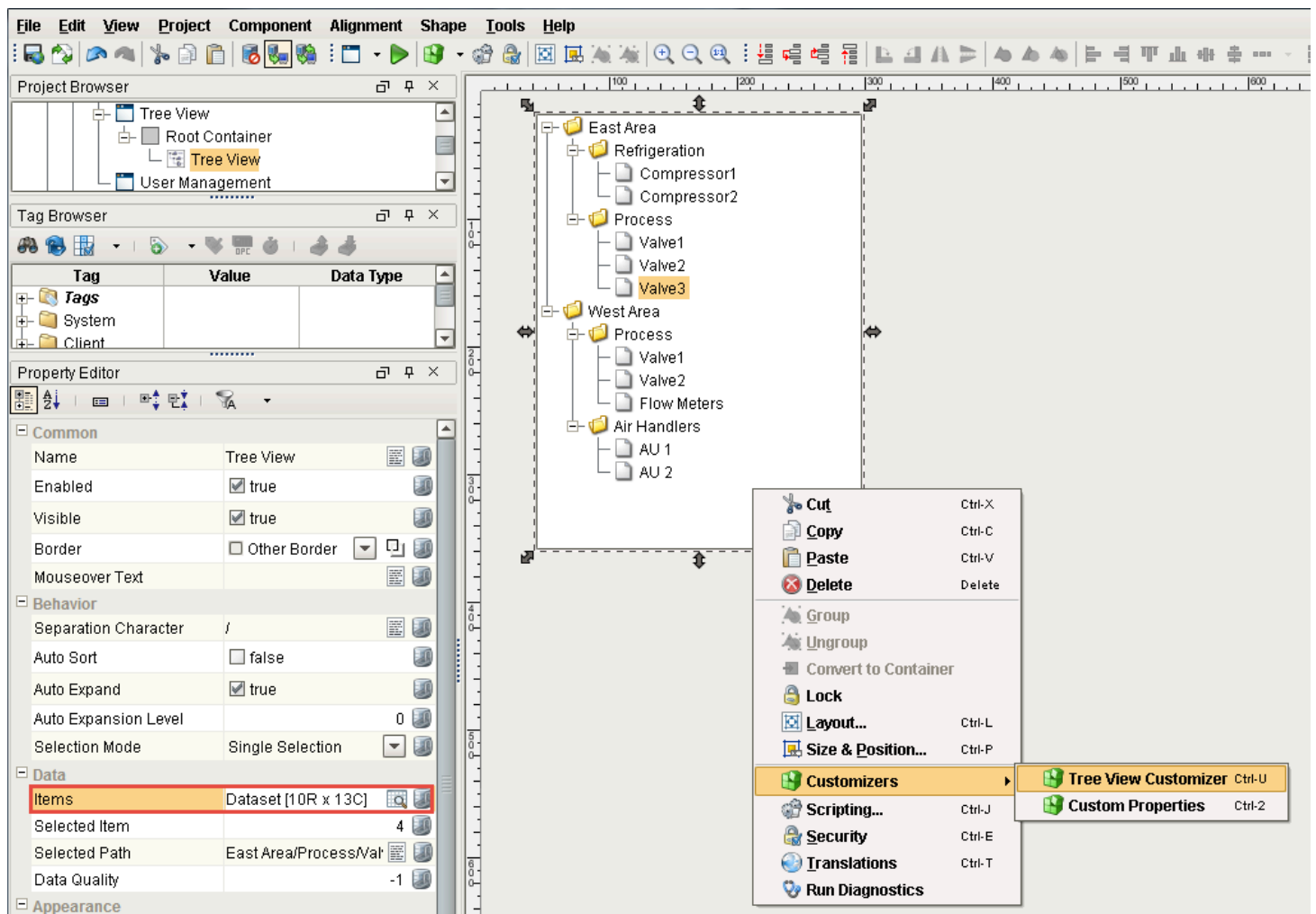
The **Tree View** component displays and organizes data in a dataset in a tree hierarchy. The Tree View component is found under the Tables tab of the component palette.

Drag the Tree View component on to your window. In **Preview Mode**, expand the folders to see the data. The Tree View component can be used for navigation in Ignition.

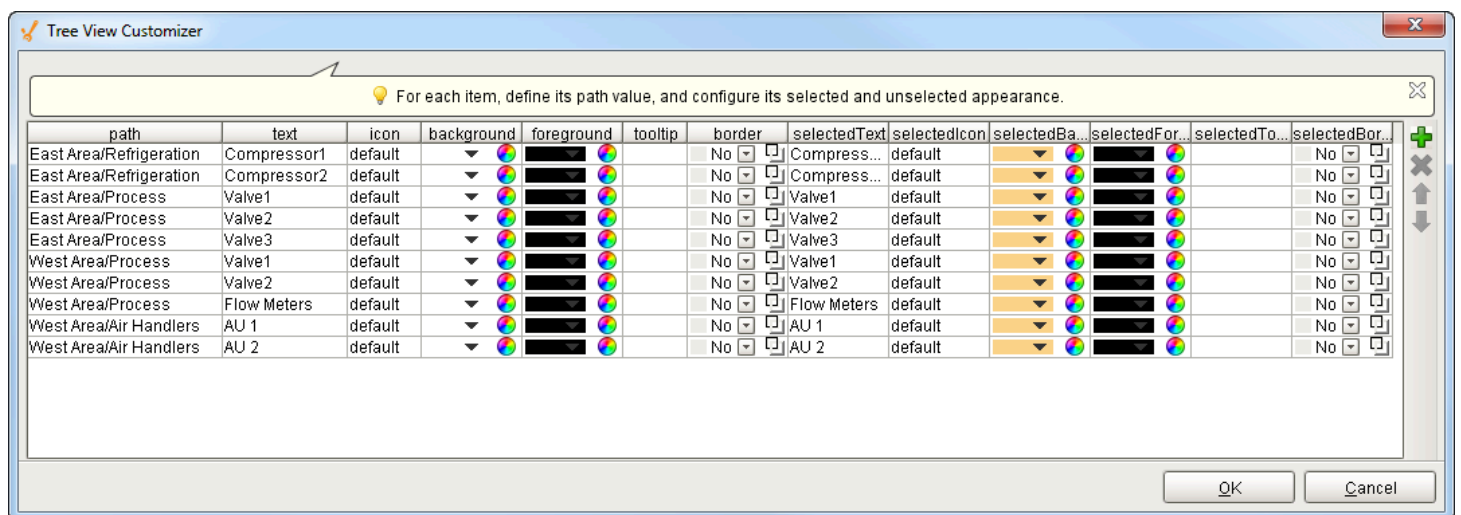


In fact, Ignition has a built in skeleton project called Multi-tier Navigation that uses the Tree View component specifically for that purpose. There is one difference between the Multi-tier Navigation and the Tree View component. The Multi-tier Navigation has one extra column in the Dataset Viewer called **windowPath** that does not show up in the Tree View Customizer. You will need to open the Dataset Viewer to modify it.

To configure the Tree View component, go to the **Data property** called **Items** next to the Dataset Viewer. This contains all the items in the tree. **Right-click** on the **Tree View component** and scroll down to **Customizers > Tree View Customizers** to configure these items.

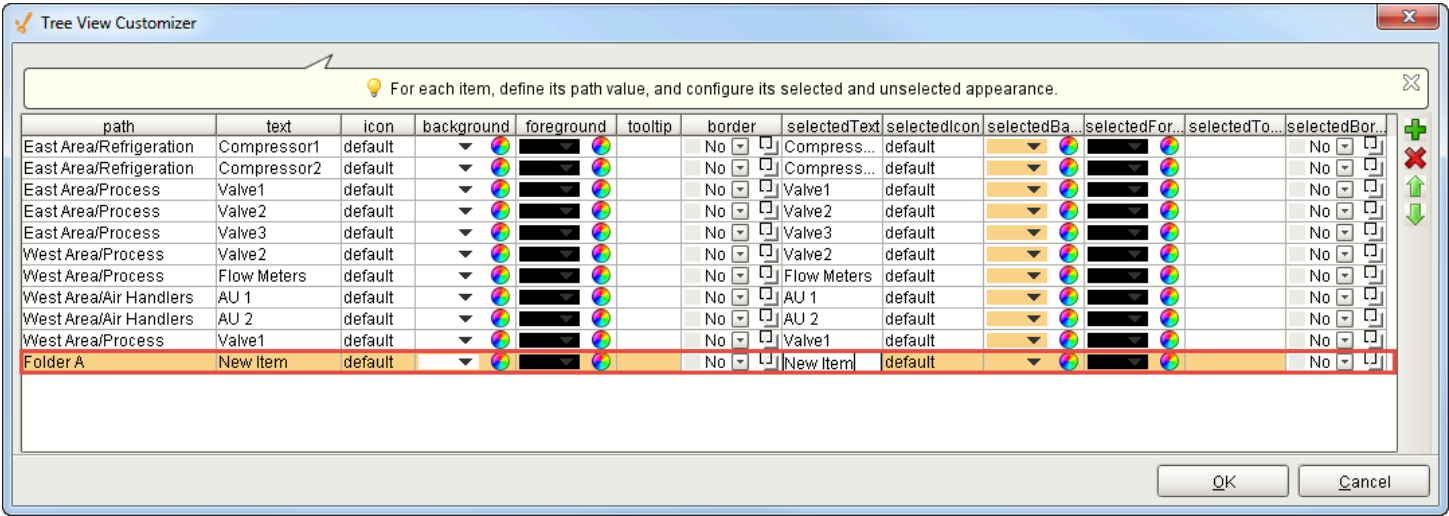


You can add one or more rows of data in the Tree View Customizer. Every row must have a **path** and **text** to display. In the example below, look at the first and second rows. The path is East Area/Refrigeration. This path denotes which folder to put an item inside of. The path is separated by a '/' which is a **Separation Character** by default, and shows that there is an East Area inside a Refrigeration folder containing 2 items called Compressor 1 and Compressor 2. You will notice in the Tree View Customizer that you have other columns you can customize based on how you want particular items to look.

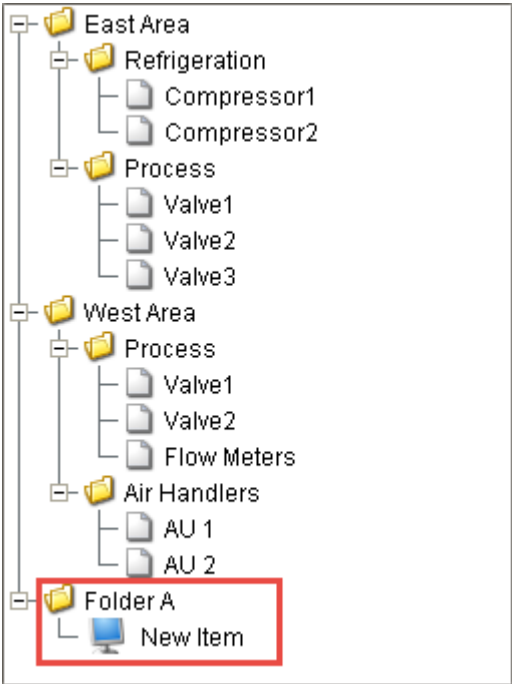


In the following example, you can add another row after the last item in the Tree View Customizer window and set the path to **Folder A**. For this example, enter the name of your item as 'New Item' in

the **text** field. You can also set an icon, the foreground, a tooltip, border, when the item is selected and when it's not selected if you desire Make sure the name of the item is the same for the **text** and **selectedText** fields. You can add one or more rows and move them up and down.



When you press **OK**, your new row will be saved in the dataset and you can see it on the Tree View. If you expand Folder A, you will see the 'New Item' with the icon that you specified.

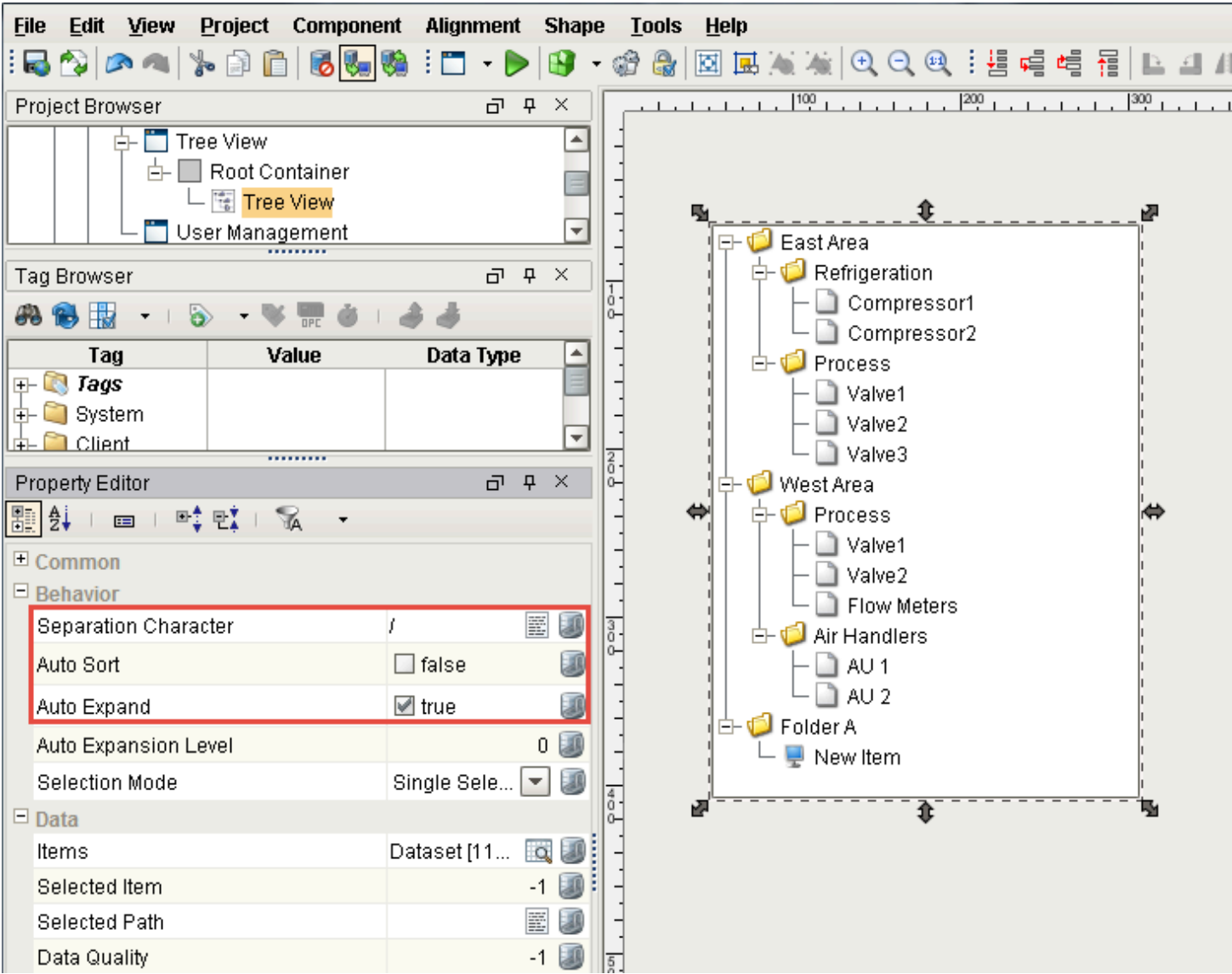


The Tree View component has a couple of other properties that are important like the **Separation Character** which we are using in our path name. By using the **Separation Character** or '/', the path is broken up into folders.

There is a property called **Auto Sort** that automatically sorts all the items in the tree into alphabetical order. If you uncheck **Auto Sort**, it will be sorted by the order of the dataset as shown in the example below.

There is another property called **Auto Expand**. If set to True, the folders will all automatically be expanded. If your Tree View is not expanded, right click to close the window, and by opening it again

you will see that all the folders are now expanded in the Tree View.



Now, the dataset has all items shown in the tree view. You can bind this to a SQL query to bring back data from the database in order to organize this data into the tree.

File Edit View Project Component Alignment Shape Tools Help

Project Browser

- Tree View
 - Root Container
 - Tree View
 - User Management
- Popups
 - Motor Detail

Tag Browser

Tag	Value	Data Type
Tags		
Data Types		
Device Status		
gensim		
M1		

Property Editor

Common

Name	Tree View
Enabled	☒ true
Visible	☒ true
Border	☐ Other Border
Mouseover Text	

Behavior

Separation Character	/
Auto Sort	☐ false
Auto Expand	☐ false
Auto Expansion Level	0
Selection Mode	Single Selection

Data

Items	Dataset [11R x 13C]
Selected Item	-1
Selected Path	
Data Quality	-1

East Area

- Refrigeration
 - Compressor1
 - Compressor2
- Process
 - Valve1
 - Valve2
 - Valve3
- West Area
 - Process
 - Valve1
 - Valve2
 - Flow Meters
 - Air Handlers
 - AU 1
 - AU 2
- Folder A
 - New Item

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Tree View Scripting Functions

This page details the various component and extension functions available for Vision's Tree View component.

Component Functions

.clearSelection()

- Description
 - Clears the current selection.
- Parameters
 - Nothing
- Return
 - Nothing
- Scope
 - Client

.collapseAll()

- Description
 - Collapses all nodes in the tree.
- Parameters
 - Nothing
- Return
 - Nothing
- Scope
 - Client

.expandAll()

- Description

- Expands all nodes in the tree.
- Parameters
 - Nothing
- Return
 - Nothing
- Scope
 - Client

.getSelectedItems()

- 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
- Scope
 - Client

.getSelectedPaths()

- 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
- Scope
 - Client

Extension Functions

This component does not have extension functions associated with it.

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Comments Panel

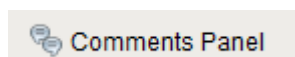
! INFO

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".

+ Add Note	
Bob Feb 28, 15 4:33 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.
Derek Mar 2, 15 4:53 AM	Can we expedite this? Not having any inverse reactive current is really a becoming a problem, starting to get some sinusoidal deplenaration.
Bob Feb 28, 15 4:18 AM	Found it. One of the cardinal gram-meters was unsynchronized. Ordering a new one.
Jane Feb 26, 15 7:10 PM	It's happening on our shift too. Maybe its the hydrocoptic marzel veins?
Bob Feb 24, 15 1:56 PM	The turbo encabulator is faulting frequently...

Component Palette Icon:



Description



INDUCTIVE
UNIVERSITY

Comments Panel

[Watch the video](#)

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

NOTE

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.

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)

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	

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

Binding Types

- ☐ Tag
 - ☐ Tag History
 - ☐ Tag
 - ☐ Indirect Tag
- ☐ Property
 - ☐ Property
 - ☐ Expression
- ☒ Database
 - ☐ DB Browse
 - ☒ SQL Query
- ☐ Other
 - ☐ Cell Update
 - ☐ Functions

SELECT Query

```
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
```

Database Connection

Polling Mode

- ☐ Off
- ☒ Relative
- ☐ Absolute

Property Editor

Tag	Value	Data Type
Tags		
System		
Client		
All Providers		

Data

Tag	Value	Data Type
Data	Dataset [4R x 6C]	
Data Quality	192	

mary First Mary note
Mar 8, 16 3:47 PM
admin First admin note

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
Antialias	Enabled antialiasing when drawing the component.	boolean	.antialias	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. NOTE that 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
Data Quality	The data quality code for any tag bindings	int	.dataQuality	Data

Name	Description	Property Type	Scripting	Category
	on this component.			
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.	Color	.foreground	Appearance
Header Color	The background color of the header notes.	Color	.headersColor	Appearance

Name	Description	Property Type	Scripting	Category
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.	Color	.noteColor	Appearance
Padding	The amount of padding between the notes.	int	.padding	Appearance
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.	Color	.stickyHeaderColor	Appearance
Sticky Note Color	The background color for sticky notes.	Color	.stickyNoteColor	Appearance

Name	Description	Property Type	Scripting	Category
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

Scripting

See the [Comments Panel Scripting Functions](#) page for the full list of scripting functions available for this component.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page.

Customizers

- [Component Customizers](#)

Examples

CAUTION

The following examples may need to be modified to match the table and column names in your database.

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 2 since it's an escape
character
    pathAndFile = filename.rsplit('\\', 1)
    filename = pathAndFile[1]

# build the query and the arguments
query = "INSERT INTO Notes (note, whoid, tstamp, attachment, filename, sticky)
VALUES (?, ?, CURRENT_TIMESTAMP, ?, ?, ?)"
arguments = [note, user, attachmentBytes, filename, sticky]
# insert the note
insertId = system.db.runPrepUpdate(query, arguments)

# insert a row onto the itemNotes table
# replace 'MYID' with the proper code to fetch your id
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]

# insert the note. You may need to modify this query depending on how your
# database handles binary data.

#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 on this line, which is
#the preferred datatype for binary data in MSSQL.
#query = "INSERT INTO Notes (note, whoid, tstamp, attachment, filename, sticky)
VALUES (?, ?, CURRENT_TIMESTAMP, CONVERT(VARBINARY(MAX),?), ?, ?)"

arguments = [note, user, attachmentBytes, filename, sticky]
system.db.runPrepUpdate(query, arguments)

```

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Comments Panel Scripting Functions

This page details the various component and extension functions available for Vision's Comments Panel component.

Component Functions

This component does not have component functions associated with it.

Extension Functions

insertNote

- 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
- Scope
 - Client

deleteNote

- 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
- Scope
 - Client

unstickNote

- 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
- Scope
 - Client

downloadAttachment

- 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
- Scope
 - Client

canDelete

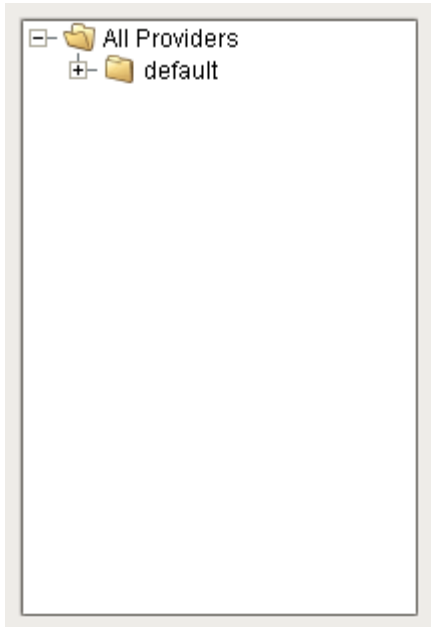
- Description
 - Returns whether or not a note with the given id can be deleted. Notes that return True will show a 'delete' link.
- Parameters
 - component **self** - A reference to the component that is invoking this function

- 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.
- Scope
 - Client

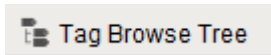
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Tag Browse Tree



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
Antialias	Draw with antialias on? Makes text smoother.	boolean
Border	The border surrounding this component. NOTE that the border is unaffected by rotation.	Border
Data Quality	The data quality code for any tag bindings on this component.	int
Include Historical Tags	Whether or not to display historical tags.	boolean
Include Realtime Tags	Whether or not to display non-historical tags.	boolean
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String
Name	The name of this component.	String
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/	String

Name	Description	Property Type
	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 <pre>[DB``/``ignition:default]machine_1``/</pre>	
Selected Paths	Contains the paths that should be selected on the tree which should be in the format of a single string column.	DataSource
Selection Mode	What kind of selection regions does the tree allow.	int
Show Root Handles	Whether or not to show handles next to parent nodes.	boolean
Show Root Node	Whether or not to show the root node of the tree.	boolean
Tag Tree Mode	Choose whether the tree is built using tags from the default provider or the historical provider.	boolean, treeView
Visible	If disabled, the component will be hidden.	boolean

Scripting

See the [Tag Browse Tree Scripting Functions](#) page for the full list of scripting functions available for this component.

Event Handlers

Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the [Component Events](#) page.

Customizers

- Component Customizers

Example

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.value.value)  
    system.gui.messageBox(value)  
  
def showLastChange(self):  
    lastChange = str(clickedTag.value.timestamp)  
    system.gui.messageBox(lastChange)  
  
itemsDict = {"Show Value": showValue, "Show Last Change": showLastChange}  
JPopupMenu = system.gui.createPopupMenu(itemsDict)  
return JPopupMenu
```

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Tag Browse Tree Scripting Functions

This page details the various component and extension functions available for Vision's Tag Browse Tree component.

Component Functions

This component does not have component functions associated with it.

Extension Functions

filterTag

- 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
- Scope
 - Client

createPopupMenu

- 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
- Scope
 - Client

This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Web Browser

The following component gives you the ability to add a web browser to your client.



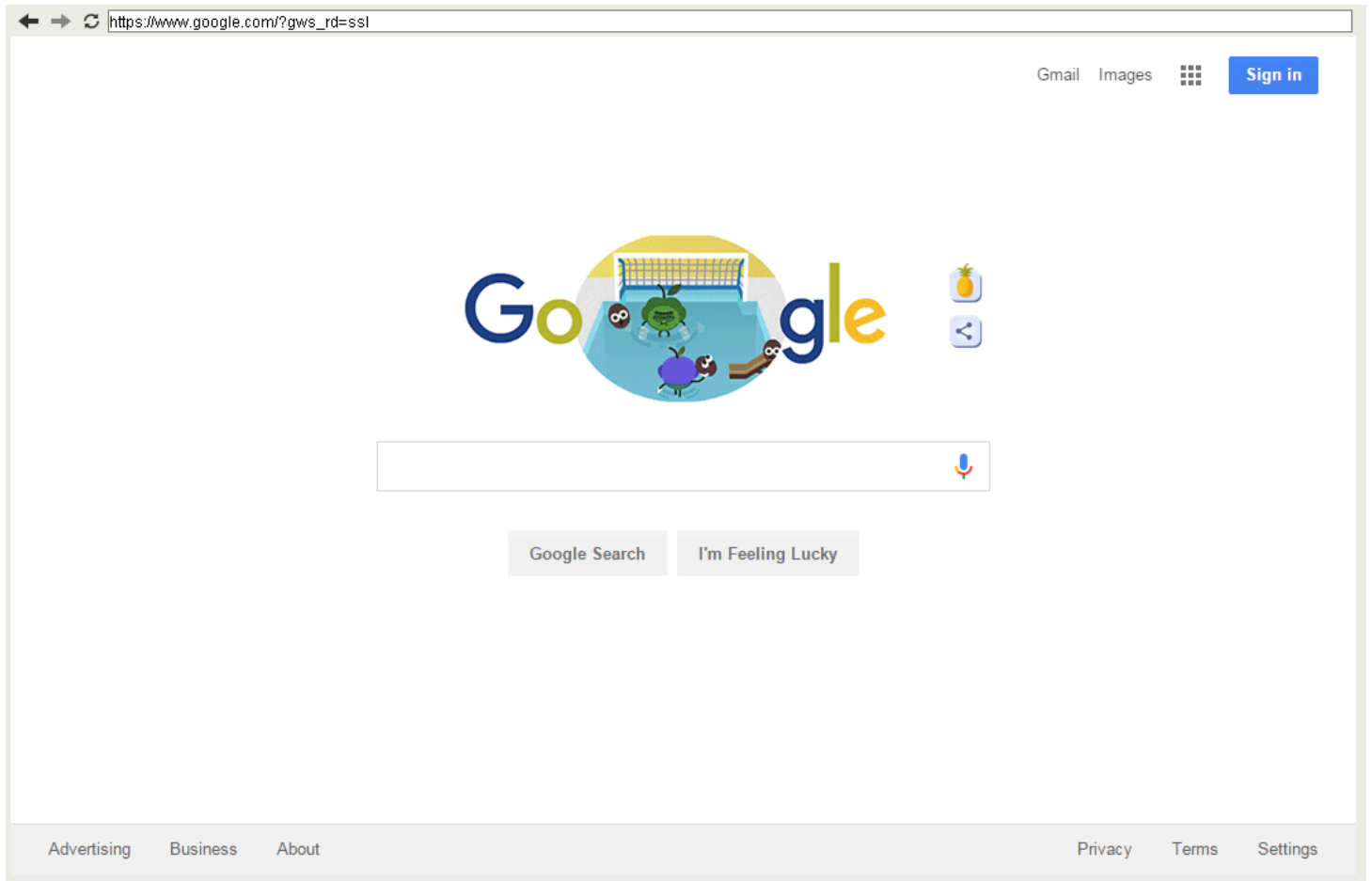
Web Browser Component

The **Web Browser** component in **Designer** allows you to embed a full web browser inside of an Ignition Client.

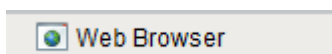
This is documentation for **Ignition 7.9**, which is no longer actively maintained.

For documentation on the latest stable release, see the [latest version](#) (8.3).

Web Browser Component



Component Palette Icon



Description



INDUCTIVE
UNIVERSITY

Web Browser

[Watch the video](#)

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.

CAUTION

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.

New in 7.9.8 As of 7.9.8, the underlying JxBrowser was updated and now has a new set of system requirements. If your Web Browser used to work, but does not meet the requirements of the newer versions, it is possible to install an older Web Browser module on a newer Ignition Gateway.

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.

CAUTION

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.

NOTE

The Web Browser Component will only support the following 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

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Note, that the border is unaffected by rotation.	Border	.border	Common
Enabled	If disabled, a component cannot be used. New in 7.9.13 The "Enabled" property is no longer available for the Web Browser component when attempting to bind with another component. Binding through scripting and tags is still available. If there was a property binding setup before the upgrade, then that will still be active and function to a certain extent. More information can be found at JxBrowser Support.	boolean	.componentEnabled	Common
FTP Proxy Port	FTP Proxy Port sets the proxy port for FTP connections.	int	.ftpProxyPort	Data

Name	Description	Property Type	Scripting	Category
	This setting is only used when Use Proxies is checked.			
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

Name	Description	Property Type	Scripting	Category
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 <code>"*foo.com,
<local>,127.0.1"</code> This setting is only used when Use Proxies is checked.	String	.proxyExceptions	Data
Proxy Password	The password to use for proxy	String	.proxyPassword	Data

Name	Description	Property Type	Scripting	Category
	authentication. This setting is only used when Use Proxies and Use Proxy Authentication are checked.			
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
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 <code><html><body>&nbsp;</body></html></code> 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
Use Proxies	If checked, the Web Browser will try to	boolean	.useProxies	Data

Name	Description	Property Type	Scripting	Category
	use the proxy settings.			
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

.getBrowser()

- Description
 - This function will return the underlying browser object. See [JxBrowser Programmer's Guide](#) for more information.
- Parameters

- none
- Return
 - **Object** - The Browser Object
- Scope
 - Client

Extension Functions

initialize

New in 7.9.13

- Description
 - Called when the Web Browser component is initialized. Provides a chance to initialize the browser further. Enabling or disabling this function will cause the Web Browser component to re-initialize.
- Key Arguments
 - **Component** self: A reference to the component that is invoking this function.
 - **JxBrowser** browser: The underlying JxBrowser instance of the Browser class.
 - **BrowserView** browserView: The underlying rendering class that contains the Browser instance.
- Return
 - Nothing
- Scope
 - Client

Event Handlers

propertyChange

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

Property	Description
.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. NOTE: 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](#)

Examples

Code Snippet

*# This code example can be used inside the initialize() extension function.
It is designed to pass the Web Browser component's internal browser logging
through to the Ignition gateway logs.*

```
from com.teamdev.jxbrowser.chromium.events import ConsoleListener
```

```
class IgnitionListener(ConsoleListener):  
    def __init__(self, target='Web Browser'):  
        self.logger = system.util.getLogger(target)  
        self.levels = {'DEBUG': self.logger.debug, 'ERROR': self.logger.error,  
            'LOG': self.logger.info, 'WARNING': self.logger.warn}  
  
    def onMessage(self, event):  
        self.levels[str(event.getLevel())](event.getMessage())  
  
browser.addConsoleListener(IgnitionListener())
```