

.
propertyNa
me

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

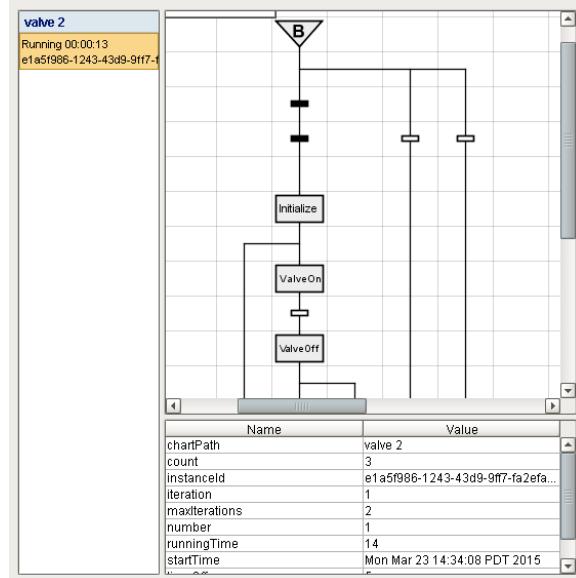
- [Vision Component Customizers](#)

Examples

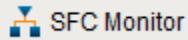
There are no examples associated with this component.

Vision - SFC Monitor

General



Component Palette Icon:



Description

A component to monitor Sequential Function Chart performance. In addition the component allows for the operator to control the chart instance through the charts instance 'id' property. The chart scoped variables are available through the scope dataset property.

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. Note: The border is unaffected by rotation.	Border	.border	Common
Instance ID	The UUID of the sequential function chart to monitor.	String	.instanceId	Data
Instance List Visible	Shows or hides the list of SFC instances on the left.	boolean	.instanceListVisible	Appearance
Legend Visible	Shows or hides the step and transition state legend.	boolean	.legendVisible	Appearance
Name	The name of this component.	String	.name	Common
Scope Dataset	Dataset containing the variables in chart scope.	Dataset	.scopeDataset	Data
Scope Table Visible	Shows or hides the chart scope inspection table.	boolean	.scopeTableVisible	Appearance

Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Zoom	The zoom multiplier to display the chart's status at.	float	.zoom	Appearance

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

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

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

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

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

shiftDown

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

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

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

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

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

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

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

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

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

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

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

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

Customizers

- [Vision Component Customizers](#)

Examples

There are no examples associated with this component.

Vision - Alarming Palette

Alarming Components

The following components give you options for displaying Alarm information.

[In This Section ...](#)

Vision - Alarm Status Table

General

	Active Time	Display Path	Priority	Event Value	Label	Current State
<input type="checkbox"/>	6/12/19, 12:25 PM	Motors/Motor 4/Amps/Low Amps	Critical	47	Low Amps	Active, Unack.
<input type="checkbox"/>	6/12/19, 12:45 PM	Motors/Motor 2/Amps/Low Amps	Critical	47	Low Amps	Active, Unack.
<input type="checkbox"/>	6/12/19, 12:45 PM	Motors/Motor 3/Amps/Low Amps	Critical	48	Low Amps	Active, Unack.
<input type="checkbox"/>	6/12/19, 12:46 PM	Motors/Motor 6/Amps/Low Amps	Critical	49	Low Amps	Active, Unack.
<input type="checkbox"/>	6/12/19, 12:46 PM	Motors/Motor 1/Amps/Low Amps	Critical	48	Low Amps	Active, Unack.
<input type="checkbox"/>	6/12/19, 12:47 PM	Motor Plant/Motor 3/Amps/Low ...	Critical	25	Low Amps	Active, Unack.
<input type="checkbox"/>	6/12/19, 12:47 PM	Motors/Motor 5/Amps/Low Amps	Critical	50	Low Amps	Active, Unack.
<input type="checkbox"/>	6/12/19, 12:47 PM	Motor Plant/Motor 1/Amps/Low ...	Critical	23	Low Amps	Active, Unack.
<input type="checkbox"/>	6/12/19, 12:47 PM	Ramp/Ramp&OPC Alarm	High	316.6133	OPC Alarm	Active, Unack.
<input type="checkbox"/>	6/11/19, 3:26 PM	Tank Level 2/Low SP2	Critical	22	Low SP2	Cleared, Unack.
<input type="checkbox"/>	6/12/19, 12:17 PM	Motors/Motor 4/Amps/Low Amps	Critical	53	Low Amps	Cleared, Unack.

Acknowledge | Shelve |    

Component Palette Icon:



Description

The alarm status table displays the current state of the alarm system. It can be configured to show active, unacknowledged, cleared, and acknowledged alarms. By default it shows all non-cleared/non-ack'd alarms.

Acknowledgement is handled by selecting (checking) alarms and pressing the "Acknowledge" button. If any of the selected alarms require acknowledgement notes, then a small text area will be presented in which the operator must add notes to the acknowledgement.

Note: The Alarm Status Table component allows you to select an individual alarm, multiple alarms, or the Select All checkbox in the header bar. You can also use the Shift+Click multi select feature to select a range of alarms for acknowledging and shelving. Check one alarm and Shift+Click another alarm several rows down. All of the alarms between them, including the one you shift clicked, will be selected.

Shelving is supported by pressing the "Shelve" button when an alarm is selected. This will temporarily remove the alarm from the entire alarm system (not just the local client). When the time is up, if the alarm is still active, it will pop back into the alarm system. The times shown to the user are customizable by editing the values inside the "Shelving Times" dataset property. The alarms that have been shelved can be un-shelved by pushing the shelf management button in the lower right-hand side of the component.

If a more simplified alarm status table is needed, many of the features of the status table can be removed, for example, the header, footer, and multi-selection checkboxes. If a very short alarm status table is needed, turn on the "Marquee Mode" option, which will automatically scroll through any alarms if there is not enough vertical space to show all of them at once.

To change the columns that are displayed, the order of the columns, and/or the column width, put the Designer into preview mode. Then right-click on the table header to show/hide columns. Click and drag to re-order columns, and drag the margins of the columns to resize their width. No further action is necessary - the column configuration will remain in place after the window is saved.

For alarms that originate from Tags that have Tag history turned on, users can see an automatic ad-hoc chart for the value of the source Tag by pressing the chart button.

Note: For information on how to configure the Alarm Status Table refer to [Alarming in Vision](#).

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border.	Border	.border	Common



Alarm Status Table

[Watch the Video](#)

	Note: The border is unaffected by rotation.			
Chart Resolution	The resolution for the ad-hoc tag historian chart.	int	.chartResolution	Behavior
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 a custom display path isn't set. Specify multiple paths by separating them with commas. Supports the wildcard "*".	String	.displayPathFilter	Filters
Duration Format	Formats styles for fields like Active and Ack durations: Long, Short, Compact, and Abbreviated. Duration Format property, allows users to format the time units on the Active Duration column.	int	.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
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
Provider Filter	Filter alarms by Tag provider. Specify multiple providers by separating them with commas. A value of "." denotes the default Tag provider.	String	.providerFilter	Filters
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Refresh Rate	The rate at which this table will poll changes to the alarm status, in milliseconds.	long	.refreshRate	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
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. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector .	Color	.selectionColor	Appearance
Selection Thickness	The size of the selection border.	int	.selectionThickness	Appearance
Shelving Times	This dataset holds the times that are suggested when shelving an alarm. Allowable units are "second", "minute", or "hour".	Dataset	.shelvingTimes	Data
Show Ack	Show the acknowledge button on the footer panel.	boolean	.showAck	Appearance

Button				
Show Active and Acked	Show alarms that are active and acknowledged.	boolean	.activeAndAcked	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 Acked	Show alarms that are cleared and acknowledged.	boolean	.clearAndAcked	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
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. Supports the wildcard "**".	String	.sourceFilter	Filters
Stay Delay	The time (in mSec) to wait between scrolls	int	.stayDelay	Behavior
Table Background	The background of the alarm table. See Color Selector .	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
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

- Description

This specialized print function will paginate the table onto multiple pages. This function accepts keyword-style invocation.

- Keyword Args

`boolean` fitWidth - If true, the table's width will be stretched to fit across one page's width. Rows will still paginate normally. If false, the table will paginate columns onto extra pages. (default = true) [optional]

`string` headerFormat - A string to use as the table's page header. The substring "{0}" will be replaced with the current page number. (default = None) [optional]

`string` footerFormat - A string to use as the table's page footer. The substring "{0}" will be replaced with the current page number. (default = "Page {0}") [optional]

`boolean` showDialog - Whether or not the print dialog should be shown to the user. Default is true. [optional]

`boolean` landscape - Used to specify portrait (0) or landscape (1) mode. Default is portrait (0). [optional]

- Return

`Boolean`- True if the print job was successful.

- Scope

Client

- 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

- 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

- 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

- 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
- 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
- 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
- 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

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

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

Customizers

Alarm Row Styles

The Alarm Status Table has a customizer.

- [Vision Component Customizers](#)

Examples

Code Snippet

```
#The following code is an example of the filter alarm expression function.  
#The function results in advanced filtering for the alarm table.  
#In this example the alarm table will only show alarms with a name that matches the value of the  
"AreaName" property located on the container the Alarm Status Table resides in.
```

```
name = self.parent.AreaName  
if name == alarmEvent.get('name'):  
    return True  
else:  
    return False
```

Gallery

Alarm Status Table with a Single Alarm

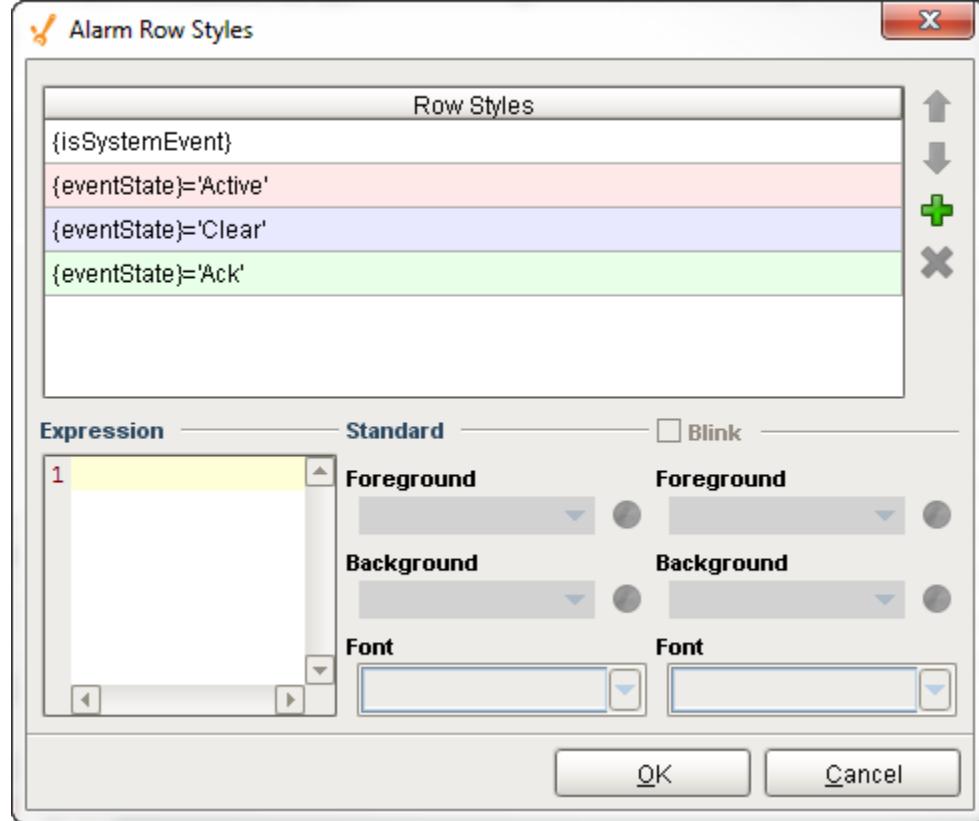
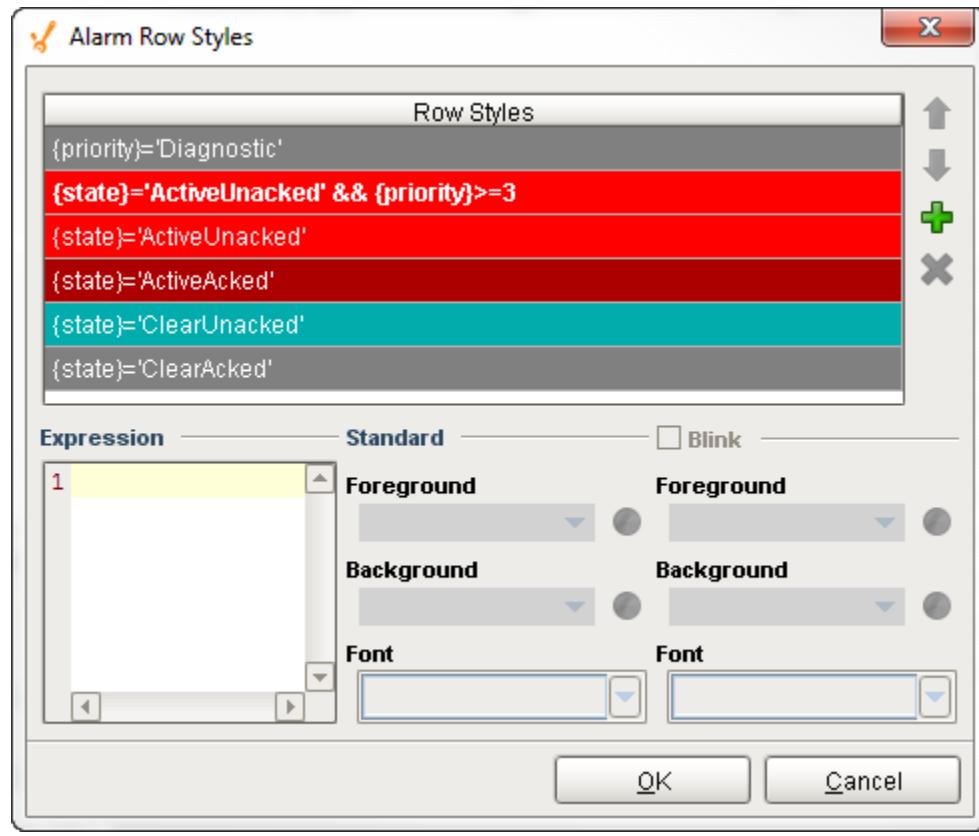
<input type="checkbox"/> Active Time	Display Path	Current State	Priority
<input type="checkbox"/> 1/20/15 8:26 AM	London	Active, Unacknowledged	Low

Vision - Alarm Row Style Customizer

Alarm Row Styles - Alarm Status Table

Alarm Row Styles - Alarm Journal Table



Description

The Alarm Row Styles Customizer manages the way the Alarm Status Table and the Alarm Journal Table render each alarm. The Alarm Row Styles Customizer allows you to change the styles of the alarms and the logic that governs each style. Both the Alarm Status Table and the Alarm Journal Table evaluate each alarm and applies the logic of the expression block to decide to implement a style. If the expression returns a logical "True" then the Alarm Row Styles Customizer applies the color formatting options defined in the area to the right of the Expression block. If the expression returns a logical "False" then the Alarm Row Styles Customizer evaluates the next expression associated with the next row style. The process continues until an expression returns a logical "True." There can be many rows with different logic and styles. You can add and remove rows by selecting the "plus" button or "delete" button.

Customizers

The Alarm Row Styles Customizer is used by both the Alarm Status Table and the Alarm Journal Table components. Each table comes with their own predefined set of colors. The Alarm Row Styles Customizer is where you can modify an existing style, add more styles, delete a style, and change the order. Each row style has an expression, a color, and the option to make it blink. The Alarm Row Styles Customizer already has some preset states and predefined styles to help you get started. It works by changing colors on each of the individual rows styles based on the state of the alarm.

Alarm Rows Styles Customizer - Property Descriptions

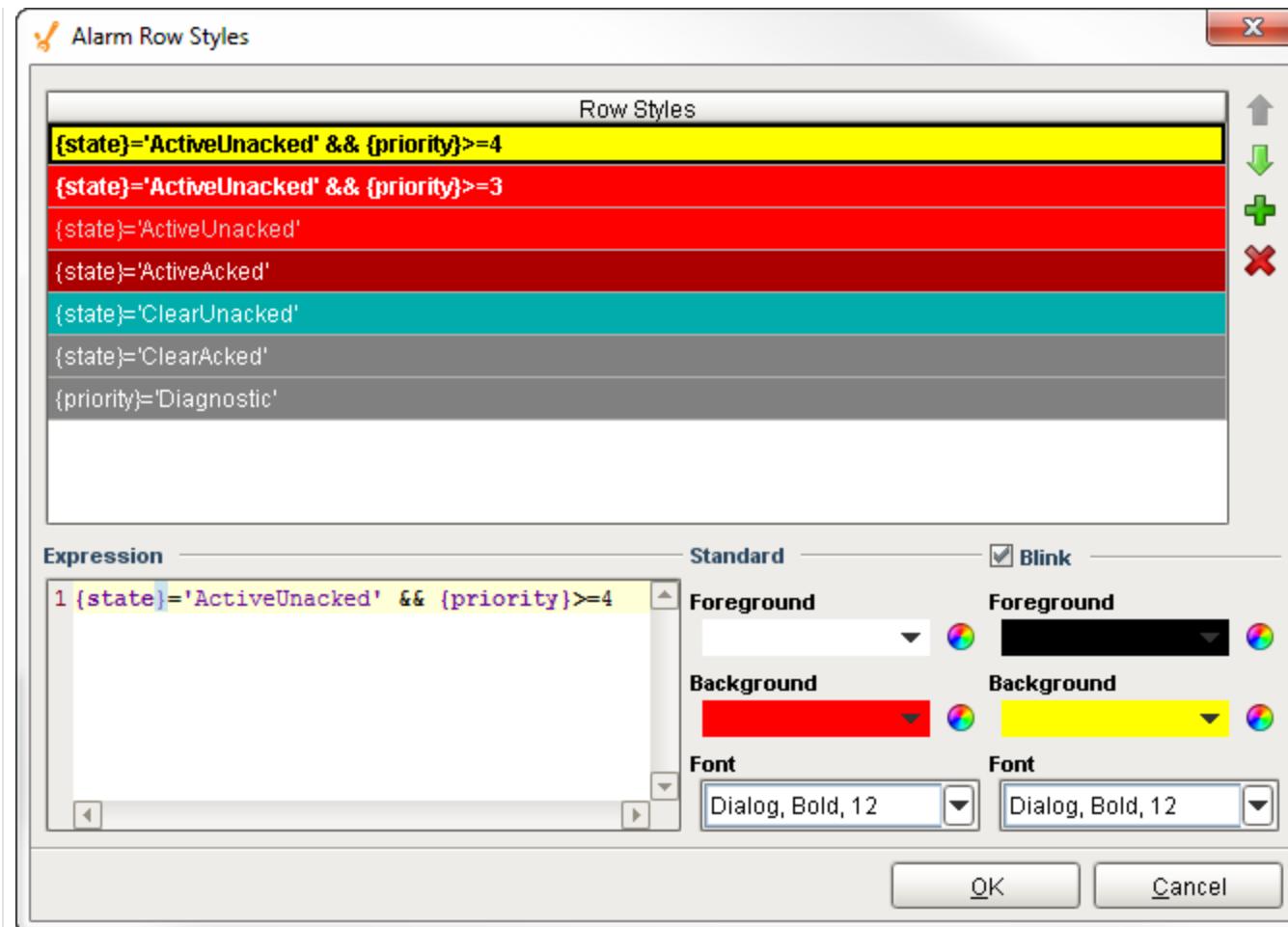
Property	Description
Row Styles	Each row has a unique style associated with each of the alarm states. You can add and delete row styles, and change the order of the rows with the up or down arrow buttons.
Expression	Each style has an expression. The expression allows you to do any evaluation you want using any parts of the alarm: Priority, State, Display Path, Active Time, 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](#)

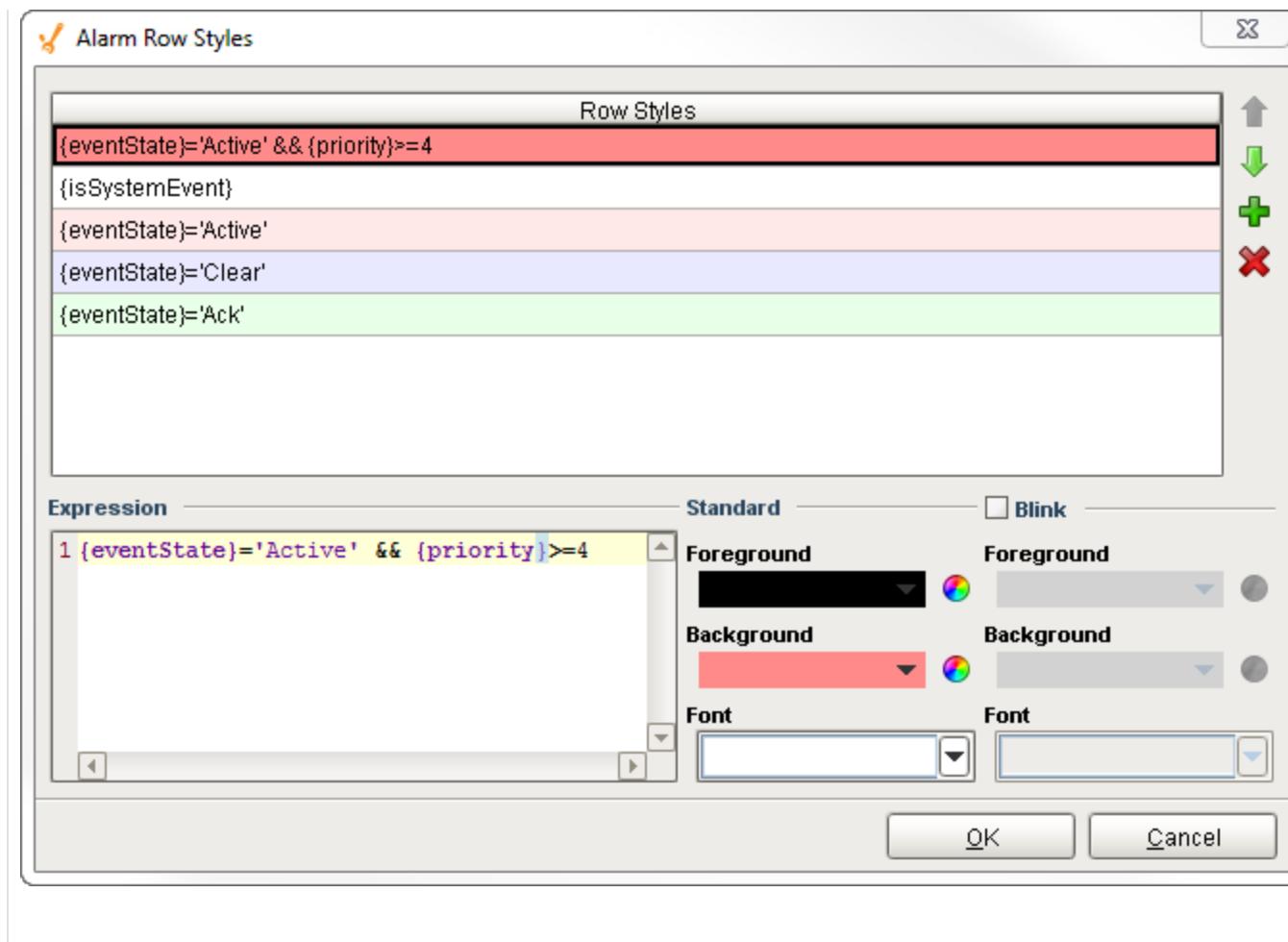
Examples

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 Journal Table - Alarm Row Styles



Vision - Alarm Journal Table

General

Event Time	Display Path	Event State	Priority	Event Value	Current State
6/12/19, 12:51 PM	Motor Plant/Motor 2/Amps/Low Amps	Ack	Critical		Cleared, Ackno...
6/12/19, 12:51 PM	Motor Plant/Motor 2/Amps/Low Amps	Clear	Critical	26	Cleared, Unackn...
6/12/19, 12:51 PM	Sine/Sine3/High Setpoint	Clear	High	49.0706	Cleared, Unackn...
6/12/19, 12:51 PM	Motor Plant/Motor 3/Amps/Low Amps	Active	Critical	19	Active, Unackno...
6/12/19, 12:51 PM	Motor Plant/Motor 3/Amps/Low Amps	Ack	Critical		Cleared, Ackno...
6/12/19, 12:51 PM	Motor Plant/Motor 3/Amps/Low Amps	Clear	Critical	26	Cleared, Unackn...
6/12/19, 12:51 PM	Sine/Sine3/High Setpoint	Active	High	51.7299	Active, Unackno...
6/12/19, 12:51 PM	Sine/Sine3/High Setpoint	Ack	High		Cleared, Ackno...
6/12/19, 12:51 PM	Motor Plant/Motor 1/Amps/Low Amps	Clear	Critical	26	Cleared, Unackn...

15,311 events



**INDUCTIVE
UNIVERSITY**

Alarm Journal Table

[Watch the Video](#)

Description

The alarm journal table provides a built-in view to explore alarm history that has been stored in an alarm journal. If you only have one alarm journal specified on your Gateway, then you do not need to specify the journal name. If you have more than one specified, then you need to provide the name of the journal you'd like to query.

The journal table shows the alarm history that is found between the Start Date and End Date properties. When you first put an alarm journal table on a window, these properties will be set to show the most recent few hours of journal history. Note that without further configuration, the journal table will always show the few hours before it was created. To properly configure an alarm journal table, bind its start and end date properties to something 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 acked events.	boolean	.includeAckedEvents	Filters
Active Events	Show active events.	boolean	.includeActiveEvents	Filters
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. Note: The border is unaffected by rotation.	Border	.border	Common
Cleared Events	Show cleared events.	boolean	.includeClearedEvents	Filters
Date Format	A date format pattern used to format dates in the table. If blank, the default format for the locale is used.	String	.dateFormat	Appearance
Display Path	Filter alarms by alarm display path, falling back to the source path if display path isn't set. Specify multiple paths by separating them with commas. Supports the wildcard "**".	String	.displayPath	Filters

Filter			Filter	
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
End Date	The ending date for the displayed history range. If left blank, will default to the current time when the component was loaded.	Date	.endDate	Behavior
Is Filtered	True if the results are filtered. (Read-only)	boolean	.isFiltered	Behavior
Journal Name	The name of the alarm journal to query.	String	.journalName	Behavior
Max Priority	The maximum priority to display.	int	.maximumPriority	Filters
Min Priority	The minimum priority to display.	int	.minimumPriority	Filters
Name	The name of this component.	String	.name	Common
Notes Area Border	The border surrounding the notes area.	Border	.notesAreaBorder	Appearance
Notes Area Font	The font for the notes area.	Font	.notesAreaFont	Appearance
Notes Area Location	The location of the notes display area.	int	.notesAreaLocation	Appearance
Notes Area Size	The size of the notes area, in pixels.	int	.notesAreaSize	Appearance
Number Format	A number format string to control the format of the value column.	String	.numberFormat	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Read Timeout	The timeout, in milliseconds, for running the alarm history query.	int	.readTimeout	Behavior
Row Height	The height, in pixels, for each row of the table.	int	.rowHeight	Appearance
Row Styles	A dataset containing the different styles configured for different alarm states.	Dataset	.rowStyles	Appearance
Search String	Filter alarms by searching for a string in both source path and display path.	String	.searchString	Filters
Selected Alarms	A dataset containing each selected alarm. (Read-only)	Dataset	.selectedAlarms	Data
Selection Color	The color of the selection border. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector .	Color	.selectionColor	Appearance
Selection Thickness	The size of the selection border.	int	.selectionThickness	Appearance
Show Table Header	Toggles visibility of the table's header.	boolean	.showTableHeader	Appearance
Source Filter	Filter alarms by alarm source path. Specify multiple paths by separating them with commas. Supports the wildcard "*".	String	.sourceFilter	Filters
Start Date	The starting date for the displayed history range. If left blank, will default to 8 hours prior to when the component was loaded.	Date	.startDate	Behavior
System Events	Show system events such as startup and shutdown.	boolean	.includeSystemEvents	Filters
Table Background	The background of the alarm table. See Color Selector .	Color	.tableBackground	Appearance
Table Font	The font for the Alarm Journal's rows.	Font	.font	Appearance
Touchscr	Controls when this input component responds if touchscreen mode is enabled.	int	.	Behavior

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

Scripting

Scripting Functions

- Description

This specialized print function will paginate the table onto multiple pages. This function accepts keyword-style invocation.

- Keyword Args

boolean fitWidth - If true, the table's width will be stretched to fit across one page's width. Rows will still paginate normally. If false, the table will paginate columns onto extra pages. (default = true) [optional]

String headerFormat - A string to use as the table's page header. The substring "{0}" will be replaced with the current page number. (default = None) [optional]

String footerFormat - A string to use as the table's page footer. The substring "{0}" will be replaced with the current page number. (default = "Page {0}") [optional]

boolean showDialog - Whether or not the print dialog should be shown to the user. Default is true. [optional]

boolean landscape - Used to specify portrait (0) or landscape (1) mode. Default is portrait (0). [optional]

- Return

boolean - True if the print job was successful.

- Scope

Client

- Description

Returns a dataset of the alarms currently displayed in the Alarm Journal Table component. The columns will be: EventId, Source, DisplayPath, EventTime, State, Priority and IsSystemEvent

- Keyword Args

None

- Return

Dataset - A dataset of alarms.

- Scope

Client

Extension Functions

- Description

Returns a popup menu that will be displayed when the user triggers a popup menu (right click) in the table. Use [system.gui.createPopupMenu\(\)](#) to create the popup menu.

- Parameters

Component self - A reference to the component that is invoking this function.

List selectedAlarmEvents - The alarm events selected on the Alarm Status Table. For an individual alarmEvent, call alarmEvent.get('propertyName') to inspect. Common properties: 'name', 'source', 'priority'.

- Return

JPopupMenu - A popup menu that was created with `system.gui.createPopupMenu()`
- Scope

Client
- 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.getPropertyName()` to inspect. Common properties: 'name', 'source', 'priority'.
- Return

Boolean
- Scope

Client
- Description

Called when an alarm is double-clicked on to provide custom functionality. Does not return a value.
- Parameters

Component self - A reference to the component that is invoking this function.
Alarm Event alarmEvent - The alarm event itself. Call `alarmEvent.getPropertyName()` to inspect. Common properties: 'name', 'source', 'priority'.
- Return

Nothing
- Scope

Client

Event Handlers

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

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

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

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

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

.source	The component that fired this event.
.key	The key code for this event. Used with the <code>keyPressed</code> and <code>keyReleased</code> events. See below for the key code constants.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Customizers

The Alarm Row Styles Customizer manages the way the Alarm Journal renders each alarm.

- [Vision Component Customizers](#)

Examples

There are no examples associated with this component.

Vision - Containers Palette

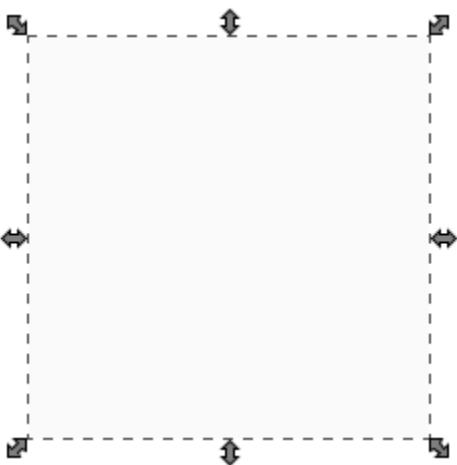
Container Components

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

[In This Section ...](#)

Vision - Container

General



The diagram shows a rectangular container component with a dashed border. Inside, there are several small black squares representing resize handles at the corners and midpoints of the sides. The interior of the container is empty.

Component Palette Icon:



A small icon representing a container, consisting of a square with a horizontal line through it.

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.

Tip: To move a container around on a window, you need to hold the alt key while clicking and dragging.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector .	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border.	Border	.border	Common

	Note: The border is unaffected by rotation.			
CombineRepaints	Set this to true for containers with many sub-components that need to redraw frequently (flashing, rotating, animating).	boolean	.combineRepaints	Behavior
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Font	Font of text on this component.	Font	.font	Appearance
MouseoverText	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Opaque	If false, backgrounds are not drawn. If true, backgrounds are drawn.	boolean	.opaque	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Texture	Background texture image for this container.	String	.texturePath	Appearance
TileOptimized	If true, this container's children should never overlap, and you'll get better painting performance.	boolean	.optimizedDrawingEnabled	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
DataQuality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

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

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

wn	
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

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

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

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

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

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

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

.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.
------------	---

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

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

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

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

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

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

.shiftDown

True (1) if the Shift key was held down during this event, false (0) otherwise.

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

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

Customizers

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

Examples

Customized Container with Border



Property Name	Value
Border	Bevel (Double)
Background Color	255,232,204

Vision - Template Repeater

General



Component Palette Icon:



INDUCTIVE
UNIVERSITY

Template Repeater

[Watch the Video](#)

Description

The Template Repeater repeats instances of templates any number of times. It can arrange them vertically, horizontally, or in a "flow" layout, which can either be top-to-bottom or left-to-right. If there are too many to fit, a scrollbar will be shown. This makes it easy to quickly create screens that represent many similar pieces of equipment. It also can be used to create screens that are dynamic, and automatically configure themselves based on configuration stored in a database or tag structure. When first dropped on a window, the template repeater will look like any other empty container. To select the template to repeat, configure the repeater's Template Path property. There are two ways to set how many times the template should repeat:

- Count - The template will be repeated X times, where X is the value of "Repeat Count". The repeat count starts at zero and increments X amount of times. Each value for X will be inserted into the custom property of the template that will be repeated. Template repeater inserts the value of X into the custom property on the template with the same name as the template repeater's "Index Property Name." For example, if the template has a custom property of "index" and the template repeater's Index Property Name is also "index," then the template will be repeated X many time with the value of X being inserted into the template's custom property called "index."
- Dataset - The template will be repeated once for each row in the "Template Parameters" dataset. The template's custom properties with the same names as the dataset's column names will assume the values of each row of the dataset.



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

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

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector .	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. <div style="border: 1px solid #ccc; padding: 5px; width: fit-content;">Note: The border is unaffected by rotation.</div>	Border	.border	Common
Flow Alignment	Alignment for "Flow" layout style.	int	.flowAlignment	Appearance
Flow Direction	When the layout style is flow, this property controls if the components in the container flow horizontally or vertically.	int	.flowDirection	Appearance
Horizontal	The gap size to use for horizontal gaps.	int	.	Appearance

Gap			horizontalGap	
Index Parameter Name	A name of an integer parameter on the template that will be set to an index number.	String	.indexParamName	Behavior
Layout Style	Controls how the repeated template instances are laid out inside the repeater.	int	.layoutStyle	Appearance
Marquee Mode	Turn the repeater into a scrolling marquee.	boolean	.marqueeMode	Behavior
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Repeat Behavior	"Count" will repeat the template a number of times, assigning each template an index number. "Dataset" will repeat the template once per row in the template parameter's dataset.	int	.repeatBehavior	Behavior
Repeat Count	The template will be repeated this many times, if the repeat behavior is set to "Count."	int	.repeatCount	Behavior
Scroll Delay	The time (in milliseconds) to wait between performing each step in a scroll.	int	.scrollDelay	Behavior
Stay Delay	The time (in milliseconds) to wait between scrolls.	int	.stayDelay	Behavior
Template Parameters	This dataset will be used to control the number of templates and the parameters set on the templates if the repeat behavior is set to "Dataset."	Dataset	.templateParams	Behavior
Template Path	The path to the template that this container will repeat.	String	.templatePath	Behavior
Vertical Gap	The gap size to use for vertical gaps.	int	.verticalGap	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

- Description

Returns a list of templates loaded into the Template Repeater. Properties on the components within each instance can be references by calling getComponent().
- Parameters

None
- Return

[List of Templates](#)
- Scope

Client

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

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

Customizers

- [Vision Component Customizers](#)

Examples

Code Snippet: getLoadedTemplates()

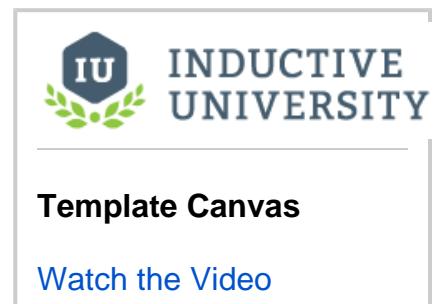
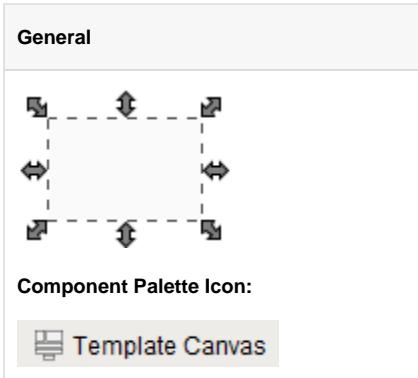
```
#This script will call getLoadedTemplates() on a Template Repeater, and
#then print the text property of a Label component in each instance

#Store a reference to the Template Repeater component in a variable
repeater = event.source.parent.getComponent('Template Repeater')

#Store the list of templates in another variable
templateList = repeater.getLoadedTemplates()

#Iterate through the list
for template in templateList:
    #find a component named "Label" in the instance,
    #and print the value of the text property
    print template.getComponent('Label').text
```

Vision - Template Canvas



Description

The template canvas is similar to the template repeater but allows for more control of the templates than the template repeater.

The Templates property on the template canvas is a dataset. Each row in this dataset represents a manifestation of a template. It can be the same template or a different template on each row. This dataset allows for control over the size, position and layout of the template. There are two methods of controlling the layout of each template inside the template canvas:

- **Absolute Positioning** - The location of the template is explicitly managed through the "X" and "Y" columns of the Templates property's dataset. Consequently the columns labeled Width and Height control the size of the template.
- **Layout Positioning** - The template canvas uses "MiG Layout" to manage the location of the template. MiG Layout is a common albeit complicated layout methodology. It supports layouts that wrap the templates automatically as well as docking the template to one side of the template canvas. You can learn more about MiG Layout at <http://www.miglayout.com>

In addition, control over data inside each template can be achieved by adding a column with the name Parameters to the dataset and populating this column with dictionary style key words and definitions.

Additional templates can be added to the template canvas by inserting an additional row to the Templates property's dataset. The same applies to removing the templates but with removing the rows from the dataset.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector .	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">Note: The border is unaffected by rotation.</div>	Border	.border	Common
Layout Constraints	The overall layout constraints for the canvas.	String	.layoutConstraints	Behavior
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Scroll Behavior	Controls which direction(s) the canvas will scroll in.	int	.scrollBehavior	Behavior
Show Loading	If false, the loading indicator will never be shown.	boolean	.showLoading	Appearance
Templates	A dataset containing a row per template to instantiate.	Dataset	.templates	Data
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties					
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated	

Scripting

Scripting Functions

- Description

Returns a list of the templates that comprise the template canvas.
- Parameters

Nothing
- Return

List - A list of VisionTemplate definitions. Each instance in the canvas will return its definition's name. The names of each instance can be accessed with `getInstanceName()`. Individual components in each instance can be accessed with `getComponent()`.
- Scope

Client
- 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 accessed by calling `.propertyName`
- Scope

Client

Extension Functions

- Description

This will be called once per template that is loaded. This is a good chance to do any custom initialization or setting parameters on the template.
- Parameters

Component `self`- A reference to the component that is invoking this function.
Vision Template `template` - The template. The name of the template in the dataset will be available as `template.instanceName`
- Return

Nothing
- Scope

Client

Event Handlers

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

.source	The component that fired this event
---------	-------------------------------------

.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Not all components include an accurate oldValue in their events.
.propertyName	<p>The name of the property that changed.</p> <p>Note: Remember to always filter out these events for the property that you are looking for! Components often have many properties that change.</p>

Customizers

This component has its own customizer called the Template Canvas Customizer. The Template Canvas Customizer allows you to create multiple instances of a template. Here is where you can configure some of the properties of the template instance that are inside the Template Canvas. To edit a template instance, select it from the Instances list. To cancel your edit and add a new instance instead, click the Cancel button in the bottom left.

Templates Property

The "Templates" property, in the Property Editor, stores all the data that is entered into the customizer. New template instances can be created directly on the "Templates" property as well. To edit or view the dataset, click the Dataset Viewer next to the "Templates" property.

Template Canvas Customizer - Property Description

Property	Description
Instances	A list of the templates currently in the Template Canvas.
Add/Edit Instances	Section of the Template Canvas Customizer where you add new instances and edit existing instances. Select an instance from above to edit that instance.
Name	Name of the selected template instance.
Z-Index	The index position along the Z axis that should be used for the instance. If left empty, then Z order will be determined by the row index position of the instance as it sits in the Template Canvas' Templates property.
Template	The template path for the selected template instance.
Absolute Positioning	Sets the position and size of the components inside the template. In order from left to right, the four boxes are X, Y, Width, and Height.
Layout Positioning	Uses MiGLayout to manage template location. It allows you to easily determine the layout of components or templates within a container (i.e., "span,wrap"). To learn more, go to http://www.miglayout.com
Parameters	Shows a list of all the parameters that are defined in the selected template. Specify the values for each template parameter. To make this dynamic, you must bind the Templates property of the Template Canvas.

More information on the Template Canvas Customizer can be found on the [Component Customizers](#) page.

Data Types and the Parameters Field

The "Parameters" field in the customizer accepts string values, but attempts to convert the value if the underlying template parameter is set to a non-string type. In some cases this may require special formatting on the supplied string. The table below provides some examples.

Data Type	Expected Format	Format Examples
Color	Colors may be entered in as either a name, or an RBG string	red 0,0,255
Date	Date objects may be entered as either a UNIX timestamp in milliseconds, or in the following notation. In all cases, quotation marks should not be added.	1591374627000

```

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

```

2020-03-28 06:
38:00:000

Examples

Code Snippet

```

#This example demonstrates how to pull value information from templates that are inside the template
canvas.
#This example assumes that each template has a custom property called ContentValue

#Get all the template instances of the canvas.
templates = event.source.parent.getComponent('Template Canvas').getAllTemplates()

#The templates are a list therefore you can iterate through them.
for template in templates:

    #You can access the properties of the template. This example prints the ContentValue custom
    property to the console.
    print template.ContentValue

```

Code Snippet - Search by Name

```

#This example demonstrates how to iterate through each template in a template canvas
#looking for a named instance. Once found, print the value of a property on a component in
#that instance.

#This assumes that the canvas contains a template instance named "timerTemplate" and
#a Timer component (named Timer) is inside the instance.

#Create a reference to the Template Canvas
canvas = event.source.parent.getComponent('Template Canvas')

#Retrieve all template instances in the canvas
tempInstance = canvas.getAllTemplates()

#Iterate through each template instance
for template in tempInstance:

    #Compare the name of each instance.
    if template.getInstanceName() == "timerTemplate":

        #Print the Value property on the Timer component inside the template
        print template.getComponent("Timer").value

```

Code Snippet - Read User Input Example

```

#This script will retrieve a list of all templates in a template canvas, and record user input.

#The code is designed to work with the a User Input example,
#but can be easily modified to work with different templates.

#Reference the template canvas component, and call the getAllTemplates() method.
#This will return a list of every instance in the canvas

```

```
templateList = event.source.parent.getComponent('Template Canvas').getAllTemplates()

#Initialize a list. User input from each text field will be stored in this variable
userInput = []

#Iterate through each template instance inside the canvas
for template in templateList:

    #add the user inputted value to the userInput list. The values are originally returned in Unicode.
    #the Python str() function is casting the Unicode values as string values.
    userInput.append(str(template.TextField_Text))

>Show the values in a messageBox. This could be replaced with an INSERT query, or some other action.
#str() is used again to case the list as a string. This only required to work with the messageBox
function
#since the function requires a string argument be passed in
system.gui.messageBox(str(userInput))
```

Related Topics ...

- [Vision Component Customizers](#)

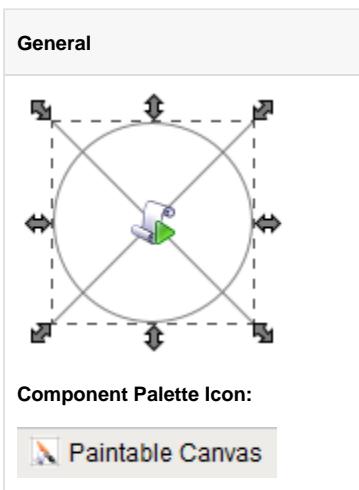
Vision - Misc Palette

Misc Components

The following components give you various ways to create or animate displays.

[In This Section ...](#)

Vision - Paintable Canvas



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. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector .	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. Note: The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Focusable	If the component is focusable, it will receive keyboard input and can detect if it is the focus owner.	boolean	.focusable	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground	The foreground color of the component. See Color Selector .	Color	.foreground	Appearance

nd Color				
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

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

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

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

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

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

controlDown	
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

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

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

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

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

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

.source	The component that fired this event
---------	-------------------------------------

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

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

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

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

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

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

.source	The component that fired this event

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

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

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

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

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

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

.source	The component that fired this event
.button	The code for the button that caused this event to fire.

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

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

.source	The component that fired this event
.graphics	An instance of java.awt.Graphics2D that can be used to paint this component. The point (0,0) is located at the upper left of the component.
.width	The width of the paintable area of the component. This takes into account the component's border.
.height	The height of the paintable area of the component. This takes into account the component's border.

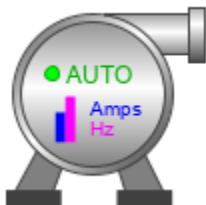
Customizers

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

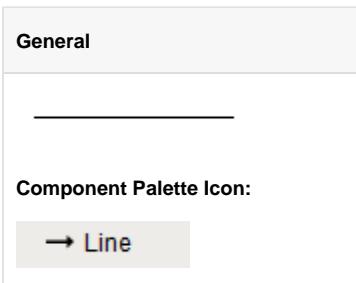
Examples

There are no examples associated with this component. However examples are available in the component itself.

The component comes prescripted to render the following pump:



Vision - Line



Component Palette Icon:

→ Line

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!

Note: If you are looking for the Line component used in Reporting, refer to [Report - Line Shape](#).

Properties

Name	Description	Property Type	Scripting	Category
Color	Set the color of the line. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector .	Color	.foreground	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Dash Pattern	Enter a string of comma-delimited numbers which indicate the stroke pattern for a dashed line. For instance, "3,5" means three pixels on, five pixels off.	String	.strokePattern	Appearance
Left Arrow	Draw an arrow head on the left/top of the line?	boolean	.leftArrow	Appearance
Left Arrow Size	The size of the left arrow, if present.	int	.leftArrowSize	Appearance
Line Mode	The line mode determines where in the rectangle the line is drawn.	int	.lineMode	Appearance
Line Style	The line style determines how the shape of the line looks. Options are: Plane, Dashed, Sinusoidal, Sinusoidal-Dashed, Loop, and Loop-Dashed.	int	.lineStyle	Appearance
Line Width	Set the width of the line in pixels.	int	.lineWidth	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Right Arrow	Draw an arrow head on the right/bottom of the line?	boolean	.rightArrow	Appearance
Right Arrow Size	The size of the right arrow, if present.	int	.rightArrowSize	Appearance
Sine Height	Sets the amplitude of the sine wave to be drawn.	int	.sineHeight	Appearance
Sine Length	Sets the wavelength of the sine wave to be drawn.	int	.sineLength	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

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

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

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

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

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

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

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

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

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

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

.source	The component that fired this event
---------	-------------------------------------

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

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

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

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

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

Customizers

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

Examples

Line with Sinusoidal Pattern



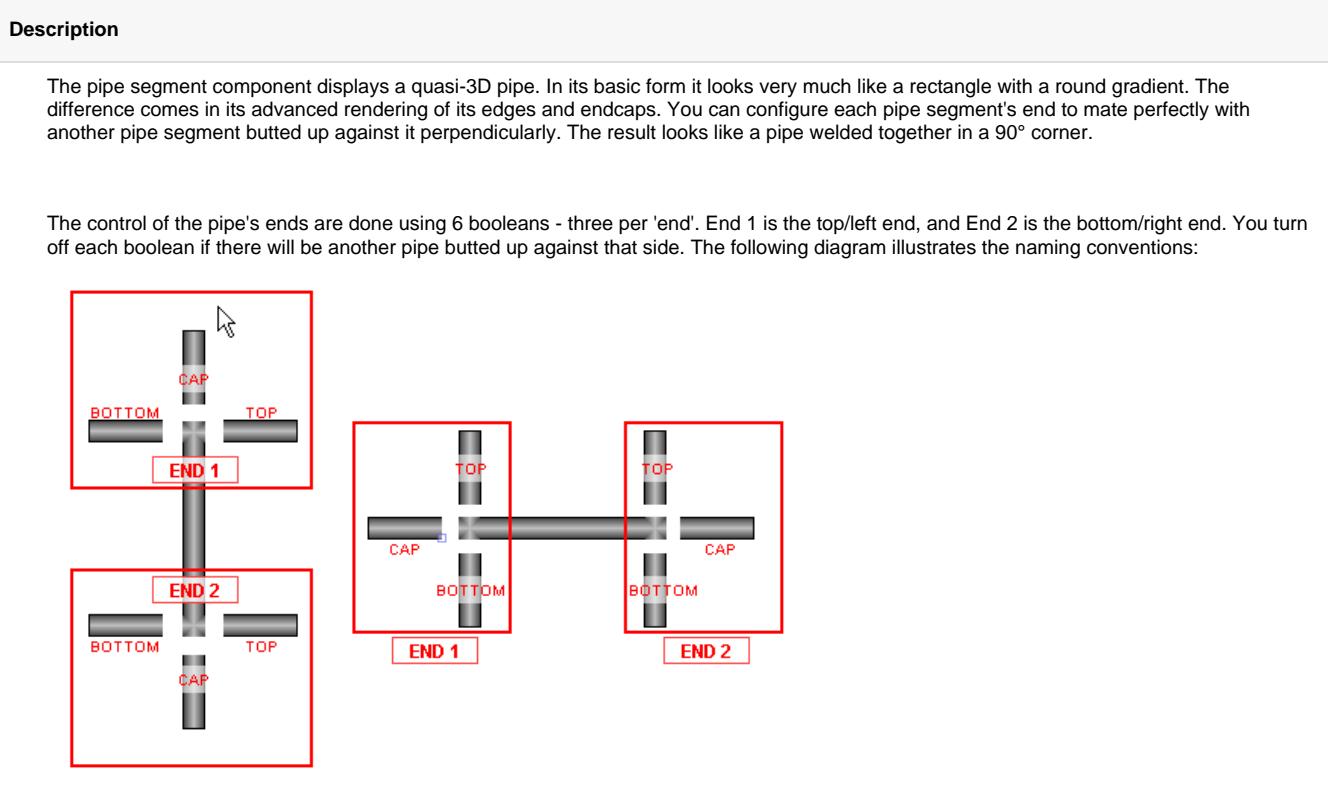
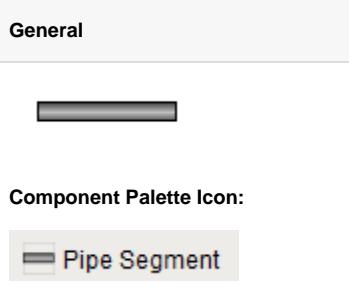
Property	Setting
Color	0,0,255
Line Style	Sinusoidal

Line with Arrow Endpoints



Property	Setting
Color	217,0,0
Line Style	Plain
Left Arrow	True
Left Arrow Size	25
Line Width	4
Right Arrow	True
Right Arrow Size	25

Vision - Pipe Segment



Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. Note: The border is unaffected by rotation.	Border	.border	Common
Center Fill	The center of the fill gradient. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector .	Color	.mainColor	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Edge Fill	The edge of the fill gradient. See Color Selector .	Color	.secondaryC	Appearance

			olor	
End 1 Bottom?	Draw the border at end #1's bottom?	boolean	.end1Bottom	Appearance
End 1 Cap?	Draw the border at end #1's cap?	boolean	.end1Cap	Appearance
End 1 Top?	Draw the border at end #1's top?	boolean	.end1Top	Appearance
End 2 Bottom?	Draw the border at end #2's bottom?	boolean	.end2Bottom	Appearance
End 2 Cap?	Draw the border at end #2's cap?	boolean	.end2Cap	Appearance
End 2 Top?	Draw the border at end #2's top?	boolean	.end2Top	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Outline Color	The color of the outline border. See Color Selector .	Color	.outlineColor	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

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

shiftDown

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

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

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

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

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

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

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

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

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

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

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

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

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

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

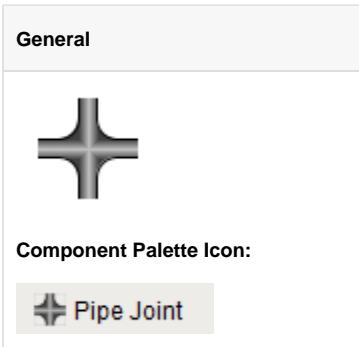
Customizers

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

Examples

There are no examples associated with this component.

Vision - Pipe Joint



Description
The pipe joint displays a fancy joint component two join two pipe segments together. By turning off the cardinal directions, this will display a two-, three-, or four-pipe union. This component is optional, as pipes can butt up against each other and look joined.

Properties				
Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. Note: The border is unaffected by rotation.	Border	.border	Common
Bottom?	Indicates if the joint has an outlet at the bottom.	boolean	.bottom	Appearance
Center Fill	The center of the fill gradient. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector .	Color	.mainColor	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Edge Fill	The edge of the fill gradient. See Color Selector .	Color	.secondaryColor	Appearance
Left?	Indicates if the 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. See Color Selector .	String	.name	Common
Outline Color	The color of the outline border. See Color Selector .	Color	.outlineColor	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Right?	Indicates if the joint has an outlet at the right.	boolean	.right	Appearance
Styles	Contains the component's styles	Dataset	.styles	Appearance
Top?	Indicated if that joint has an outlet at the top.	boolean	.top	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Deprecated Properties				
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

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

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

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

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

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

.source	The component that fired this event

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

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

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

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

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

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

.source	The component that fired this event
.button	The code for the button that caused this event to fire.

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

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

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

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

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

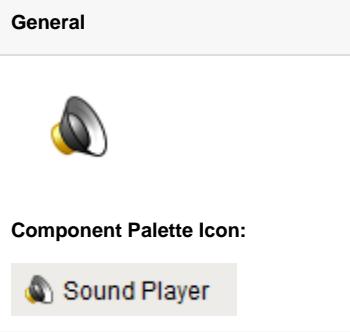
Customizers

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

Examples

There are no examples associated with this component.

Vision - Sound Player



Component Palette Icon:

Sound Player

Description

The Sound Player component is an invisible component that facilitates audio playback in the client. Each Sound Player component has one sound clip associated with it, and will play that clip on demand. There is a built in triggering system, as well as facilities to loop the sound while the trigger is set. The sound clip needs to be a *.wav file. The clip becomes embedded within the window that the sound player is on. Clients do not need access to a shared *.wav file.

Properties

Name	Description	Property Type	Scripting	Category
Loop Count	If Loop Mode is "Loop N Times", this is the "N".	int	.loopCount	Behavior
Loop Mode	The Loop Mode determines how many times the sound is played when triggered.	int	.loopMode	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Mute	If true, the clip will be muted during playback.	boolean	.mute	Behavior
Name	The name of this component.	String	.name	Common
Play Mode	The Play Mode determines whether the sound is played automatically on trigger or manually.	int	.playMode	Behavior
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Sound Data	The clip that this component will play.	byte[]	.soundData	Data
Trigger	The clip will be played when the trigger is true, if Play Mode is "ON_TRIGGER"	boolean	.trigger	Data
Volume	The volume to use for playback (from 0.0 to 1.0).	double	.volume	Behavior

Deprecated Properties

Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated
--------------	---	-----	--------------	------------

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

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

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

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

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

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

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

.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.
------------	---

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

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

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

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

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

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

.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.
------------	---

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

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

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

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

Customizers

- [Vision Component Customizers](#)

Examples

There are no examples associated with this component.

Vision - Timer

General



Component Palette Icon:



Description

The timer button is an invisible button that can be used to create repeated events in a window. This is often used for animations or repetitive scripts within a window. When running, the timer's Value property is incremented by the Step By value, until the value 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 $((value + step) \bmod \text{bound})$	int	.value	Data

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

.source	The component that fired this event
---------	-------------------------------------

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

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

Customizers

This component does not have any custom properties.

Examples

Expression Binding Example

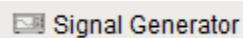
```
//Suppose that you have images that make up frames of animation.
//Name your images: "Frame0.png", "Frame1.png", "Frame2.png". Set the timer's Bound to be 3, then bind
the image path of animate component to the following expression:
"Frame" + {Root Container.Timer.value} + ".png"
```

Vision - Signal Generator

General



Component Palette Icon:



Description

The signal generator is similar to the Timer component, but its value isn't simply a counter. Instead, you can choose from a variety of familiar signals. You configure the frequency by setting the `Period` property, which is in milliseconds. You configure the resolution by setting the `ValuesPerPeriod` property.

For example, if you choose a sine wave signal with a period of 2000 milliseconds and 10 `valuesPerPeriod`, your sine wave will have a frequency of 0.5 Hz, and its value will change 10 times every 2 seconds.

Properties

Name	Description	Property Type	Scripting	Category
Lower Bound	The lower bound of the signal value.	double	.lower	Data
Name	The name of this component.	String	.name	Common
Period	The period of the signal in milliseconds.	int	.period	Behavior
Running?	Determines whether or not the signal is being generated.	boolean	.running	Behavior
Signal Type	The signal type (shape) of the signal value.	int	.signalType	Behavior
Upper Bound	The upper bound of the signal value.	double	.upper	Data
Value	The current value of this signal generator.	double	.value	Data
Values/Period	The number of value changes per period.	int	.valuesPerPeriod	Behavior

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

.source The component that fired this event

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

.source The component that fired this event

.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Not all components include an accurate oldValue in their events.
.propertyName	<p>The name of the property that changed.</p> <p>Note: Remember to always filter out these events for the property that you are looking for. Components often have many properties that change.</p>

Customizers

This component does not have any custom properties.

Examples

This component does not have any examples associated with it.

Vision - Reporting Palette

Reporting Components

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

[In This Section ...](#)

Vision - Report Viewer

General

Downtime Minutes Report

Run	Occurrences	Minutes
Run 05/05 01:58 PM, Line 1	1	0.25
Run 05/05 01:58 PM, Line 1	2	0.48
Run 05/05 01:58 PM, Line 1	2	1.27
Run 05/05 01:58 PM, Line 1	2	0.95
Run 05/05 01:58 PM, Line 1	1	0.5
Run 05/05 01:58 PM, Line 1	1	0.25
Run 05/05 01:58 PM, Line 1	2	0.78
Run 05/05 01:58 PM, Line 1	1	0.23
Run 05/05 01:58 PM, Line 1	1	0.48
Run 05/05 01:58 PM, Line 1	1	0.27
Run 05/05 01:58 PM, Line 1	3	0.77
Run 05/05 01:58 PM, Line 1	1	0.27
Run 05/05 01:58 PM, Line 1	1	0.25
Run 05/05 01:58 PM, Line 1	3	1.72
Run 05/05 01:58 PM, Line 1	3	1.27
Run 05/05 01:58 PM, Line 1	1	0.23
Run 05/05 01:58 PM, Line 1	2	0.73
Run 05/05 01:58 PM, Line 1	3	1.03
Run 05/05 01:58 PM, Line 1	2	1.3
Run 05/05 01:58 PM, Line 1	3	1.43
Run 05/05 01:58 PM, Line 1	1	0.48
Run 05/05 01:58 PM, Line 1	2	0.75
Run 05/05 01:58 PM, Line 1	1	0.52
Run 05/05 01:58 PM, Line 1	1	0.25
Run 05/05 01:58 PM, Line 1	1	0.22
Run 05/05 01:58 PM, Line 1	1	0.25

< > 100% ▾

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. To find documentation on the deprecated Report Viewer prior to Ignition 7.8, see the [Legacy Report Viewer](#) documentation.

Properties

Name	Description	Type	Scripting	Category
Background Color	Color that lays underneath the report. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector .	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border.	Border	.border	Common
<p>Note: The border is unaffected by rotation.</p>				
Current Page	Current page in the report you would like to view.	Int	.currentPage	Data

EndDate	End date for the report.	Date	.EndDate	Report Parameters
Fit Panel	Ignore the zoom and fit the report to the component.	Boolean	.fitPanel	Data
Foreground Color	The foreground color the labels on the component. See Color Selector .	Color	.foreground	Appearance
Name	The name of this component.	String	.name	Common
Page Count	Number of pages in the report.	Int	.pageCount	Data
Report Loading	Returns true while the report is loading. Note: 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
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 controls.	Boolean	.showControls	Appearance
StartDate	Start date for the report.	Date	.StartDate	Report Parameters
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

Scripting

Scripting Functions

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

Nothing

- Scope

Client

- 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

- 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

- 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

- 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

- 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

- 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

Event Handlers

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

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

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

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

.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

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

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

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

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

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

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

.controlDown	True (1) if the Control key was held down during this event, false (0) otherwise.
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

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

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

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

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

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

.source	The component that fired this event
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. 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.

Examples

print()

```
#calls print on a Report Viewer component located in the same window  
  
reportViewer = event.source.parent.getComponent('Report Viewer')  
reportViewer.print()
```

print() with default printer, no dialog

```
#calls print on a Report Viewer component located in the same window  
#bypasses the print dialog window and uses the default printer  
  
reportViewer = event.source.parent.getComponent('Report Viewer')  
reportViewer.print(None, False)
```

saveAsPDF()

```
#Saves the file as a PDF to a user selected location.  
#The file selection window defaults to a name of "Daily Report"  
  
reportViewer = event.source.parent.getComponent('Report Viewer')  
reportViewer.saveAsPDF("Daily Report")
```

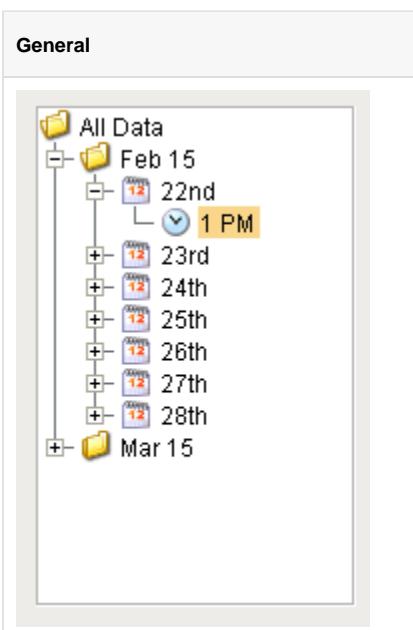
Utilizing reportLoading

```
#This example will change the value of a parameter, wait for a new version of the report  
#to generate, and then print the new report.  
#It is assumed that this script is being called from a component-based script, and the source component  
is in the same container as the Report Viewer  
  
#Reference the report viewer  
reportViewer = event.source.parent.getComponent('Report Viewer')  
  
#Make a change to one of the Report Parameters on the Report Viewer.  
#This will cause a new report to generate.  
reportViewer.EndDate = system.date.now()  
  
#We'll call this function once the report has finished loading.  
def finished():  
    #We're simply printing the report here, but you could place any amount of work  
    #here that you want to execute once the new report has been generated  
    reportViewer.print()  
  
#We'll use this function to constantly check to see if the report has finished loading  
def waiting():  
  
    #While we're waiting....  
    while reportViewer.reportLoading:  
        #...don't do anything, just keep checking  
        pass  
  
    #Once we get to this point, the report has finished loading, so lets call finished()  
    #Using invokeLater so we can finish up in the Event Dispatch Thread  
    system.util.invokeLater(finished)  
  
#Start the waiting process in an asynchronous thread.  
system.util.invokeAsynchronous(waiting)
```

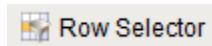
Customizers

This component does not have any custom properties.

Vision - Row Selector



Component Palette Icon:



Row Selector

[Watch the Video](#)

Description

The row selector is a component that acts like a visual filter for datasets. It takes one dataset, chops it up into various ranges based on its configuration, and lets the user choose the splices. Then it creates a virtual dataset that only contains the rows that match the selected splices.

The most common way to splice the data is time. You could feed the row selector an input dataset that represents a large time range, and have it break it up by Month, Day, and then Shift, for example. Then you could power a report with the output dataset, and that would let the user dynamically create reports for any time range via an intuitive interface.

To configure the row selector, first set up the appropriate bindings for its input dataset. Then use its Customizer to alter the levels that it uses to break up the data. In the customizer, add various filters that act upon columns in the input dataset, sorting them by various criteria. For example, you could choose a date column, and have it break that up by quarter. Then below that, you could have it use a discrete filter on a product code. This would let the user choose quarterly results for each product. Each level of filter you create in the customizer becomes a level in the selection hierarchy. Note that the output data is completely unchanged other than the fact that rows that don't match the current user selection aren't present.

This component is very handy for driving the Report Viewer, Table, and Classic Chart components, among others.

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

Properties

Name	Description	Property Type	Scripting	Category
All Data Node Text	Text for the All Data node, if it is displayed.	String	.allDataNodeText	Appearance
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector .	Color	.background	Appearance
Border		Border	.border	Common

	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. Note: The border is unaffected by rotation.			
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	Cursor	.cursor	Common
Data In	The input of the row selection tree. The filter tree changes based on this Dataset.	Dataset	.dataIn	Data
Data Out	The output of the row selection tree. Changes based on user selection in the filter tree.	Dataset	.dataOut	Data
Expand All Data Node	If true, the 'All Data' (root) node will be expanded and selected when the user opens this window.	boolean	.expandAllDataNode	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See Color Selector .	Color	.foreground	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.tooltiptext	Common
Name	The name of this component.	String	.name	Common
Opaque	If false, backgrounds are not drawn. If true, backgrounds are drawn.	boolean	.opaque	Common
Selection Background	The background color of the selected node. See Color Selector .	Color	.selectionBackground	Appearance
Show All Data Node	Should the All Data (root) node be shown or hidden?	boolean	.showAllDataNode	Behavior
Show Node Size	If true, the number of rows in each node will be shown.	boolean	.showNodeSize	Behavior
Show Root Handles	Should root-level nodes have collapse handles?	boolean	.showRootHandles	Behavior
Unknown Node Icon	Icon for any Unknown nodes (nodes where the data didn't match the filter).	String	.unknownIconPath	Appearance
Unknown Node Text	Text for any Unknown nodes (nodes where the data didn't match the filter).	String	.unknownNodeText	Appearance
Visible	If disabled, the component will be hidden.	Boolean	.visible	Common

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

.source	The component that fired this event
.button	The code for the button that caused this event to fire.

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

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

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

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

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

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

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.	The number of mouse clicks associated with this event.

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

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

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

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

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

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

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.

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

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

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

Customizers

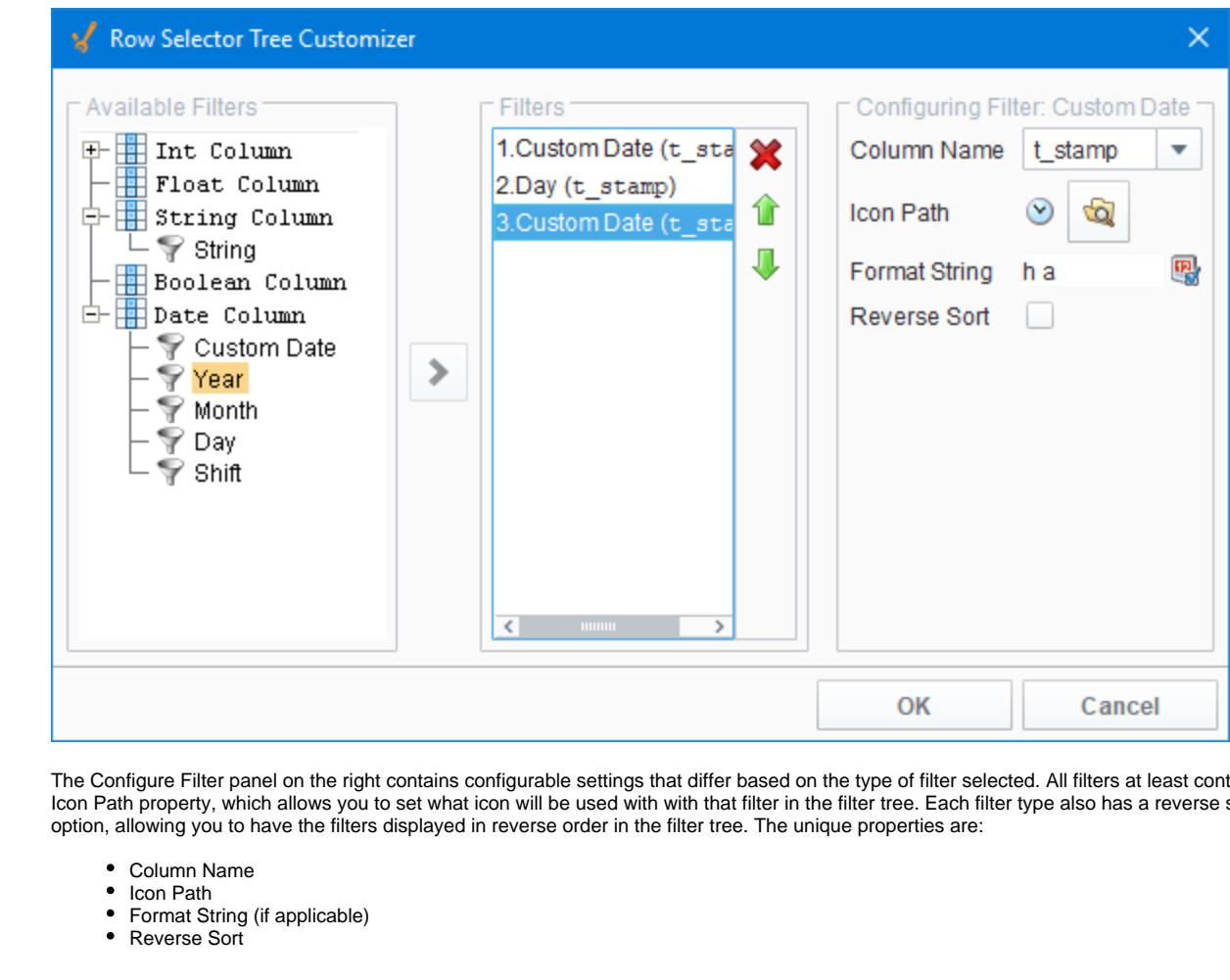
The Row Selector has its own Row Selector Tree Customizer and allows users to customize the row filtering. The customizer provides some default filters which you can use, or customized based on the dataset.

The Row Selector Tree Customizer allows you to build and configure a tree of the data in the input dataset which can then be used to filter it. There are three main parts to the customizer. The left panel contains a list of available filters, the center panel contains a list of filters that will be used, and the right panel will contain configurable properties for the filter currently selected in the center panel.

In the Available Filters section on the left, a list of all of the columns of the dataset are shown. These can be expanded to show the filters available for that column type. Some columns might not have any filters, while others can have many, it just depends on the data type of

column. These filters can then be dragged into the center panel, or highlighted and the Right Arrow  icon pressed to push the filter into the center panel where it becomes an active filter.

The Filters panel in the center contains a list of filters that are being used with each filter being followed by the name of the column that it originated from, and is where you can decide on the order of the filters. The order is important because it is the order in which they will be used in the component. Using the image below as an example, The component will first show a list of years. You can select a particular year, and the output dataset will only contain rows from that year. Alternately, you can expand a year where you will then see a list of strings that are in rows with that year. Selecting one of the strings will display all rows with strings like the one that you selected, that are also in the same year.



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:

- Column Name
- Icon Path
- Format String (if applicable)
- Reverse Sort

Examples

There are no examples associated with this component. Refer to the examples in the [Common Reporting Tasks](#).

Vision - Column Selector

General

Tank Columns

- area
- displayname
- tankcode



INDUCTIVE
UNIVERSITY

Column Selector

[Watch the Video](#)

Component Palette Icon:



Description

The column selector component is conceptually similar to the Row Selector, except that instead of filtering rows, it filters columns from its output dataset. Each column from the input dataset is shown as a checkbox. As the user checks and un-checks columns, the output dataset has those columns added or removed. This is very handy for driving the Table and Classic Chart components. In addition, this component can bring in multiple datasets and output just as many filtered datasets.

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

Properties

Name	Description	Property Type	Scripting	Category
Alphabetize	If true, checkboxes will be ordered alphabetically by their text.	Boolean	.alphabetize	Behavior
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector .	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. Note: The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	Int	.cursorCode	Common
Data In	Input dataset. This is the default when first dropping the component on the window, the name may change based on configuration and there may be more of these input dataset properties.	Dataset	.Data_in	Custom Properties
Data Out	Output dataset. This is the default when first dropping the component on the window, the name may change based on configuration and there may be more of these output dataset properties.	Dataset	.Data_out	Custom Properties

Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See Color Selector .	Color	.foreground	Appearance
Group By Dataset	If true, checkboxes will be grouped by their dataset. Otherwise, checkboxes will be arranged flat.	Boolean	.grouping	Behavior
Horizontal Gap	The horizontal gap between checkboxes or grouping panels.	Int	.hGap	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Normalize Widths	If true, all checkboxes will be assigned the same width, which causes them to line up in columns.	Boolean	.normalizeWidths	Appearance
Vertical Gap	The vertical gap between checkboxes and grouping panels.	Int	.vGap	Appearance
Visible	If disabled, the component will be hidden.	Boolean	.visible	Common

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

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

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

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.	The number of mouse clicks associated with this event.

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

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

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

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

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

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

.source	The component that fired this event
.button	The code for the button that caused this event to fire.
.clickCount	The number of mouse clicks associated with this event.

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

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

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

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

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

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

.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. Note that not all components include an accurate oldValue in their events.

propertyNa
me

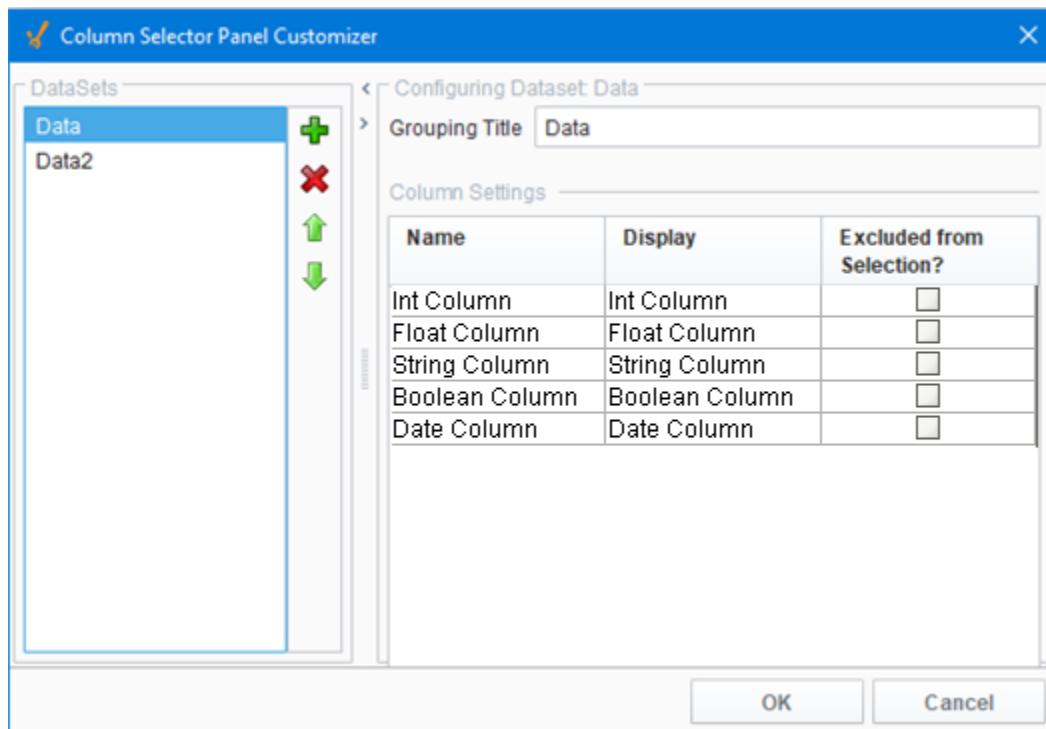
The name of the property that changed.

Note: Remember to always filter out these events for the property that you are looking for. Components often have many properties that change.

Customizers

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

The Column Selector Customizer contains two basic parts. The left side of the customizer allows you to configure how many datasets can be brought in for filtering. Each dataset added will add two additional custom properties to the Column Selector; an In dataset property and an Out filtered dataset property. Datasets can also be removed here, or moved up or down in the list. If there are multiple datasets, the columns from the first dataset in the list will be displayed at the top of the Column Selector, while the columns from the last will be at the bottom.

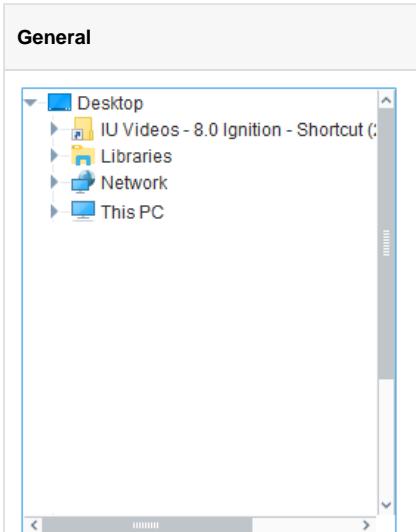


The right side of the customizer allows you to configure the settings for each dataset. When a dataset is highlighted on the left, we can see some basic information about it on the right, such as the Grouping Title and a list of all of the columns in that dataset. The Grouping Title is only used if there is more than one dataset in the Column Selector. In the component, each dataset's columns will be contained in a border and will display the Grouping Title. This can be configured to be anything, so that it is easier for a user to distinguish what each set of columns is for. In the Column Settings table, we see each one of the columns in that dataset listed out. Here, the Display column allows us to alter what name that column will display on the component, again allowing you to create names that are more meaningful to the user. Finally, the Excluded from Selection column allows you to exclude certain columns from being filtered. Columns that have this enabled will not show up in the list of columns on the component. This will not filter them out in the output dataset, but instead forces them to be in the output dataset.

Examples

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

Vision - File Explorer



Component Palette Icon:



File Explorer and PDF Viewer

[Watch the Video](#)

Description

The File Explorer component displays a filesystem tree to the user. It can be rooted at any folder, even network folders. It can also filter the types of files that are displayed by their file extension (i.e., "pdf"). The path to the file that the user selects in the tree is exposed in the bindable property Selected Path.

The File Explorer component is typically used in conjunction with the PDF Viewer component in order to create a PDF viewing window. This is very useful for viewing manuals, documents, or reports from within your project. To use this component to drive a PDF Viewer component, refer to the section on [File Explorer and PDF Viewer](#).

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector .	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. <div style="border: 1px solid #ccc; padding: 5px; margin-top: 10px;">Note: The border is unaffected by rotation.</div>	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Enabled	If disabled, the component can't be used.	boolean	.componentEnabled	Common
File extension filter	Semi-colon separated list of extensions to filter out files, such as pdf or txt. Example "pdf;html;txt" shows pdf, html, and text documents.	String	.fileFilter	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See Color Selector .	Color	.foreground	Appearance
Mouseover	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common

er Text				
Name	The name of this component.	String	.name	Common
Root Directory	A directory to act as the root of the file explorer.	String	.rootDir	Behavior
Selected Path	The selected file or folder's path.	String	.selectedPath	Data
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

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

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

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

wn	
.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.

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

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

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

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

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

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

.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.
------------	---

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

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

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

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

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

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

Customizers

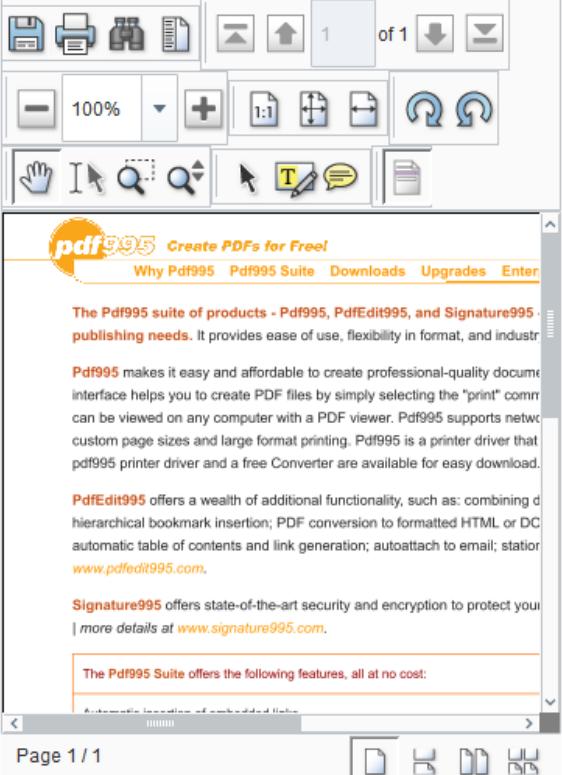
The File Explorer component does not have a customizer.

Examples

Refer to the examples on the [File Explorer](#) and [PDF Viewer](#) pages.

Vision - PDF Viewer

File Explorer and PDF Viewer



The screenshot shows the 'File Explorer and PDF Viewer' component. At the top is a toolbar with icons for saving, printing, zooming, and navigating. Below the toolbar is a file list showing one item ('1 of 1'). The main area contains a PDF viewer window displaying the Pdf995 website. The website header reads 'pdf995 Create PDFs for Free!' with links for 'Why Pdf995', 'Pdf995 Suite', 'Downloads', 'Upgrades', and 'Enter'. The content area discusses the Pdf995 suite of products, mentioning Pdf995, PdfEdit995, and Signature995. It highlights features like ease of use, professional-quality documents, and support for various formats. The PdfEdit995 section mentions combining documents, hierarchical bookmark insertion, and automatic table of contents. The Signature995 section discusses security and encryption. A note at the bottom states 'The Pdf995 Suite offers the following features, all at no cost!'. At the bottom of the PDF viewer window are standard browser navigation buttons.

Component Palette Icon:



**INDUCTIVE
UNIVERSITY**

**File Explorer and
PDF Viewer**

[Watch the Video](#)

Description

The PDF Viewer component displays a PDF that exists as a file in some accessible file system, or as a URL. Note that this component is simply for viewing existing PDFs. To create dynamic reports, or view dynamically generated reports use the [Reporting Module](#).

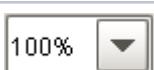
This component is typically used in conjunction with the File Explorer component, in order to create a PDF viewing window. Simply bind the Selected Path property in the PDF Viewer to the File Explorer's *Selected Path* property. See the [File Explorer's documentation](#), as well as the [File Explorer](#)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

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. Note: The border is unaffected by rotation.	Border	.border	Common

File Path	Path to the .pdf file to be displayed.	String	.filePath	Data
Footer Visible	If false, the Footer is not displayed.	Boolean	.footerVisible	Appearance
Name	The name of this component.	String	.name	Common
Page Fit Mode	Mode to fit the document within the viewer. (1 = Disabled, 2 = Actual Size, 3 = Fit Height, 4 = Fit Width)	Integer	.pageFitMode	Appearance
Page View Mode	How to display PDF in Viewer (1 = One Page, 2 = One Column, 3 = Two Page Left, 4 = Two Col Left, 5 = Two Page Right, 6 = Two Col Right)	Integer	.pageViewMode	Appearance
Toolbar Visible	Sets the top PDF control toolbar to visible.	Boolean	.toolBarVisible	Appearance
Utility Visible	Sets the Utility Sidebar to visible.	Boolean	.utilityPaneVisible	Appearance
Visible	If disabled, the component will be hidden.	Boolean	.visible	Common

PDF Viewer Toolbar

Toolbar Buttons	Name	Function
	Save As	Will save the currently loaded pdf to the local computer.
	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: <ul style="list-style-type: none"> 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 Number	Will show the current page number out of the total number of pages, also allowing a page number to be entered which will jump to that page immediately.
	Next Page	Will navigate forward one page of the pdf.
	Last Page	Will navigate forward to the last page of the pdf.
	Zoom Out	Will zoom out from the pdf.
	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.
	Zoom Dynamic Tool	Will zoom in and out using the scroll wheel.
	Select Tool	Can be used to select objects on the pdf such as annotations.
	Highlight Annotation Tool	Can be used to highlight text in the pdf. Can also be done from the Utility Panel and can be configured there as well.
	Text Annotation Tool	Can be used to place a text comment on the pdf. Can be configured in the Utility Panel.
	Show/Hide Form Highlighting	Show or hide highlighting on the form.
	Single Page View Non-Continuous	View the pdf file one page at a time.
	Single Page View Continuous	View the pdf file one page wide with continuous scrolling.
	Facing Page View Non-Continuous	View the pdf file two pages at a time.
	Facing Page View Continuous	View the pdf file two pages wide with continuous scrolling.

Scripting

Scripting Functions

- Description

This function will pass in the bytes of a PDF and load them into the PDF Viewer component. Please see [Storing Files in a Database](#) for more details

- Parameters

`string` bytes - The bytes of the PDF to be displayed on the component

`string` name - The name of the PDF

- Return
 - Nothing
- Scope
 - Client
- Since 7.8.2
- Description

This function will print the PDF.
- Parameters
 - `boolean showDialog`- If true, shows the user a print dialog. Default is true [optional]
- Return
 - Nothing
- Scope
 - Client
- Since 7.8.2
- Description

This function will set the current zoom level of the PDF, adjusted to stay within the minimum / maximum zoom range. Will zoom in on center of page.
- Parameters
 - `float zoom`- Zoom factor to use. 1.0 is no zoom.
- Return
 - Nothing
- Scope
 - Client

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

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

.shiftDown	True (1) if the Shift key was held down during this event, false (0) otherwise.
------------	---

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

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

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

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

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

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

shiftDown

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

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

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

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

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

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

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

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

Customizers

The PDF Viewer component does not have a special customizer, however, it does use the Style Customizer and Custom Properties.

- [Vision Component Customizers](#)

Examples

Refer to the examples on the [File Explorer](#) and [PDF Viewer](#) pages.

Vision - Web Browser Palette

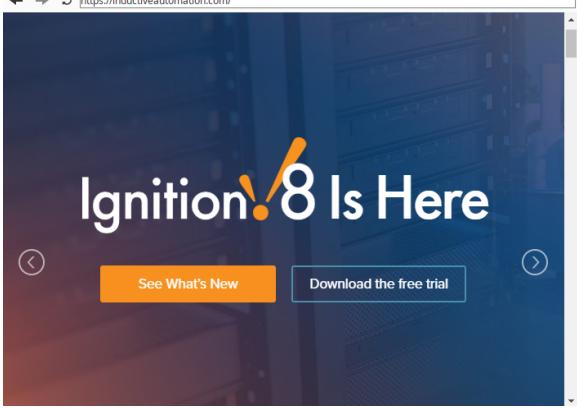
Web Browser Components

The following component gives you the ability to add a web browser to your client.

[In This Section ...](#)

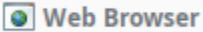
Vision - Web Browser Component

General



The screenshot shows a web browser window displaying the Ignition 8 landing page. The page features a dark blue background with the text "Ignition 8 Is Here" in large white and orange letters. Below the main title are two buttons: "See What's New" and "Download the free trial". The browser interface includes standard navigation buttons (back, forward, search) and a URL bar at the top.

Component Palette Icon:



**INDUCTIVE
UNIVERSITY**

Web Browser

[Watch the Video](#)

Description

The **Web Browser** component in the Designer allows you to embed a full web browser inside of an Ignition Client. This component becomes available in Designer after you download the [Web Browser module](#) from the Inductive Automation's website. The Web Browser module installs the same way as any other modules. Once this component is added onto a window, it will behave just like any other web browser when it is inside a Client.

Client machines need to meet the following minimum requirements to use this component. The component may not work properly if the requirements are not met.

Operating System Requirements

Windows

- Microsoft Windows XP (SP2), 7, 8, Vista, Server 2003 (SP1), Server 2008/2012, 32-bit and 64-bit.
 - Windows version 8 and 8.1 require Java 6 update 38 or greater
 - Oracle (Sun) JRE 1.6.x and higher, 32-bit and 64-bit.

Linux

- Ubuntu 12.04+, Debian 7.7, RedHat Enterprise Linux 7, openSUSE 13.1, Fedora 20, 32-bit and 64-bit
 - Oracle (Sun) JRE 1.6.x and higher, 32-bit and 64-bit.

Required Linux Libraries
Missing Libraries: 32-bit Ubuntu 12.04

Some 32-bit Linux distros are missing a needed library for running the Web Browser: [libXss.so.1](#)

Steps that fixed it in Ubuntu 12.04:

```
sudo apt-get update
sudo apt-get upgrade
sudo apt-get install ia32-libs-multiarch
```

Missing Libraries: Ubuntu 17.04

Ubuntu 17.04 is missing a library that is required for the component to run. Running the following command can resolve the issue:

```
sudo apt-get install libgconf-2-4
```

Mac OS X

- Mac OS X 10.7.x - 10.10.x (Intel)
- Apple or Oracle (Sun) JRE 1.6.x and higher, 32-bit and 64-bit.

Windows

- Microsoft Windows 7, 8, 8.1, 10, Server 2008 R2, Server 2012, Server 2016, 32-bit and 64-bit.
 - Windows version 8 and 8.1 require Java 6 update 38 or greater
- Oracle (Sun) JRE 1.6.x and higher or IBM JRE 1.7.x and higher, 32-bit and 64-bit.

Linux

- Ubuntu 14.04+, 17.04 Desktop, Debian 8+, RedHat Enterprise Linux 7, openSUSE 13.3+, Fedora 24+, 64-bit only
- Oracle (Sun) JRE 1.6.x and higher or IBM JRE 1.7.x and higher, 32-bit and 64-bit.

Required Linux Libraries

Missing Libraries: Ubuntu 17.04

Ubuntu 17.04 is missing a library that is required for the component to run. Running the following command can resolve the issue:

```
sudo apt-get install libgconf-2-4
```

Mac OS X

- Mac OS X 10.9.x - 10.13.x (Intel)
- Apple or Oracle (Sun) JRE 1.6.x and higher, 32-bit and 64-bit.

Note: The Web Browser Component will only support the following audio and video codecs: Opus, Theora, Vorbis, VP8, VP9, and WAV.

The underlying browser component is available in scripting through the `getBrowser()` method. Documentation on the browser component is available at the [JxBrowser Programmer's Guide](#). The Inductive Automation support team is unable to provide detailed advice on scripting with this component. Furthermore, they are unable to provide troubleshooting beyond the basic functionality of the module.

Hardware Notes

ARM

Currently, the jxBrowser does not support the ARM architecture, thus the component will not work properly when used in conjunction with the ARM architecture.

Properties

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Note: The border is unaffected by rotation.	Border	.border	Common

Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
FTP Proxy Port	FTP Proxy Port sets the proxy port for FTP connections. This setting is only used when Use Proxies is checked.	int	.ftpProxyPort	Data
FTP Proxy Server	FTP Proxy Server sets the proxy server for FTP connections. This setting is only used when Use Proxies is checked. Can be empty	String	.ftpProxyServer	Data
HTTP Proxy Port	HTTP Proxy Port sets the proxy port for HTTP connections. This setting is only used when Use Proxies is checked.	int	.httpProxyPort	Data
HTTP Proxy Server	HTTP Proxy Server sets the proxy server for HTTP connections. This setting is only used when Use Proxies is checked. Can be empty	String	.httpProxyServer	Data
HTTPS Proxy Port	HTTPS Proxy Port sets the proxy port for HTTPS connections. This setting is only used when Use Proxies is checked.	int	.httpsProxyPort	Data
HTTPS Proxy Server	HTTPS Proxy Server sets the proxy server for HTTPS connections. This setting is only used when Use Proxies is checked. Can be empty	String	.httpsProxyServer	Data
Mode	Data source for browser. Mode controls whether Starting URL or Starting HTML will be used.	int	.mode	Data
Name	The name of this component.	String	.name	Common
Popups Allowed	This flag is used to allow popups in the web page displayed.	boolean	.popupsAllowed	Behavior
Proxy Exceptions	A comma delimited list of rules for websites that will bypass the proxy servers. An example string would be "*foo.com,<local>,127.0.1". This setting is only used when Use Proxies is checked.	String	.proxyExceptions	Data
Proxy Password	The password to use for proxy authentication. This setting is only used when Use Proxies and Use Proxy Authentication are checked.	String	.proxyPassword	Data
Proxy Username	The username to use for proxy authentication. This setting is only used when Use Proxies and Use Proxy Authentication are checked.	String	.proxyUsername	Data
SOCKS Proxy Port	The port number for SOCKS proxies.	int	.socksProxyPort	
SOCKS Proxy Server	The host name to use for SOCKS proxies. Can be empty.	String	.socksProxyServer	
Show Navigation Buttons	Show the navigation buttons at the top of the frame.	boolean	.showNavigation	Behavior
Starting HTML	<p>The initial HTML displayed when the Mode is set to HTML.</p> <p>Starting HTML is</p> <pre><html><body>&nbsp;</body></html></pre> <p>by default, which gives a blank page.</p>	String	.startingHtml	Data
Starting URL	The initial URL displayed when the Mode is set to URL. Starting URL is blank by default.	String	.startingUrl	Data
Touchscreen Mode	Controls when this input components responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
Use Proxies	If checked, the Web Browser will try to use the proxy settings.	boolean	.useProxies	Data
Use Proxy Authentication	If checked, the browser will use the username and password for proxy authentication. This setting is only used when Use Proxies is checked.	boolean	.useProxyAuthentication	Data
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Zoom Level	The zoom level the web page is displayed in. 0.0 is normal, positive numbers zoom in, negative numbers zoom out.	double	.zoomLevel	Behavior

Scripting

Scripting Functions

- Description

This function will return the underlying browser object. See [JxBrowser Programmer's Guide](#) for more information.

- Parameters

none

- Return

[Object](#) - The Browser Object

- Scope

Client

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

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

Customizers

- [Vision Component Customizers](#)

Examples

Examples

Setting Chromium Switches via JVM Arguments

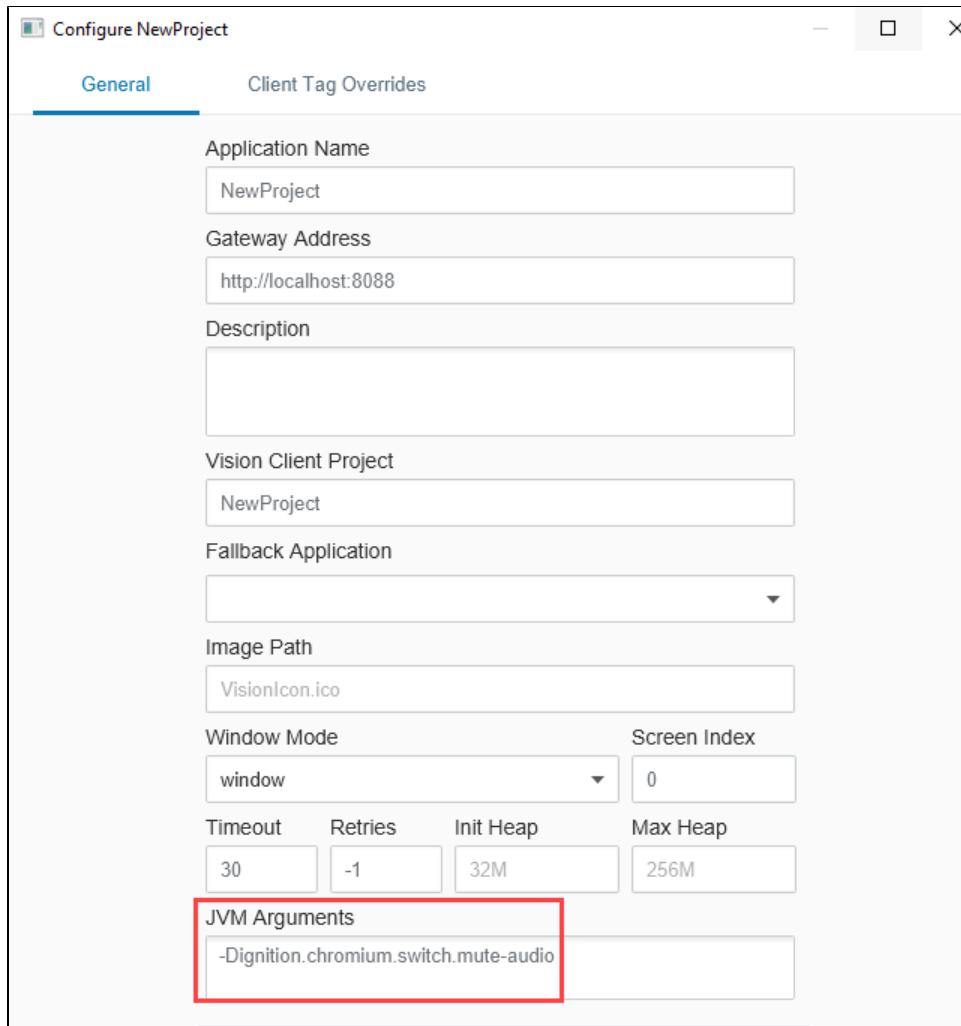
The Web Browser component is based off of the [JxBrowser library](#), which in turn is based upon the Chromium engine. As a result, the Web Browser component can be further customized by manipulating [Chromium Switches](#).

Caution: Implementing these switches is considered **unsupported** because they can drastically change the behavior of the Web Browser component. The exception to this case is when a member of our support team requests a switch be added to help troubleshoot an issue. For the sake of clarity, instructions on how to manipulate the switches via the Designer Launcher and Vision Client Launcher are listed below, but we generally do not recommend users implement these switches.

If you're going to make use of a switch, then you would do so on the Designer Launcher's/Vision Client Launcher application, under the JVM Arguments field. Below is an example on how to configure a switch for a client using the Vision Client Launcher. The same method applies for the Designer Launcher.

1. Open the Vision Client Launcher.
2. Once open, either create a new application or [manage the settings](#) on an existing application.
3. Once the Settings are open, add a new entry into the JVM Arguments text area. Arguments for Chromium Switches must have a prefix of "-Dignition.chromium.switch." followed by the argument. Below is a example where we set the argument "mute-audio":

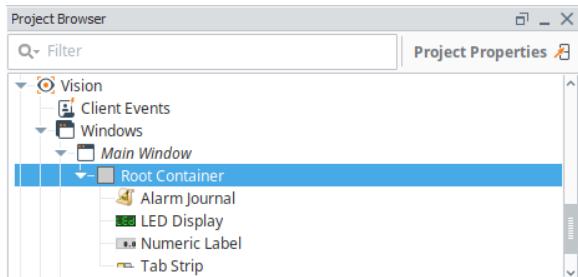
```
-Dignition.chromium.switch.mute-audio
```



4. Following this change, audio from the Web Browser Component will be muted once the client is launched.

Vision - The Window Object

General



INDUCTIVE
UNIVERSIT

Window Properties

[Watch the Video](#)

Description

Window

Windows are the top-level unit of design for Vision projects. A window is identified by its path, which is the name of all its parent folders plus its name, with forward slashes (/) in between. For example, the path to a window in the top level called MainWindow would simply be its name, whereas the path to a window named **UserOptions** under a folder called **OptionsWindows** would be: **OptionsWindows/UserOptions**.

A window may display a Titlebar and/or a Border. The titlebar allows the user to drag the window around in the client, and houses the window's close and maximize/restore buttons. The border of a window can be used to resize the window in the client when it is floating or docked. Whether or not the titlebar and border are displayed depends on the values of the window's titlebar and border display policy properties, and its current state. Commonly, a window will display both a titlebar and border when it is configured as a popup. It is often desirable to remove titlebars and borders on main windows so they join seamlessly with docked windows.

The user manual describes different [Window Types](#), but technically there is only a single window object in the Vision module: different "types" of windows are simply instances of the window object configured in different ways. See [Window Types](#) for more information about changing types.

Root Container

Inside a window is always the Root Container. The Root Container is where you will place all of your components in the window. This is exactly the same as a normal [container](#) component except that it cannot be deleted. When in the designer, "resizing" the window from the main Vision workspace is really changing the size of the Root Container.

Window Opening Event Order

Window objects have several event handlers that trigger when the window opens. However, each event handler occurs at a separate time. Because of this, it is important to understand the order that these events occur:

Opening a window - When opening a window for the first time in a designer, the following event handlers are called in order:

1. visionWindowOpened - Important to notice the description on this event: it occurs before any bindings on the window are evaluated.
2. internalFrameOpened - If the window has been cached, this will not fire on sequential opens.
3. internalFrameActivated - The last event, but also repeatable while the window is opened, since this event will trigger again if the window loses and then regains focus without being closed in between.

Closing a window - When closing a window, the following event handlers are called in order:

1. internalFrameClosing - This event would be ideal to "clean up" in the window, since the window is still technically open at this point.
2. visionWindowClosed - Triggers when the window is closed. Functionally, this is similar to internalFrameClosed, but happens slightly earlier.
3. internalFrameDeactivated - This triggers when the window is closed, or when the window loses focus, so you may want to avoid this event if your script should only trigger when the window is closed.
4. internalFrameClosed - Similar to visionWindowClosed. Triggers when the Java windowing system has finished closing the window.

Properties

Name	Description	Property Type	Scripting	Category

Border Display Policy	Determines if the window's border is shown in various window states.	int	. borderDisplayPolicy	Behavior				
	<table border="1"> <thead> <tr> <th>Integer</th><th>Property</th></tr> </thead> <tbody> <tr> <td>0</td><td>Always</td></tr> <tr> <td>1</td><td>Never</td></tr> <tr> <td>2</td><td>When Not Maximized</td></tr> </tbody> </table>				Integer	Property	0	Always
Integer	Property							
0	Always							
1	Never							
2	When Not Maximized							
Cache Policy	By default this property is set to Auto , which keeps a window in a memory cache for a while after it is closed, so that if it is opened again it will be quick. The window isn't "active" while it is closed: all of its bindings and scripts are shut down.	int	.cachePolicy	Behavior				
	Setting this property to Never causes a fresh copy of the window to be deserialized every time it is opened. This is a performance hit, but it also is a convenient way to "clear out" the values of the window from the last time it was opened, which can be helpful in data-entry screens.							
	Setting the property to Always will trade memory for higher performance, causing the window to always remain cached after the first time it is opened. This means the window will open very fast, but your Client will need lots of memory if you do this to a large amount of windows.							
Closeable	Determines whether or not to draw the close (X) button in the upper right corner.	boolean	.closable	Behavior				
Dock Index	Determines the order of docked windows if multiple windows are open on the same edge. Lower numbers are on the outside (closest to the edge the window is docked to), and higher numbers are closer to the center.	int	.dockIndex	Layout				
Dock Position	Determines the position this window is docked to, or if it is floating.	int	.dockPosition	Layout				
Layer	Sets the layer that this window is in. Default layer is 0, which is the bottom layer. Windows in higher layers will always be shown on top of windows in layers beneath them. A common strategy for using the layer property is to set Main Windows and Docked windows to 0, Popups to 1 and very important popups to 2.	int	.layer	Layout				
Location	The starting location that this window will open up at. Only applicable to floating windows that are not set to start maximized. This value will be overridden when an open window script specifies where to open.	Point	.startingLocation	Layout				
Maximizable	Determines whether or not to draw the maximize button in the upper right corner.	boolean	.maximizable	Behavior				
Maximum Size	The maximum size that this window will allow itself to be resized to.	Dimension	.maximumSize	Layout				
Minimum Size	The minimum size that this window will allow itself to be resized to.	Dimension	.minimumSize	Layout				
Resizable	Determines whether or not to let the user resize the window.	boolean	.resizable	Behavior				
Size	The dimensions of the window. This can be manipulated by selecting the window and dragging the resize handles along the windows right and bottom edges.	Dimension	.size	Layout				
Start Maximized	When set to true, the window will become maximized when it is opened.	boolean	.startMaximized	Behavior				
Title	The title to be displayed in this window's titlebar. The title is also used in the Client's Windows menu.	String	.title	Appearance				
Titlebar Display Policy	Determines if window's titlebar is shown in various window states.	int	.titlebarDisplayPolicy	Appearance				
	<table border="1"> <thead> <tr> <th>Integer</th><th>Property</th></tr> </thead> <tbody> <tr> <td>0</td><td>Always</td></tr> </tbody> </table>				Integer	Property	0	Always
Integer	Property							
0	Always							

	1	Never			
	2	When Not Maximized			
Titlebar Font	The font of the window title in the titlebar.		Font	.titlebarFont	Appearance
Titlebar Height	The height of the window's titlebar.		int	.titlebarHeight	Appearance

Scripting

Scripting Functions

- Description

Returns a reference to the Root Container in the window.

- Parameters

none

- Return

Object - a reference to the Root Container, which is functionally just a [Vision - Container](#).

- Scope

Client

- Description

Returns a reference to a component. The path parameter allows you to specify the full path to the component as a string.

- Parameters

String path - The path to the component, using a period as a delimiter, such as "Root Container.Group.Label".

- Return

Object - to the component specified, or None if there is a typo in the path.

- Scope

Client

Extension Functions

This component does not have any extension functions.

Event Handlers

An "internalFrame" refers to the underlying object Java windowing system that windows in the Vision module use.

Fires whenever the window is shown or focused. If you want a script to fire every time a window is opened, use this event.

.source	The window that fired this event. Use source.rootContainer to get the root container.
---------	---

Fires when a window is closed.

.source	The window that fired this event. Use source.rootContainer to get the root container.
---------	---

Fires right before a window is closed.

.source	The window that fired this event. Use source.rootContainer to get the root container.
---------	---

Fires when a window loses focus.

.source | The window that fired this event. Use source.rootContainer to get the root container.

Fires the first time a window is opened. When windows are closed, they may be cached. If a window in a client is cached, subsequent attempts to open the window will not trigger this event. If you disable caching (by setting the **Cache Policy** property to **Never**) then this event will trigger every time the window is opened. Alternatively, you could use **internalFrameActivated** instead to consistently trigger a script when a window is opened.

.source | The window that fired this event. Use source.rootContainer to get the root container.

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

.source	The component that fired this event.
.newValue	The new value that this property changed to.
.oldValue	The value that this property was before it changed. 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.

This event is fired each time the window is opened and before any bindings are evaluated.

.source | The window that fired this event. Use source.rootContainer to get the root container.

This event is fired each time the window is closed.

.source | The window that fired this event. Use source.rootContainer to get the root container.

Examples

For examples of windows, please see the [Vision Windows](#) section.

Copy of -- WIP Page just with frequently used notes

Note: The border is unaffected by rotation.

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

Note: Remember to always filter out these events for the property that you are looking for. Components often have many properties that change.

Report Design Components

Reporting Design Components

The Report Module features many different report-specific components that are exclusively used inside the Report Module.

Here is a complete list of the Report Design components and a link pointing to a page containing the component's description, properties, and an example of how to configure it.

[Report Design - Chart Palette](#)

[Report Design - Shape Palette](#)

[Report Design - Table Palette](#)

[Report Design - Display Palette](#)

[Report Design - Object Palette](#)

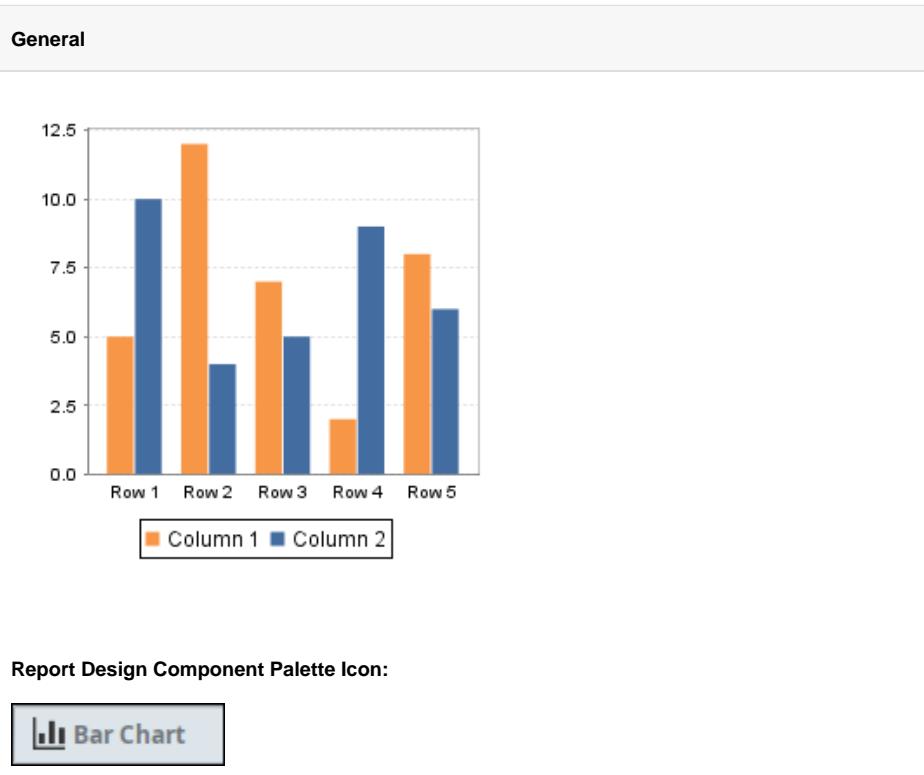
Report Design - Chart Palette

Chart Components

The following components give you various chart types for displaying data in a report.

[In This Section ...](#)

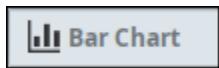
Report - Bar Chart



Bar Chart

[Watch the Video](#)

Report Design Component Palette Icon:

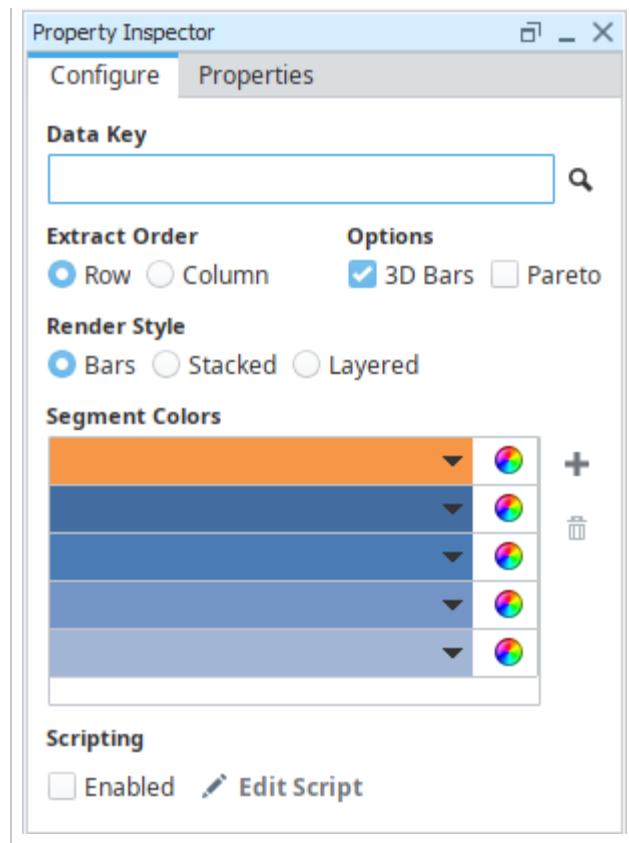


Description

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). See [Color Selector](#).

Color Wheel Tool for creating color combinations for bars. See [Color Selector](#).

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.

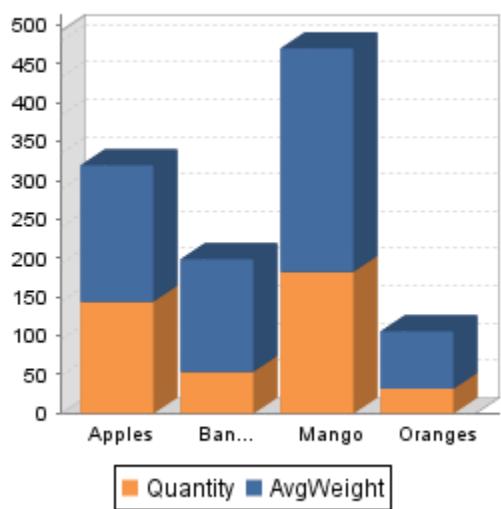
Properties Tab	Property	Description	Category
	Data Key	Data key of the dataset that drives this pie chart.	Chart Options
	Category Margin	Spacing between groups of bars on the chart.	Chart Options
	Extract Order	Determines whether the series are defined by rows or columns in the underlying dataset.	Chart 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
	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

Chart Options	
Data Key	
Category Margin	0.2
Extract Order	Row
Item Margin	0.02
Render Style	Bars
3D Bars	<input checked="" type="checkbox"/>
Show Legend	<input checked="" type="checkbox"/>
Legend Font	Dialog, Plain, 12
Vertical Chart	<input checked="" type="checkbox"/>
Pareto Options	
Pareto	<input type="checkbox"/>
80% Line	<input checked="" type="checkbox"/>
Pareto Color	<input type="color"/>
Pareto Series Label	Percent
Axis Options	
Axis Autorange	<input checked="" type="checkbox"/>
Axis Font	Dialog, Plain, 12
Axis Format	
Axis Label	
Axis Low	0
Axis High	100
Label Position	Standard
Label Options	
Bar Labels	<input type="checkbox"/>
Label Color	<input type="color"/>
Label Font	Dialog, Plain, 12
Label Offset	2
Stroke and Fill	
Opacity	1
Stroke Style	Hidden
Stroke	
Basic Properties	
Roll	0
Scale X	1
Scale Y	1
Visible	<input checked="" type="checkbox"/>
Width	442
Height	344.432
X	50
Y	87.568

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 off, 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
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

Bar Chart in Action



Bar Chart Sample CSV Data

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

Property Inspector

Configure Properties

Data Key: Fruit Data

Extract Order: Row (radio button selected) Options: 3D Bars (checkbox checked), Pareto (checkbox unselected)

Render Style: Stacked (radio button selected, highlighted with a red box)

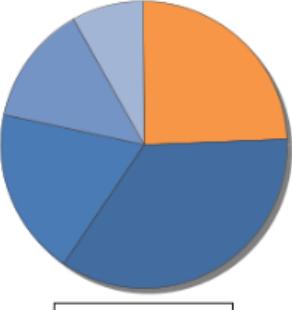
Segment Colors: A list of five color swatches, each with a dropdown arrow and a color palette icon. The first color is orange, and the others are shades of blue.

Scripting: Enabled (checkbox unselected), Edit Script (button)

Property	Value
3D Bars	True
Render Style	Stacked

Report - Pie Chart

General



A pie chart divided into five segments. Segment A is orange, Segment B is dark blue, Segment C is medium blue, Segment D is light blue, and Segment E is very light blue. Below the chart is a legend with colored circles corresponding to the segments: A (orange), B (dark blue), C (medium blue), D (light blue), and E (very light blue).

Report Design Component Palette Icon:



A small icon representing a pie chart, located within the Report Design Component Palette.



INDUCTIVE
UNIVERSITY

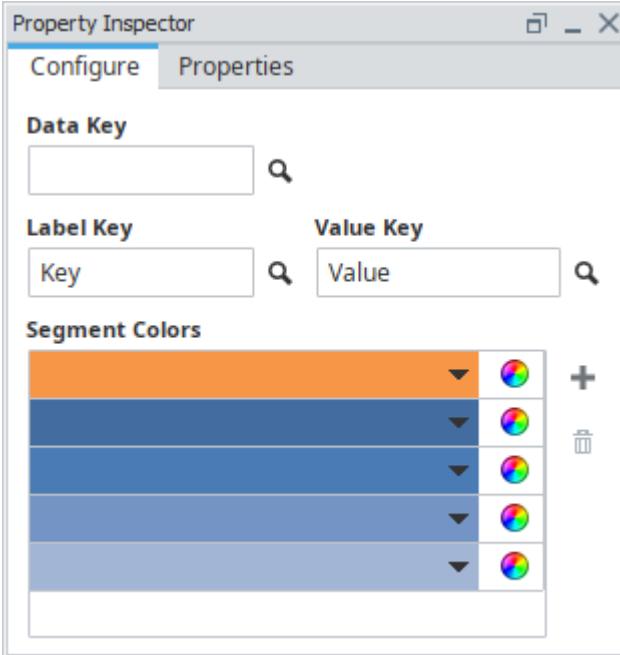
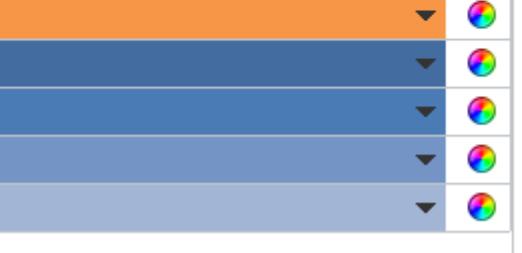
Pie Chart

[Watch the Video](#)

Description

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

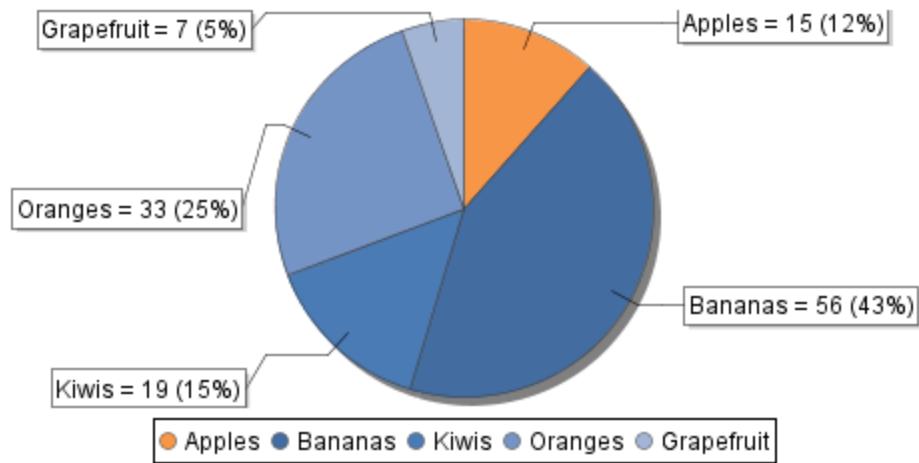
Properties

Configure	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 	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.

Property Inspector		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. See Color Selector .	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
		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

Pie Chart in Action



Pie Chart Sample CSV Data

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

Property Inspector

Configure Properties

Data Key
Fruit Data

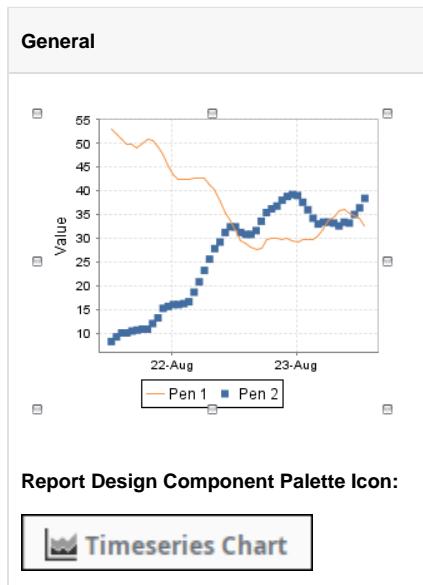
Label Key **Value Key**
Key Quantity

Segment Colors

Orange	<input type="button" value="Color"/>
Dark Blue	<input type="button" value="Color"/>
Light Blue	<input type="button" value="Color"/>
Medium Blue	<input type="button" value="Color"/>
Light Purple	<input type="button" value="Color"/>
Blank	<input type="button" value="Color"/>

Property	Value
Label Style	Outset
Value Key	Quantity

Report - Timeseries Chart



Description

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 /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.
		Deletes a pen.

Property Inspector

Chart Options Properties

Data Key **Domain Key**

Pens

Range Key	Pen Name	Preview
Test1	Pen 1	
Test2	Pen 2	

Axes

Name	Label	Auto?	Low	High
Default Axis	Value	<input checked="" type="checkbox"/>	0	100

Scripting

Enabled [Edit Script](#)

Remove	
Pen	
Axes	<p>There are several properties for each axis exposed on the Chart Options tab:</p> <p>Name: The name of the axis.</p> <p>Label: Text next to the axis. Commonly used to describe what the values of each pen represent.</p> <p>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.</p> <p>Low: The low range value the axis will use when 'Auto?' is disabled.</p> <p>High: The high range value the axis will use when 'Auto?' is disabled.</p>
Add Axis	Add an axis.
Edit Axis	Opens Axis Configuration area for editing.
Remove	Deletes an axis.
Axis	
Scripting Enabled	Enables the script for this component to run.
Edit Script	Allows the chart to be modified via Python scripting before the report is rendered.

Properties Tab	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. See Color Selector .	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,	Stroke and Fill

Property Inspector

Chart Options

- Data Key**
- Axis Label Font**: Dialog, Plain, 12
- Axis Tick Font**: Dialog, Plain, 10
- Bar Width**: 1,000
- Date Format**
- Domain Key**: t_stamp
- Gap Threshold**: 3
- No Data Msg**: No Data
- Plot Background**
- Show Legend**:

Stroke and Fill

- Opacity**: 1
- Stroke Style**: Hidden
- Stroke**

Basic Properties

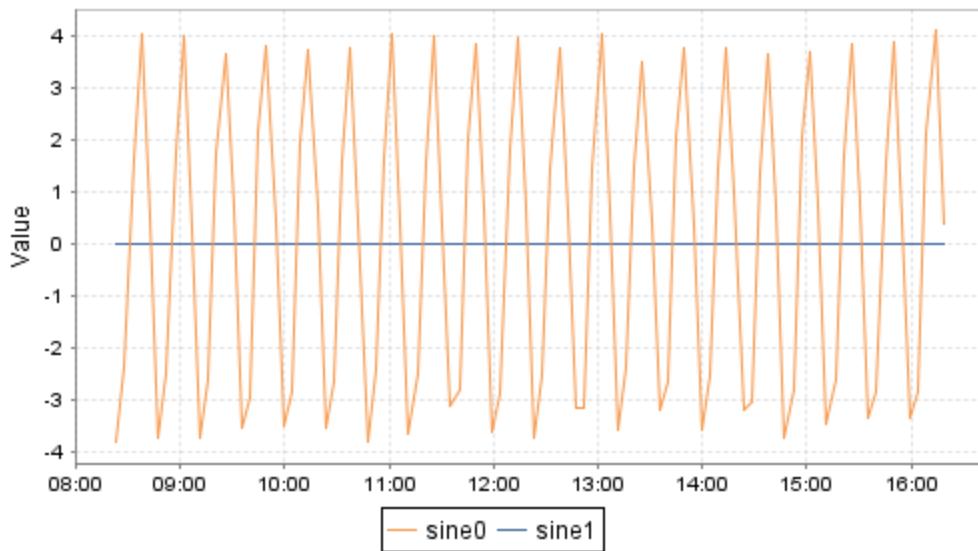
- Roll**: 0
- Scale X**: 1
- Scale Y**: 1
- Visible**:
- Width**: 300
- Height**: 200
- X**: 140.909
- Y**: 144.318

X-Axis

- Show X-Axis**:
- X-Axis Label**

	Border, or Double. To learn more about stroke styles, refer to Stroke and Fill Properties .	
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
Show X-Axis	If true, show the x-axis.	X-Axis
X-Axis Label	Label text for the x-axis.	X-Axis

Example



Property Inspector

Chart Options **Properties**

Data Key tag_history **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	<input checked="" type="checkbox"/>	0	100

Scripting

Enabled [Edit Script](#)

Example

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
```

Report - XY Chart

General

The chart displays two data series: Pen 1 (orange line) and Pen 2 (blue dotted line). Both series exhibit a periodic, wave-like pattern across the domain from 0 to 10. Pen 1 starts at approximately 0.25, peaks at 1.0 around x=1, dips to -0.8 around x=4.5, and returns to 1.0 around x=8. Pen 2 starts at 1.0, dips to 0.25 around x=1.5, reaches a minimum of -1.0 around x=4, and returns to 1.0 around x=7.5.

Domain (X)	Pen 1 (Value)	Pen 2 (Value)
0	0.25	1.0
1	1.0	0.25
2	0.25	0.25
3	-0.8	-0.8
4	-0.8	-1.0
5	-0.8	-0.8
6	-0.8	0.25
7	1.0	0.25
8	0.25	1.0
9	0.25	0.25
10	1.0	-0.8

Report Design Component Palette Icon:

The icon for the XY Chart component is a blue square containing a white line-art icon of a chart with two axes and a grid. Below the icon, the text "XY Chart" is written in a smaller font.

**INDUCTIVE
UNIVERSITY**

XY Chart

[Watch the Video](#)

Description

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 /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 -pencil	Opens Pen Configuration area for editing.
	Remove Pen trash	Deletes a pen.

Property Inspector

Chart Options Properties

Data Key **Domain Key**

Pens

Range Key	Pen Name	Preview
Test1	Pen 1	
Test2	Pen 2	

Axes

Name	Label	Auto?	Low	High
Default Axis	Value	<input checked="" type="checkbox"/>	0	100

Scripting

Enabled Edit Script

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.
Remove Axis	Deletes an axis.
Scripting Enabled	Enables the script for this component to run.
Edit Script	Allows the chart to be modified before the report is rendered via Python script.

Properties Tab		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. See Color Selector .	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

Property Inspector

Chart Options **Properties**

Chart Options

Axis Label Font	Dialog, Plain, 12
Axis Tick Font	Dialog, Plain, 10
Bar Width	1
Domain Key	x
Gap Threshold	3
No Data Msg	No Data
Plot Background	
Show Legend	<input checked="" type="checkbox"/>

X-Axis

Auto Range	<input checked="" type="checkbox"/>
Axis High	100
Axis Low	0
Margin	0.05
Number Format	
Show X-Axis	<input checked="" type="checkbox"/>
X-Axis Label	

Stroke and Fill

Opacity	1
Stroke Style	Hidden
Stroke	

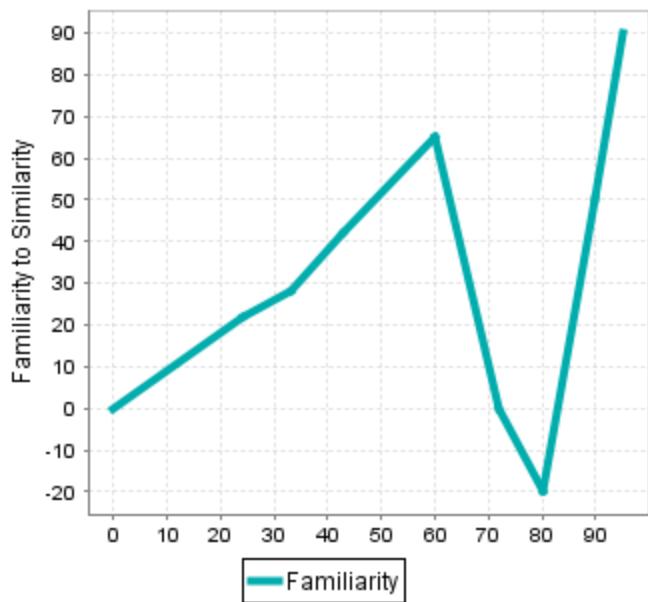
Basic Properties

Roll	0
Scale X	1
Scale Y	1
Visible	<input checked="" type="checkbox"/>
Width	300
Height	200
X	104.202
Y	521.008

Number Format	A optional number format string to override the automatic formatting on the x-axis.	X-Axis
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

XY Chart in Action



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

Property Inspector

Chart Options Properties

Data Key Domain Key

uncanny_valley Human_Similarity

Pens

Range Key	Pen Name	Preview
Familiarity	Familiarity	

Axes

Name	Label	Auto?	Low	High
Default Axis	Familiarity to ...	<input checked="" type="checkbox"/>	0	100

Scripting

Enabled  Edit Script

Report Design - Display Palette

Misc Components

The following components provide various options for displaying values and data.

[In This Section ...](#)

Report - Barcode

General



Report Design 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. Note: The actual barcode, as well as any text to be displayed under the barcode, is generated and rendered when the report is executed or opened in the Preview panel.

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 used in. If you are unsure which code you should be using, please consult your existing barcode documentation.

Barcode Formats

Standard Barcode Formats

Standard Barcode Image	Barcode Format	Description	Industry Used
A standard 1D barcode with the number 123456789999 printed below it.	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

2D Barcode Image	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



ABC 911-234

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 text. 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	Description
<p>Property Inspector</p> <p>Configure Barcode Properties</p> <p>Datakey</p> <input type="text"/> <p>Barcode Format</p> <p>CODE_39</p> <p><input type="checkbox"/> Show Text</p> <p>Barcode Format Info</p> <p>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.</p>	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	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

Property Inspector

Configure Barcode Properties

Barcode

Datakey	CODE_39
Barcode Format	CODE_39
Font	Monospaced, Plain, 12
QR Code Error Correcti...	L
QR Code Version	Auto
Show Text	<input type="checkbox"/>

Stroke and Fill

Opacity	1
Stroke Style	Hidden
Stroke	

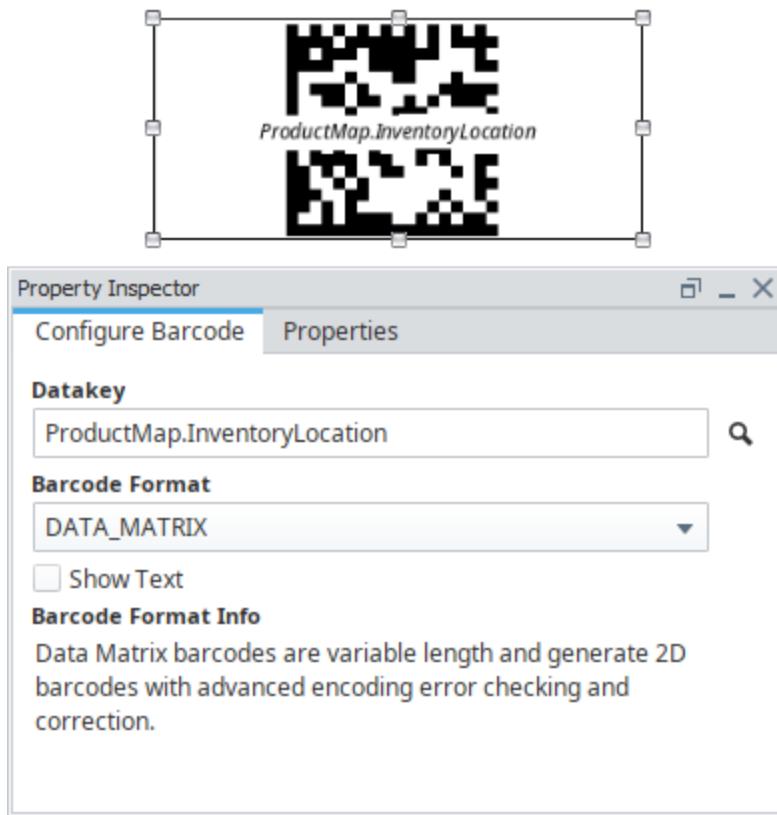
Basic Properties

Roll	0
Scale X	1
Scale Y	1
Visible	<input checked="" type="checkbox"/>
Width	140
Height	60
X	153.846
Y	521.846

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

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.

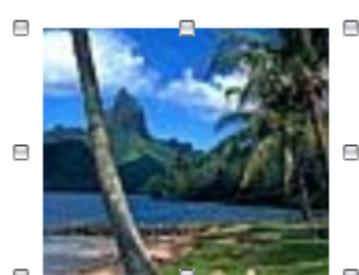
Barcodes require a **Data Key** or Text to encode. **Note**, that each **Barcode Format** encoder has specific requirements for what it will encode. **Barcode Format Info** displays some brief information about the selected format, but is not comprehensive.

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.

Report - Images

General



Report Design 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	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	If true, preserve the aspect ratio when resizing the image.	Image Shape

Property Inspector

The screenshot shows the 'Property Inspector' window with the following settings:

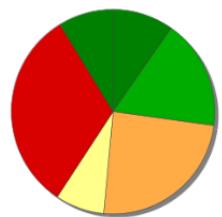
- Stroke and Fill**
 - Fill: Solid black
 - Fill Color: Black
 - Opacity: 1
 - Stroke Style: Hidden
- Image Shape**
 - Grow To Fit: Checked
 - Key: 0
 - Padding: 0
 - Page Index: 0
 - Preserve Aspect Ratio: Checked
- Rectangle Shape**
 - Radius: 0
- Basic Properties**
 - Roll: 0
 - Scale X: 1
 - Scale Y: 1
 - Visible: Checked
 - Width: 148.312
 - Height: 90.76
 - X: 99.688
 - Y: 163.24

Ratio		
Radius	The 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
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

Vegetable Report



■ Squash ■ Zucchini ■ Carrots ■ Potatoes ■ Beans ■ Peppers

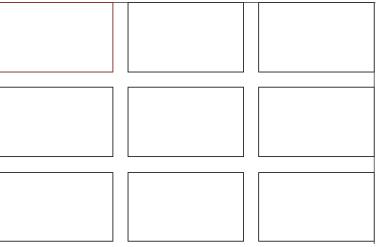
Vegetable	Qty
Squash	10
Zucchini	18
Carrots	25
Potatoes	8
Beans	33
Peppers	9

Property	Value
Key	Image Path

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

Report - Labels

General



Report Design Component Palette Icon:



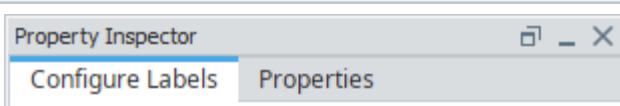
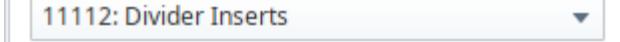
Description

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	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
		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	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

Property Inspector

Edit Text **Properties**

Text Properties

Text Color	<input type="color" value="black"/>	<input type="color"/>
Character Spacing	0	
Coalesce Newlines	<input type="checkbox"/>	
Font	Dialog.plain 12.0 (Dia...)	
Horizontal Alignm...	Left	
Line Spacing	1	
Margin	1; 2; 0; 2	
Overflow Behavior	Grow row	
Underlined	<input type="checkbox"/>	
Vertical Alignment	Top	

Data Key Format Properties

Date Format	MMM d, y	<input type="button" value="..."/>
Null Format	<N/A>	
Number Format	#0.##	<input type="button" value="..."/>
Negative In Red	<input type="checkbox"/>	

Stroke and Fill

Fill	<input type="checkbox"/>	
Fill Color	<input type="color" value="black"/>	<input type="color"/>
Opacity	1	
Stroke Style	Hidden	
Stroke		

Rectangle Shape

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

Basic Properties

Roll	0	
Scale X	1	
Scale Y	1	
Visible	<input checked="" type="checkbox"/>	
Width	108	
Height	43	
X	20.656	
Y	22.214	

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. <ul style="list-style-type: none"> 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
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
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 Company



Candy Miller
Designer
Speciality Cards



Jimmy Rush
Marketing Specialist
Vegan Foods

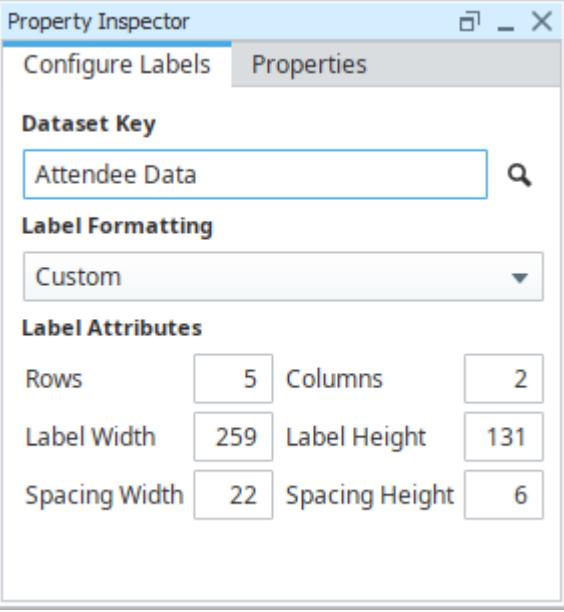


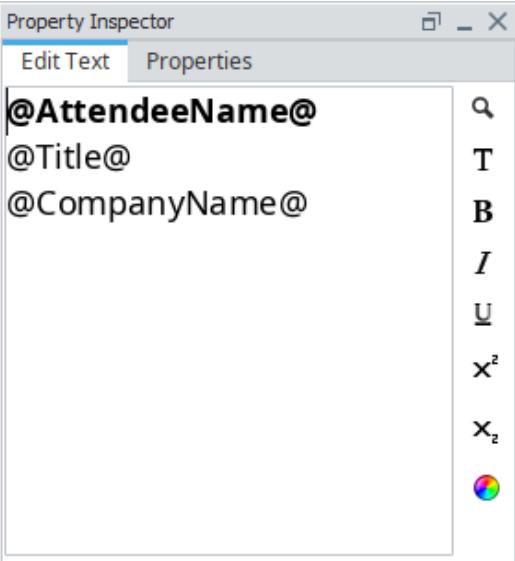
Valerie Rain
Landscape Designer
Elegant Landscapes

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 Company
 Candy Miller, Designer, Speciality Cards
 Jimmy Rush, Marketing Specialist, Vegan Foods
 Valerie Rain, Landscape Designer, Elegant Landscapes

Configure Labels Tab		Property	Value
		Dataset Key	Attendee Data
		Label Formatting	Custom
		Rows	5
		Columns	2
		Label Width	259 pixels
		Label Height	131 pixels
		Spacing Width	22 pixels
		Spacing Height	6 pixels

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

Report Design - Object Palette

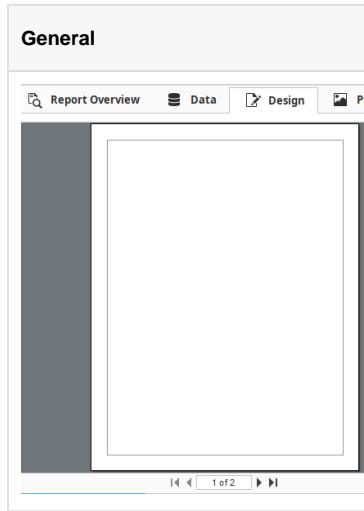
Object Components

The following are object components used solely by the Report Module.

[In This Section ...](#)

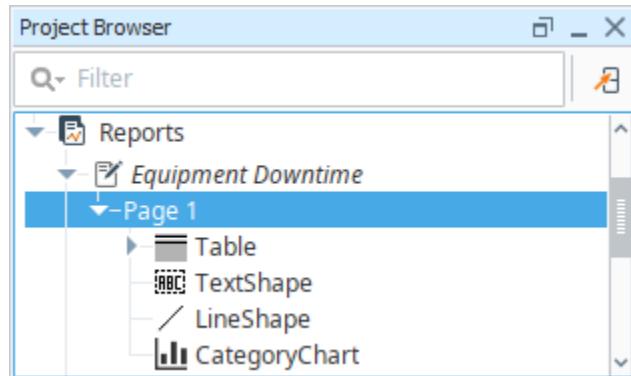
Report - Page Object

The Page Object

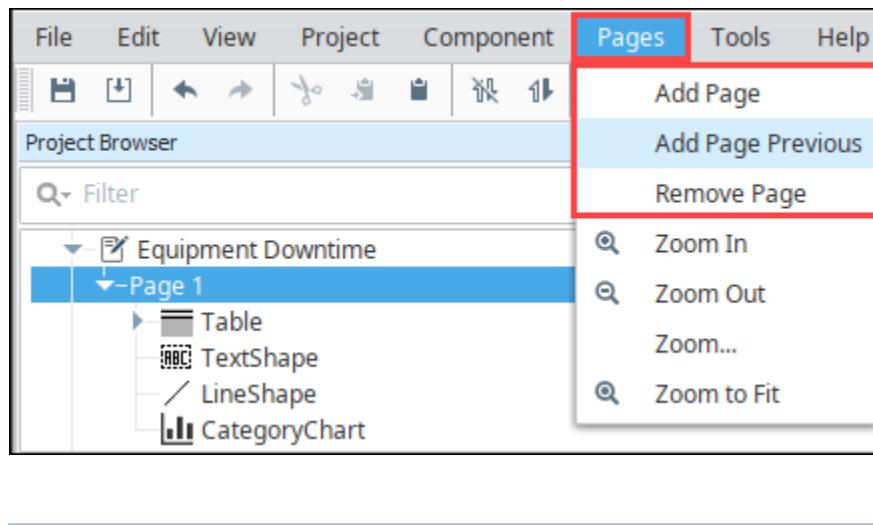


Description

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

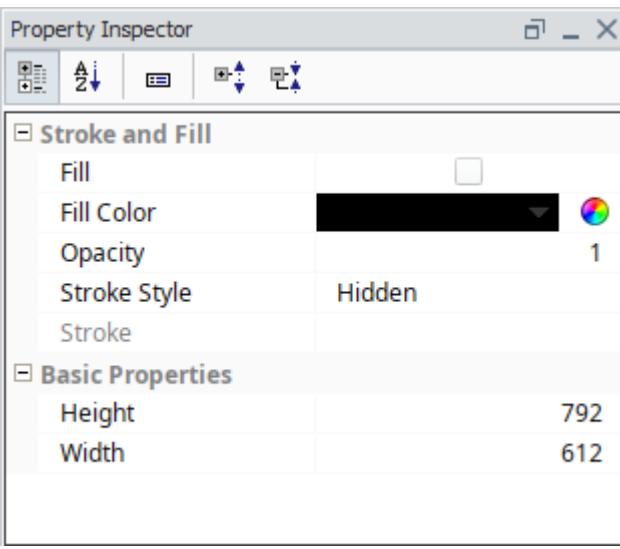


Pages are added and removed from the the menu bar at the top of the browser.



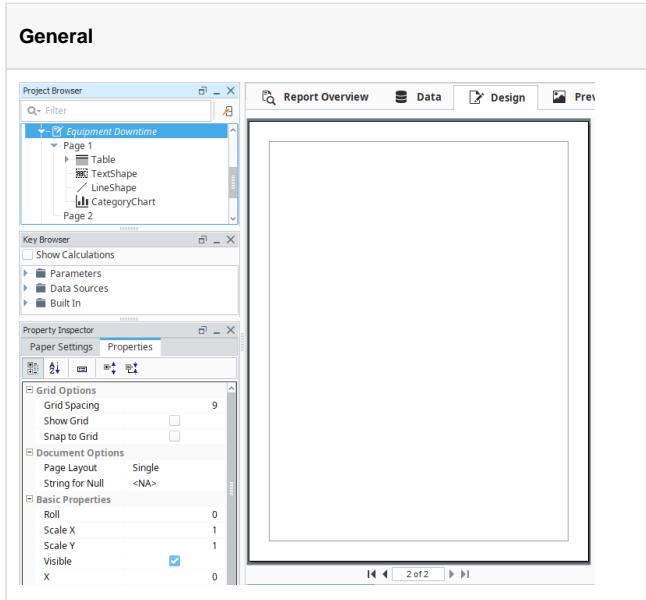
Menu Item	Description
Add Page	Adds a new page to the report after the currently selected page.
Add Page Previous	Adds a new page to the report prior to the currently selected page. A cover page may easily be added to a pre existing report by using Add Page Previous.
Remove Page	Deletes the page that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

Properties

Page Properties	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
	Height	Height of this shape in pixels	Basic Properties
	Width	Width of this shape in pixels.	Basic Properties

Report - Report Object

The 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.

Properties

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 be modifying the Paper Size property. This section also contains a Units drop down with the following units of measurement: <ul style="list-style-type: none">• 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)
Properties Tab	Margins	Allows you to specify the position of the margin on the report. The Margins property is located under the 'Grid Options' tab in the Design Panel. The Margin will never appear on the Preview Panel or on the generated reports.
	Grid Spacing	If the Show Grid property is enabled, the margin is visible on the Design Panel. The Margin will never appear on the Preview Panel or on the generated reports.
	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.

<p>The screenshot shows the 'Property Inspector' window with the 'Properties' tab selected. It includes sections for 'Grid Options' (Grid Spacing: 9, Show Grid: off, Snap to Grid: off), 'Document Options' (Page Layout: Single, String for Null: <NA>), and 'Basic Properties' (Roll: 0, Scale X: 1, Scale Y: 1, Visible: checked, X: 0, Y: 0). Below these are 'Orientation' (Portrait selected) and 'Margins' (Left: 36, Top: 36, Right: 36, Bottom: 36). Checkboxes for 'Show' and 'Snap' are also present.</p>	<p>Snap to Grid</p> <p>The Snap property, when enabled, will cause report components to snap to the margin if moved within close proximity. When dragging a component, enabling this property will cause the component to snap to the grid.</p>
<p>Page Layout</p> <ul style="list-style-type: none"> 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. 	<p>Document Options</p>
<p>String for Null</p> <p>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.</p>	<p>Document Options</p>
<p>Roll</p> <p>Number of degrees this shape is rotated clockwise.</p>	<p>Basic Properties</p>
<p>Scale X</p> <p>Amount to scale the width of this shape. 1 is scale to 100%.</p>	<p>Basic Properties</p>
<p>Scale Y</p> <p>Amount to scale the height of this shape. 1 is scale to 100%.</p>	<p>Basic Properties</p>
<p>Visible</p> <p>If true, the shape will be visible.</p>	<p>Basic Properties</p>
<p>Width</p> <p>Width of this shape in pixels.</p>	<p>Basic Properties</p>
<p>Height</p> <p>Height of this shape in pixels.</p>	<p>Basic Properties</p>
<p>X</p> <p>Horizontal distance in pixels between the left edge of this shape and the left edge of the page.</p>	<p>Basic Properties</p>
<p>Y</p> <p>Vertical distance in pixels between the top edge of this shape and the top edge of the page.</p>	<p>Basic Properties</p>

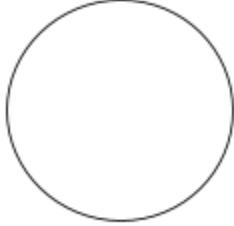
Report Design - Shape Palette

Shape Components

The following components allow users to create shapes.

[In This Section ...](#)

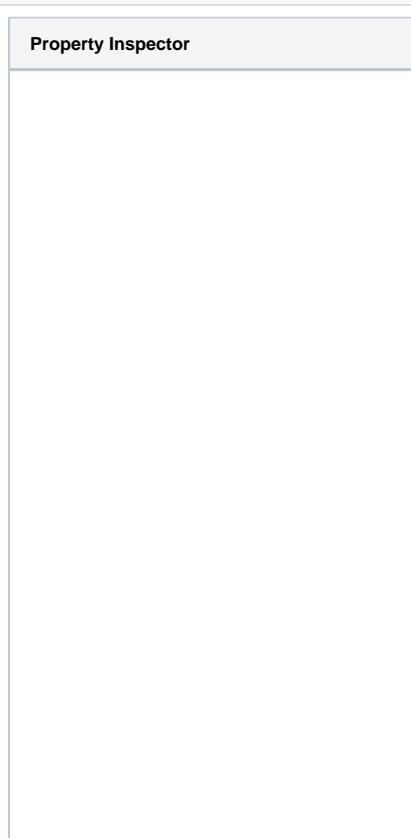
Report - Ellipse Shape

General


Report Design Component Palette Icon:



Description
<p>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.</p>

Properties			
Property Inspector	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; 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. <ul style="list-style-type: none">Color - Color to use for the stroke or border.Dash Pattern - A "Dashed Pattern" string to specify the number of pixels on and off. For example, "5,10"Width - Width of the stroke in pixels.	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

Property Inspector

Stroke and Fill

Fill	<input type="color"/>
Fill Color	<input type="color"/>
Opacity	1
Stroke Style	Shape Outline

Stroke

Color	<input type="color"/>
Dash Pattern	
Width	1

Basic Properties

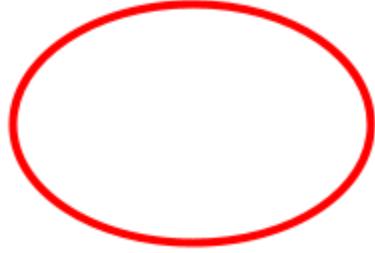
Roll	0
Scale X	1
Scale Y	1
Visible	<input checked="" type="checkbox"/>
Width	133
Height	129
X	89
Y	90

Oval Shape

Start Angle	0
Sweep Angle	360

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
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	FF0000
Stroke - Width	5 pixels
Sweep Angle	360

Report - Line Shape

General



Report Design Component Palette Icon:



Description

The Line shape component can be used to display a straight line anywhere 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 anywhere 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	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; Double. To learn more about stroke styles, refer to Stroke and Fill Properties . <ul style="list-style-type: none">Color - Color to use for the stroke or border.Dash Pattern - A "Dashed Pattern" string to specify the number of pixels on and off. For example, "5,10"Width - Width of the stroke in pixels.	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

Property Inspector

The Property Inspector window displays various properties for a selected object. It includes sections for Stroke and Fill, Stroke, Basic Properties, and Line Properties. Key settings include:

- Stroke and Fill:** Fill color is black, opacity is 1, stroke style is "Shape Outline".
- Stroke:** Color is black, width is 1.
- Basic Properties:** Roll is 0, Scale X and Y are both 1, Visible is checked, X is 100, Y is 300.
- Line Properties:** Stroke color is black, stroke width is 1.

		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
Stroke Width	Width of the stroke in pixels.	Line Properties

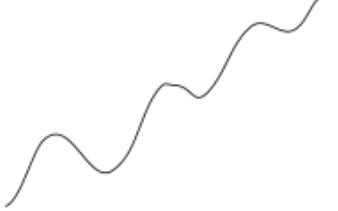
Example - Two Lines

Two thin, horizontal orange lines are displayed on a white background. The lines are positioned vertically, one above the other, centered horizontally.

Property	Value
Stroke Style	Border
Stroke Color	FF8C00
Border Bottom	True
Border Top	True
Stroke Width	5 pixels

Report - Pencil Shape

General



Report Design 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	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; 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. <ul style="list-style-type: none">Color - Color to use for the stroke or border.Dash Pattern - A "Dashed Pattern" string to specify the number of pixels on and off. For example, "5,10"Width - Width of the stroke in pixels.	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 Inspector		Height	Height of this shape in pixels.	Basic Properties
	Z ↑	X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page.	Basic Properties
	Z ↓	Y	Vertical distance in pixels between the upper edge of this shape and the top edge of the page.	Basic Properties
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 <input checked="" type="checkbox"/>				
Width 255				
Height 192.976				
X 95				
Y 451.024				

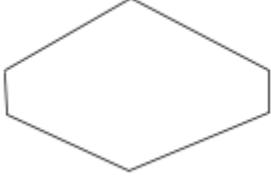
Example



Property	Value
Stroke Style	Shape Outline
Stroke Color	FF0000
Stroke Width	6 pixels

Report - Polygon Shape

General



Report Design 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	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; 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. <ul style="list-style-type: none">Color - Color to use for the stroke or border.Dash Pattern - A "Dashed Pattern" string to specify the number of pixels on and off. For example, "5,10"Width - Width of the stroke in pixels.	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

Property Inspector

Height: 124

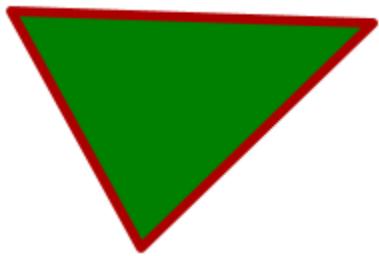
X: 206

Y: 569

Stroke and Fill	
Fill	<input type="color"/>
Fill Color	<input type="color"/> Black
Opacity	1
Stroke Style	Shape Outline
Stroke	
Color	<input type="color"/>
Dash Pattern	
Width	1
Basic Properties	
Roll	0
Scale X	1
Scale Y	1
Visible	<input checked="" type="checkbox"/>
Width	128
Height	124
X	206
Y	569

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

Example



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

Property	Value
Fill Color	008000
Stroke Style	Shape Outline
Stroke Color	AC0000
Stroke Width	7 pixels

Report - Rectangle Shape

General

Report Design Component Palette Icon:


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	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; 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. <ul style="list-style-type: none">Color - Color to use for the stroke or border.Dash Pattern - A "Dashed Pattern" string to specify the number of pixels on and off. For example, "5,10"Width - Width of the stroke in pixels.	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

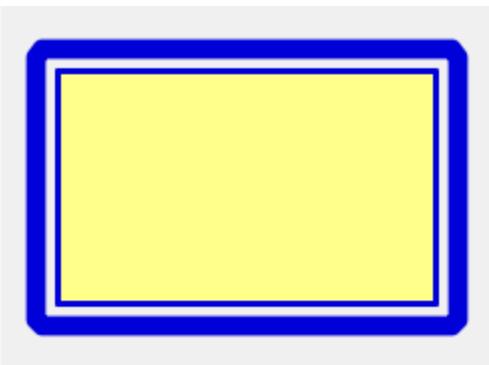
Property Inspector

Height: 103px | Width: 179px | X: 310px | Y: 543px

Stroke and Fill	
Fill	<input type="color" value="#FFFF8A"/>
Fill Color	<input type="color" value="black"/>
Opacity	1
Stroke Style	Shape Outline
Stroke	
Color	<input type="color" value="black"/>
Dash Pattern	
Width	1
Rectangle Shape	
Radius	0
Basic Properties	
Roll	0
Scale X	1
Scale Y	1
Visible	<input checked="" type="checkbox"/>
Width	179
Height	103
X	310
Y	543

		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	Value
Fill Color	FFFF8A
Stroke Style	Double
Stroke Color	0000AC
Stroke Inner Width	3 pixels
Stroke Outer Width	10 pixels
Stroke Position	Inner on path
Stroke Separation	5 pixels

Report - Star Shape

General

Report Design 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	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; 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. <ul style="list-style-type: none">Color - Color to use for the stroke or border.Dash Pattern - A "Dashed Pattern" string to specify the number of pixels on and off. For example, "5,10"Width - Width of the stroke in pixels.	Stroke and Fill
	Roll	Number of degrees this shape is rotated clockwise.	Basic Properties

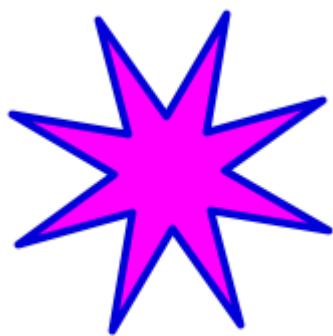
Property Inspector

A A A A

Star Shape	
Bloat	0.382
Points	5
Star Type	Star
Stroke and Fill	
Fill	<input type="color"/>
Fill Color	<input type="color"/>
Opacity	1
Stroke Style	Shape Outline
Stroke	
Color	<input type="color"/>
Dash Pattern	
Width	1
Basic Properties	
Roll	52.496
Scale X	1
Scale Y	1
Visible	<input checked="" type="checkbox"/>
Width	216.813
Height	216.813
X	77.593
Y	511.593

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

Example



Property	Value
Points	8
Fill Color	FF00FF
Stroke Color	0000D9
Roll	20

Report - Text Shape



Report Design Component Icon:

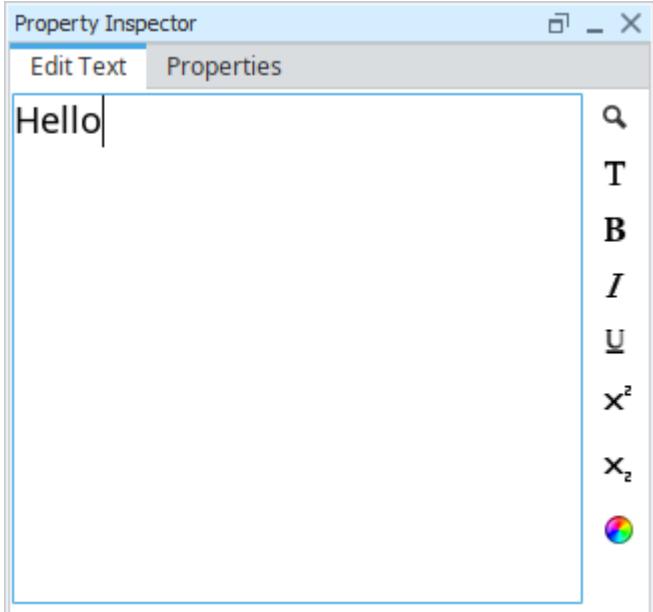
RBC Text

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

Properties Tab	Property	Description	Category
	Text	Text to display in the shape.	Text Properties

Property Inspector

Edit Text Properties

Text Hello

Text Color █

Character Spacing 0

Coalesce Newlines

Font Dialog.plain 18.0 (Dial...)

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, y █

Null Format <N/A>

Number Format #0.## █

Negative In Red

Stroke and Fill

Fill █

Fill Color █

Opacity 1

Stroke Style Hidden

Stroke

Radius 0

Basic Properties

Roll 0

Scale X 1

Scale Y 1

Visible

Width 300

Height 52

X 123

Y 597

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
Margin	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. <ul style="list-style-type: none"> • 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
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; 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
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

Used Car Inventory
April 2020



A pie chart illustrating the distribution of used car inventory by make. The chart is divided into six segments, each representing a different car make: Ford (orange), Toyota (dark blue), Mercedes (light blue), Buick (medium blue), Lexus (very light blue), and Jeep (yellow). The segments are roughly proportional to their quantity in the inventory.

● Ford	● Toyota	● Mercedes	● Buick	● Lexus	● Jeep
--------	----------	------------	---------	---------	--------

Make Quantity

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

Controller
15-Apr-20

Car Inventory Report

1 of 1

Text Shape - Used Car Inventory	Property	Value
	Text	Used Car Inventory

Property Inspector

Text Color	FFFFFF, Bold
Font Size	24 pixels

Used Car Inventory

Text Shape - April 2020

Property	Value
Text	April 2020
Text Color	FFFFFF
Font Size	18 pixels

April 2020

Property Inspector

Text	April 2020
Text Color	FFFFFF
Font Size	18 pixels

Text Properties

Text	April 2020
Text Color	FFFFFF
Font Size	18 pixels

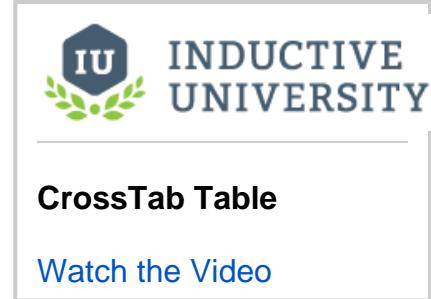
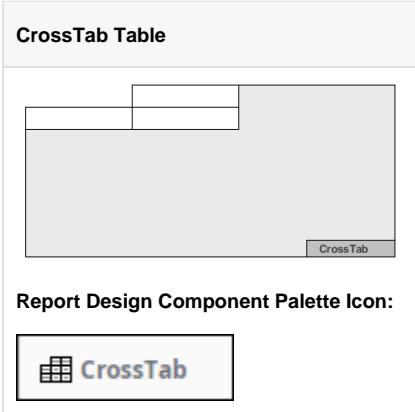
Report Design - Table Palette

Table Components

The following components give you various types of Tables for displaying values.

[In This Section ...](#)

Report - Crosstab Table



Description

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 repetitive 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 repetitive 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. To learn more about the CrossTab Table, refer to the [CrossTab and Simple Tables](#) page.

Table Properties

Table Properties	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. <ul style="list-style-type: none">• 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	Table

Property Inspector

Table		
Data Key	1	
Columns	1	
Filter Key	1	
Header Columns	1	
Header Rows	1	
Reprint Header Rows	<input checked="" type="checkbox"/>	
Rows	1	
Style	Default	
Stroke and Fill		
Fill	<input type="checkbox"/>	
Fill Color		<input type="color"/>
Opacity	1	
Stroke Style	Hidden	
Stroke		
Basic Properties		
Roll	0	
Scale X	1	
Scale Y	1	
Visible	<input checked="" type="checkbox"/>	
Width	450	
Height	375	
X	56.288	
Y	66.718	

The Style section below shows several examples of some of the style templates you can choose from.

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 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

Cell Properties	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.	Text Properties

Property Inspector

The Property Inspector window displays the following settings:

- Text Properties:**
 - Text: Text area.
 - Text Color: Black color swatch.
 - Character Spacing: 0.
 - Coalesce Newlines:
 - Font: Dialog plain 12.0 (Dialog).
 - 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, y.
 - Null Format: <N/A>.
 - Number Format: #0.##.
 - Negative In Red:
- Stroke and Fill:**
 - Fill:
 - Fill Color: Gray color swatch.
 - 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.

- 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.

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
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
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

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.

Cell Functions	Function	Description
	Clear Contents	Deletes all contents from the selected cell.

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

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

Downtime Dataset

```
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

```

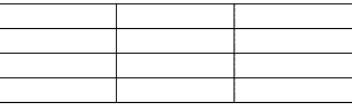
The data keys were used to populate the CrossTab Table with data. The property values used to configure the CrossTab Table are shown in the property list.

CrossTab Table in Design Panel		Property	Value
		Style	Elegant
		Text Color (for Equipment and Site Headers)	FF0000
		Font	14
		Horizontal Alignment	Center
		Horizontal Alignment (for Site)	Right



Report - Simple Table

General



Report Design Component Palette Icon:





INDUCTIVE
UNIVERSITY

Simple Table

[Watch the Video](#)

Description
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. To learn more about the Simple Table, refer to the CrossTab and Simple Tables page.

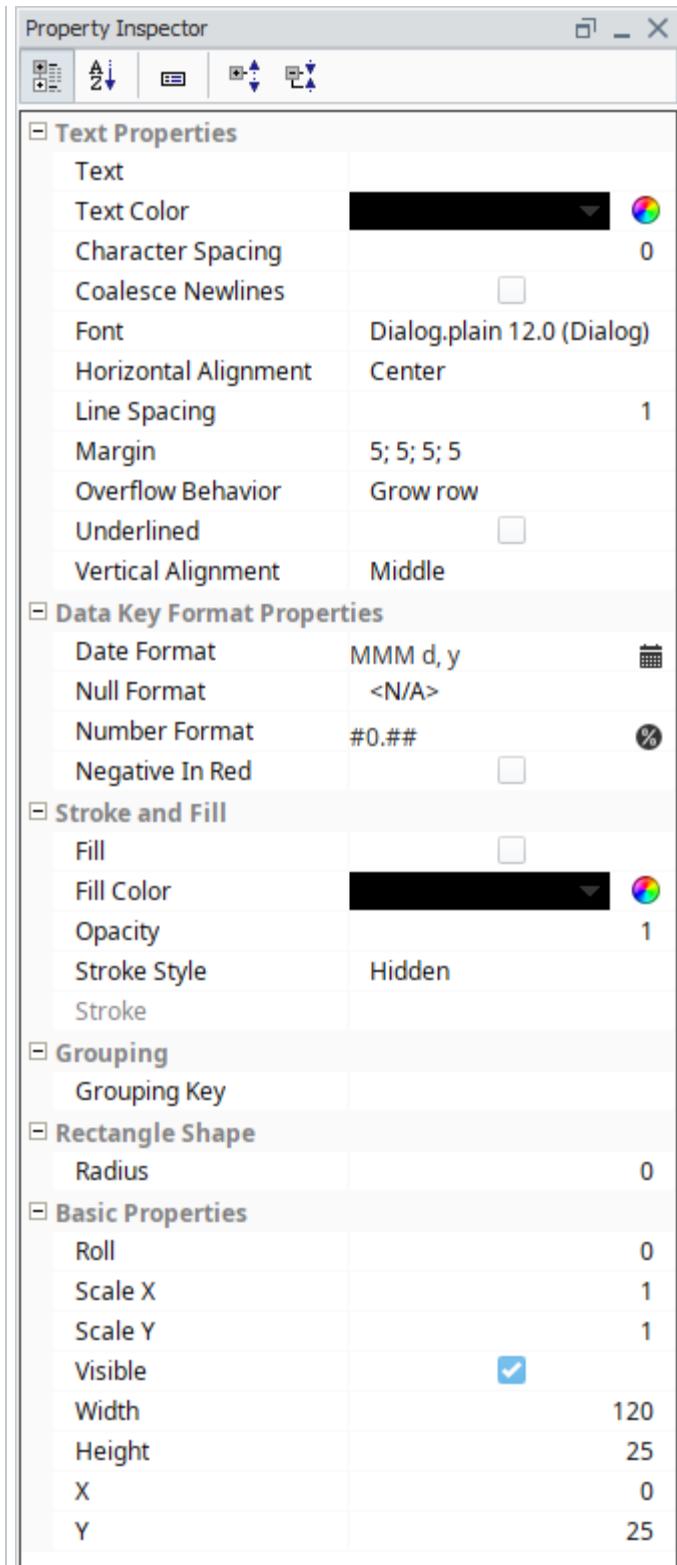
Simple Table Properties	Property	Description	Category
	Data Key	The data key associated with this Tabular component. Should be the data key of a data source that contains rows, such as a query.	Table
	Columns	Number of columns.	Table
	Filter Key	A key or expression used to prune the underlying data source's 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
	Rows	Number of Rows.	Table
	Style	Give the CrossTab or Simple Table a pre-configured style. <ul style="list-style-type: none">• 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 <p>The Style section below shows several examples of some of the style templates you can choose from.</p>	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		Stroke and

Property Inspector

Table	
Data Key	3
Columns	3
Filter Key	0
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	<input checked="" type="checkbox"/>
Width	360
Height	100
X	154.366
Y	334.648

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 .	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 and Functions			
Cell Properties	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.	Text Properties



	<ul style="list-style-type: none"> • 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. 	
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
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 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
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

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.

Cell Functions	Function	Description
	Clear Contents	Deletes all contents from the selected cell.

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

Example

This Simple Table shows the total number of downtime events and downtime minutes, as well as the average number of downtime minutes for each piece of equipment.

Equipment Downtime Events				
	Motor	Conveyor	Palletizer	Pallet Wrapper
Downtime Count	8	8	8	8
Total DT Duration (Min)	218	597	407	426
AVG DT Duration (Min)	27.25	74.62	50.88	53.25

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 below.

Simple Table in the Design Panel	Property	Value
	Style	Default
	Columns	1
	Header Columns	1
	Rows	3
	Column Header	Equipment
	Header in Row 1	Downtime Count
	Header in Row 2	Total DT Duration (Min)
	Header in Row 3	AVG DT Duration (Min)

	@Equipment@	Header Rows Color 0000D9, Bold
Downtime Count	@count@	Header Column Color 0000D9, Bold
Total DT Duration (Min)	@total.Duration_minutes@	
AVG DT Duration (Min)	@average.Duration_minutes@	

Equipment Downtime Dataset - used for this Simple Table example.

```
Equipment, Duration_minutes
"Motor", 52
"Conveyor", 154
"Palletizer", 58
"Pallet Wrapper", 251
"Motor", 33
"Conveyor", 94
"Palletizer", 10
"Pallet Wrapper", 25
"Motor", 23
"Conveyor", 60
"Palletizer", 32
"Pallet Wrapper", 25
"Motor", 12
"Conveyor", 33
"Palletizer", 8
"Pallet Wrapper", 25
"Motor", 24
"Conveyor", 124
"Palletizer", 75
"Pallet Wrapper", 25
"Motor", 43
"Conveyor", 51
"Palletizer", 120
"Pallet Wrapper", 25
"Motor", 13
"Conveyor", 36
"Palletizer", 39
"Pallet Wrapper", 25
"Motor", 18
"Conveyor", 45
"Palletizer", 65
"Pallet Wrapper", 25
```

Report - Table

Report Table



Report Design Component Palette Icon:



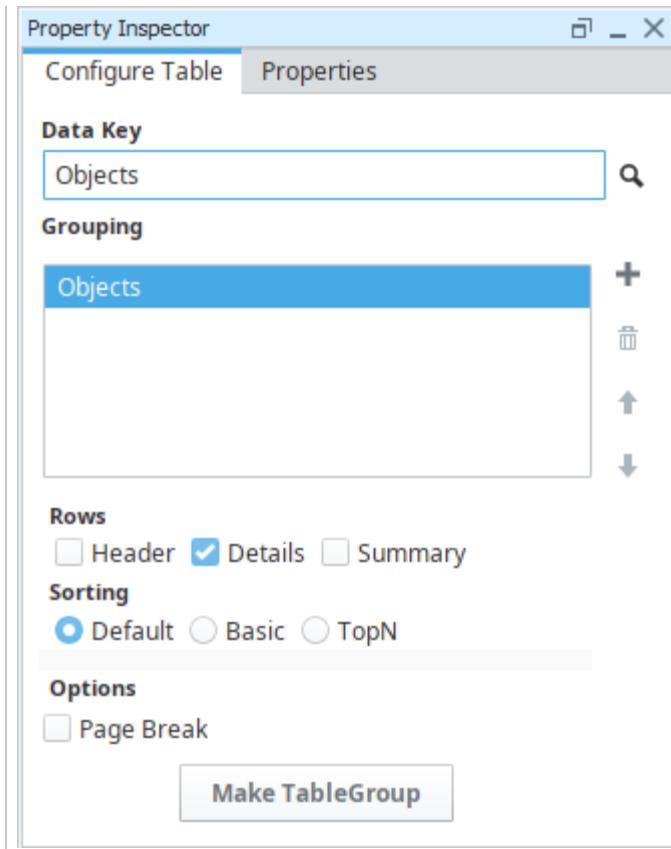
Description

The [Report Table](#) displays data in a structured, repetitive 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	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 [Table Group](#), where you can add 'child' and 'peer' tables.

Table Properties	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
	Table Repeat Count	Renamed the "Column Count" property to "Table Repeat Count". This is the number of times the table will repeat horizontally on a page before paginating. Set your table width to be the width of a single repeat, i.e., a Table Repeat Count of 3 means the width of the table should be 1/3 of the page width.	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

Property Inspector

Configure Table **Properties**

Table

Data Key	Objects
Column Spacing	10
Filter Key	
Table Repeat Count	1

Stroke and Fill

Fill	<input type="checkbox"/>
Fill Color	<input type="color"/>
Opacity	1
Stroke Style	Hidden
Stroke	

Basic Properties

Roll	0
Scale X	1
Scale Y	1
Visible	<input checked="" type="checkbox"/>
Width	400
Height	375
X	70.203
Y	400.304

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

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.

Objects Details		
Table Group		

Configure TableGroup	Name	Icon	Description
	Add Child /Peer Table		<p>Adds a new table to the group. Allows for the creation of a Child or Peer table.</p> <ul style="list-style-type: none"> Child Table: A copy of this table will be inserted in between each row of the parent table

Property Inspector

Configure TableGroup

Properties

Top Object

- Child Object
- Peer Object

Key

Top Object Start on new page false

New page per row false

Peer Table: This table will appear after any preceding peers.

Note: 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.
Caution: 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

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.

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			

Row Properties	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
	<p>Note: 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.</p>		

Property Inspector			
			Pagination
<input checked="" type="checkbox"/>	4	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.
<input type="checkbox"/>		Page Break	Data Key or expressions used to determine whether row should force a page break.
<input checked="" type="checkbox"/>		Reprint When Wrapped	If true, this row will be reprinted on a new page when its children cross a page boundary.
<input type="checkbox"/>		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.
<input type="checkbox"/>		Roll	Number of degrees this shape is rotated clockwise.
<input type="checkbox"/>		Scale X	Amount to scale the width of this shape. 1 is scale to 100%.
<input type="checkbox"/>		Scale Y	Amount to scale the height of this shape. 1 is scale to 100%.
<input type="checkbox"/>		Visible	If true, the shape will be visible.
<input type="checkbox"/>		Width	Width of this shape in pixels.
<input type="checkbox"/>		Height	Height of this shape in pixels.
<input checked="" type="checkbox"/>	72	X	Horizontal distance in pixels between the left edge of this shape and the left edge of the page.
<input type="checkbox"/>	72	Y	Vertical distance in pixels between the upper edge of this shape and the top edge of the page.
<input checked="" type="checkbox"/>			
<input type="checkbox"/>	0		
<input type="checkbox"/>	1		
<input type="checkbox"/>	1		
<input checked="" type="checkbox"/>			
<input type="checkbox"/>	540		
<input type="checkbox"/>	18		
<input type="checkbox"/>	0		
<input type="checkbox"/>	34		

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 here in the Appendix.

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

Expression Functions

The expression language is used to define dynamic values for component properties and expression tags. Expressions often involve one or more other values that are used to calculate a final value. Expressions don't do anything, other than return a value.

For an overview and syntax of the expression language, see [Expression Language and Syntax](#).

In this section, we cover all of the built in expression functions available inside Ignition. Each page will have a banner at the top that looks like this:

This function is used by **Ignition's Expression** language.

This lets you know that you are looking at a function for the expression language.

Advanced	Identity Provider	Translation
Aggregates	JSON	Type Casting
Alarming	Logic	Users
Colors	Math	
Date and Time	String	

Advanced

Advanced Functions

The following functions allow you to interact with Ignition in more advanced ways through expression bindings.

[In This Section ...](#)

columnRearrange

This function is used by Ignition's Expression language.

Description

Returns a view of the given dataset with the given columns in the order specified. Columns may be omitted in order to filter out columns from the original dataset.

Syntax

columnRearrange(dataset, [col, ...])

- Parameters

Dataset dataset - The starting dataset.

String col - Any number of column names, in the order that they should appear. The columns specified must match columns in the original dataset. [optional]

- Results

Dataset - A new dataset with columns in the order specified.

Examples

Code Snippet

```
columnRearrange(fiveColDataset, "secondCol", "thirdCol", "firstCol") // Returns a 3 column Dataset, where the columns are in the given order.
```

columnRename

This function is used by Ignition's Expression language.

Description

Returns a view of the given dataset with the columns renamed. The number of new names must match exactly with the existing column count.

Syntax

columnRename(dataset, [newName, ...])

- Parameters

Dataset dataset - The starting dataset.

String newName - Any number of new column names. The columns specified must match the number of columns in the original dataset. [optional]

- Results

Dataset - A new dataset with new column names.

Examples

Code Snippet

```
columnRename(twoColDataset, "colOne", "colTwo") // Returns a dataset with columns ["colOne", "colTwo"].
```

forceQuality

This function is used by Ignition's Expression language.

Description

Returns the given value, but overwrites the quality of that value. See the [quality codes](#) for a list of potential quality codes. If the quality argument is omitted, the quality will be GOOD (192). This is a way to have expressions opt-out of the quality overlay system. You can also force a specific quality code here by including the quality argument.

Syntax

forceQuality(value, [qualityCode])

- Parameters

Object value - The value to force a quality on.

Integer qualityCode - The qualityCode to force on the value. [optional]

- Results

Object - The value with a forced quality.

Examples

Code Snippet

```
forceQuality({Tanks/Tank15}) //Returns the value of the Tank15 Tag, but always with a good quality code.
```

Code Snippet

```
forceQuality({Tanks/Tank15}, 410) //Returns the value of the Tank15 Tag, but always with a TAG_DISABLED quality.
```

property

This function is used by Ignition's Expression language.

Description

Returns an object representing the value of the property at the path specified. It takes a single string as an argument and attempts to lookup the property value at the specified path. What makes this function useful is that the path itself can be the result of an expression, meaning it can be dynamic. Normally, you'd use the expression language's built-in bound-value [syntax](#) to use a property value in an expression.



Perspective only

The property() expression function is only accessible in Perspective.

Syntax

property(propertyPath)

- Parameters

[String](#) propertyPath - The property path to the property.

- Results

[Object](#) - The value of the property.

Examples

Code Snippet

```
property("this.custom." + {view.params.ControlType})
```

qualifiedValue

This function is used by Ignition's Expression language.

Description

Returns the given value, but overwrites the quality of that value. Provides more control over the value than the [forceQuality](#) expression.

Syntax

qualifiedValue(value, level, [subcode], [diagnosticMessage])

- Parameters

Object value - The value to force a quality on.

Object level - The level to force on the value. Possible levels are Good or 0, Uncertain or 1, Bad or 2, Error or 3.

Object subcode - The subcode to include with the quality level. See [Tag Quality and Overlays](#) for more information on possible subcodes. [optional]

Object diagnosticMessage - The a diagnostic message to add to the quality. [optional]

- Results

Object - The value with a forced quality.

Examples

Code Snippet

```
qualifiedValue(1, 'bad', 515, 'New Quality') //Returns the value 1 but with a quality of Bad_Disabled("New Quality").
```

runScript

This function is used by Ignition's Expression language.

Description

Runs a single line of Python code as an expression. If a poll rate is specified, the function will be run repeatedly at the poll rate. This is a very powerful way for you to add extensions to the expression language. For example, one could write a project script module function called shared.weather.getTempAt(zip) that queried a web service for the current temperature at a given zip code, and then bind the value of a label to the return value of that function.

The **scriptFunction** is entered as a **string** and the **pollRate** is in **milliseconds**. You can optionally add any function arguments after the poll rate.

runScript Polling in Tags

The runScript function can be used in expression tags, but the poll rate doesn't work exactly the same as in an expression binding. All Tags have a Scan Class that dictates the minimum amount of time between each evaluation. The runScript poll rate only polls **up to** the rate of the scan class set on the tag.

For example, if an Expression Tag is configured with runScript to run at a poll rate of 60 seconds and is using the "default" (1 second) scan class, the Tag's Expression will still execute every 1 second. So a scan class rate of 60 seconds will be necessary for a runScript expression to poll at any rate between 0 and 60 seconds.

Syntax - Preferred

runScript(scriptFunction, [pollRate], [arg1], [arg2], [arg...])

- Parameters

String `scriptFunction` - A single line of python code. Typically the path to a script module.

Integer `pollRate` - The poll rate of the script in milliseconds. [optional]

Object `arg` - Any number of argument objects that will be passed into the given script. This expression function can't make use of keyword invocation, so the order of the arguments passed to runScript represents how the parameters will be passed to the underlying Python function. [optional]

- Results

Object - The return value of the specified function.

Syntax - Legacy

runScript(scriptFunction, [pollRate])

- Parameters

String `scriptFunction` - A string representing a single line of code, including any arguments that will be passed to the function.

Integer `pollRate` - The poll rate of the script in milliseconds. [optional]

- Results

Object - The return value of the specified function.

Examples

Here is our scripting function we are going to run that is located in a Project Library script called **textScript**. The project the script was in was also set as the [Gateway Scripting Project](#).

Code Snippet - Python Function

```
def myFunc(text="Hello World!", moreText="Good bye"):
    return text
```

General Usage

```
// This code block shows how to use runScript without additional parameters.  
  
// Preferred syntax.  
runScript("textScript.myFunc")  
  
// Legacy syntax.  
runScript("textScript.myFunc()")
```

Passing Arguments

```
// Preferred syntax.  
runScript("textScript.myFunc", 0, "Hello Again", "See ya later")  
  
// Legacy syntax.  
runScript("textScript.myFunc('Hello Again', 'See ya later')", 0)
```

Example - Legacy Usage

```
// Legacy syntax using string concatenation.  
runScript("textScript.myFunc('" +{_gensim_/Writeable/WriteableString1} + "')") // This would run the function  
and pass in the value of the WriteableString1 tag.
```

sortDataset

This function is used by Ignition's Expression language.

Description

Takes a dataset and returns a sorted version of dataset. The sort order is determined by a single column. This works on numeric, as well as alphanumeric columns. When sorting alphanumerically, contiguous numbers are treated as a single number: you may recognize this as a "natural sort".

Sort Order

The table below represents an example of how alphanumeric values are sorted by the function (assuming a natural sort). Where **Raw Column Values** represents an initial set of values, and the Sorted columns show how the function sorts in **Ascending** and **Descending** order.

Raw Column Values	Sorted - Ascending	Sorted - Descending
a1	a1	Z3
a22	A1	z3
Z3	a4	a77z99
z3	a7z9	a77z4
a4	a22	a22
a77z4	a77z4	a7z9
a77z99	a77z99	a4
a7z9	z3	a1
A1	Z3	A1

Some caveats to be aware of:

- Null values for string columns are sorted first
- Null values for numeric columns are sorted last
- Casing is not used as a method of sorting. If the only difference between two cells is the casing, then the resulting order depends largely on where the cells were in the raw column.

Syntax

`sortDataset(dataset, colIndex, [ascending], [naturalOrdering])`

- Parameters

`Dataset` dataset - The starting dataset.

`Integer` colIndex - The index of the column to sort on.

`Boolean` ascending - A flag indicating whether or not to sort ascending. Defaults to true. [optional]

`Boolean` naturalOrdering - A flag indicating the ordering method. True for natural, false for alphabetical. Defaults to true. [optional]

- Results

`Dataset` - A sorted dataset

Syntax

`sortDataset(dataset, colName, [ascending], [naturalOrdering])`

- Parameters

`Dataset` dataset - The starting dataset.

`String` colName - The name of the column to sort on.

`Boolean ascending` - A flag indicating whether or not to sort ascending. Defaults to true. [optional]

`Boolean naturalOrdering` - A flag indicating the ordering method. True for natural, false for alphabetical. Defaults to true. [optional]

- Results

`Dataset` - A sorted dataset.

Examples

Code Snippet

```
sortDataset(dataset, 0, true) // Returns a dataset sorted ascending on column 0.
```

Code Snippet

```
sortDataset(dataset, "Column 1", false) // Returns a dataset sorted descending on the column named "Column 1".
```

tag

This function is used by [Ignition's Expression language](#).

Description

Returns an object representing the value of the Tag at the path specified. Normally, you'd use the expression language's built-in bound-value [syntax](#) to use a Tag value in an expression. What makes this function useful is that the path itself can be the result of an expression, meaning it can be dynamic. The object returned by the function may need to be converted to a standard data type. Check out the [Type Casting](#) functions for more information.

 When using the tag() function in a logic function, the Tag value will remain subscribed to, even if the logic function chooses a different outcome. This can affect Tags that are on a leased scan class.

Syntax

tag(tagPath)

- Parameters

[String tagPath](#) - The Tag path to the Tag.

- Results

[Object](#) - The value of the Tag. The object returned by the function may need to be converted to a standard data type using one of the various [Type Casting](#) functions.

Examples

Code Snippet

```
tag("Tanks/Tank5") //Returns Tank5's value.
```

Code Snippet

```
tag("Tanks/Tank" + {Root Container.TankNum}) //Returns the value for the tank represented by the dynamic property TankNum on the root container.
```

typeOf

This function is used by Ignition's Expression language.

This feature is new in Ignition version **8.1.4**.
[Click here](#) to check out the other new features

Description

Returns the simple name of the Java type.

Syntax

typeOf(value)

- Parameters
 - [Any](#) value - The object for which you want to find the Java type.
- Results
 - [String](#) - Returns the simple name of the Java type.

Examples

Code Snippet

```
// Takes in a string value, returns "String".  
typeOf("My String")
```

Code Snippet

```
// Takes in a Tag's value using the Tag expression function. Returns a string representing the datatype of  
the Tag.  
typeOf(tag("[default]WriteableInteger1"))
```

Code Snippet

```
// Takes in a color type property from a Vision component. Returns "ColorUIResource".  
typeOf({Root Container.Label.foreground})
```

Aggregates

Aggregate Functions

The following functions allow you to fetch aggregated values from datasets in expression bindings.

[In This Section ...](#)

groupConcat

This function is used by Ignition's Expression language.

Description

Concatenates all of the values in the given column of the given dataset into a string, with each value separated by the string separator. Any null values in the column are ignored.

Syntax

groupConcat(dataset, columnIndex, separator)

- Parameters

Dataset dataset - The starting dataset.

Integer columnIndex - The index of the column to concatenate.

String separator - What will be used to separate each of the values.

- Returns

String - A string with every value in the specified column of the specified dataset separated by the separator value.

Syntax

groupConcat(dataset, columnName, separator)

- Parameters

Dataset dataset - The starting dataset.

String columnName - The name of the column to concatenate.

String separator - What will be used to separate each of the values.

- Returns

String - A string with every value in the specified column of the specified dataset separated by the separator value.

Examples

Suppose you had a table with this dataset in it:

Product Code	Quality	Weight
BAN_002	380	3.243
BAN_010	120	9.928
APL_000	125	1.287
FWL_220	322	7.889

Code Snippet

```
groupConcat({Root Container.Table.data}, 1, " / ") //Would return the string: "380 / 120 / 125 / 322".
```

Code Snippet

```
groupConcat({Root Container.Table.data}, "ProductCode", " ", " ") //Would return the string: "BAN_002, BAN_010, APL_000, FWL_220".
```


max

This function is used by Ignition's Expression language.

Description

Finds and returns the maximum value in the given column of the given dataset, or the max value in a series of numbers specified as arguments. When looking up the max in a dataset, the column may be specified as an index or as a column name.

This function expects the data type of the column to be **numeric**: other data types, such as strings, will throw an exception.

Any null values in the column are ignored. If there are no rows in the dataset, zero is returned.

Syntax

max(dataset, columnIndex)

- Parameters

Dataset dataset - The dataset to search through.

Integer columnIndex - The index of the column to search through. Must be a column index of the provided dataset. Additionally, the data type of the column must be numeric.

- Returns

Integer - The maximum value in that column.

Syntax

max(dataset, columnName)

- Parameters

Dataset dataset - The dataset to search through.

String columnName - The name of the column to search through. Must match a column name in the provided dataset. Additionally, the data type of the column must be numeric.

- Returns

Integer - The maximum value in that column.

Syntax

max(value, [value, ...])

- Parameters

Integer/Float value - A number. Can be as many values as needed. Can be either a float or an integer.

- Returns

Integer - The maximum value in the list of values.

Examples

For example, suppose you had a table with the following dataset in it:

ProductCode	Quantity	Weight
BAN_002	380	3.243
BAN_010	120	9.928
APL_000	125	1.287

Code Snippet

```
max({Root Container.Table.data}, 1) //Would return 380.
```

Code Snippet

max(0, 10/2, 3.14) //Would return 5. You can also use this function to find the maximum in fixed series of numbers, specified as arguments.

Code Snippet

```
max({SomeValue}, 0) //The following example is a great way to make sure a value never goes below zero.
```

maxDate

This function is used by Ignition's Expression language.

Description

Finds and returns the maximum date in the given column of the given dataset, or the max value in a series of dates specified as arguments. When looking up the max date in a dataset, the column may be specified as an index or as a column name. Any null values in the column are ignored. If there are no rows in the dataset, null is returned.

Syntax

maxDate(dataset, columnIndex)

- Parameter

Dataset dataset - The starting dataset to search.

Integer columnIndex - The index of the column to search for the max date. Must be a column index of the provided dataset.

- Returns

Date - The maximum date of the given date column in the given dataset.

Syntax

maxDate(dataset, columnName)

- Parameter

Dataset dataset - The starting dataset to search.

String columnName - The name of the column to search for the max date. Must match a column name in the provided dataset.

- Returns

Date - The maximum date of the given date column in the given dataset.

Syntax

maxDate(date, [date])

- Parameter

Date date - A date. Can be as many dates as needed.

- Returns

Date - The maximum date of the given dates.

Examples

The following table applies to the code snippet below:

AlarmTime	Path	Severity
2010-01-08 7:28:04	Tanks/Tank5/TempHiAlarm	4
2010-01-08 10:13:22	Tanks/Tank38/LoLevel	2
2010-01-08 13:02:56	Valves/Valve2/	2

Code Snippet

```
maxDate({Root Container.Table.data}, "AlarmTime") //You could use this expression to get the date and time  
for the most recent alarm.
```

Code Snippet

```
maxDate(now(0), addMinutes(now(0), 5)) // This would return the Date that is 5 minutes from now.
```

mean

This function is used by Ignition's Expression language.

Description

Calculates the mean (a.k.a average) for the numbers in the given column of the given dataset or the mean of a series of numbers specified as arguments. When looking up the mean in a dataset, the column may be specified as an index or as a column name. Any null values in the column are ignored. If there are no rows in the dataset, zero is returned.

Syntax

mean(dataset, columnIndex)

- Parameters

[Dataset](#) dataset - The dataset to use.

[Integer](#) columnIndex - The index of the column to use. Must be a column index of the provided dataset.

- Returns

[Integer/Float](#) - The mean of the values in that column.

Syntax

mean(dataset, columnName)

- Parameters

[Dataset](#) dataset - The dataset to use.

[String](#) columnName - The name of the column to search through. Must match a column name in the provided dataset.

- Returns

[Integer/Float](#) - The mean of the values in that column.

Syntax

mean(value, [value, ...])

- Parameters

[Integer/Float](#) value - A number. Can be as many values as needed. Can be either a float or an integer.

- Returns

[Integer/Float](#) - The mean of the values.

Examples

For example, suppose you had a table with this dataset in it:

ProductCode	Quantity	Weight
BAN_002	380	3.243
BAN_010	120	9.928
APL_000	125	1.287
FWL_220	322	7.889

Code Snippet

```
mean({Root Container.Table.data}, "Weight") //... would return 5.58675.
```

Code Snippet

```
mean(1,2,3) //... would return 2.
```

median

This function is used by Ignition's Expression language.

Description

Calculates the median for the numbers in the given column of the given dataset or the median of a series of numbers specified as arguments. When looking up the median in a dataset, the column may be specified as an index or as a column name. Any null values in the column are ignored. If there are no rows in the dataset, zero is returned.

Syntax

median(dataset, columnIndex)

- Parameters

Dataset dataset - The dataset to search through.

Integer columnIndex - The index of the column to search through. Must be a column index of the provided dataset.

- Returns

Integer/Float - The median value in that column.

Syntax

median(dataset, columnName)

- Parameters

Data et dataset - The dataset to search through.

String columnName - The name of the column to search through. Must match a column name in the provided dataset.

- Returns

Integer/Float - The median value in that column.

Syntax

median(value, [value, ...])

- Parameters

Integer/Float value - A number. Can be as many values as needed. Can be either a float or an integer.

- Returns

Integer/Float - The median value in the list of values.

Examples

For example, suppose you had a table with this dataset in it:

ProductCode	Quantity	Weight
BAN_002	380	3.243
BAN_010	120	9.928
APL_000	125	1.287
FWL_220	322	7.889

Code Snippet

```
median({Root Container.Table.data}, "Weight") //... would return 5.566.
```

```
median(1,2,3,3,10) //... would return 3.
```

min

This function is used by Ignition's Expression language.

Description

Finds and returns the minimum value in the given column of the given dataset, or the min value in a series of numbers specified as arguments. When looking up the min in a dataset, the column may be specified as an index or as a column name. Any null values in the column are ignored. If there are no rows in the dataset, zero is returned.

Syntax

min(dataset, columnIndex)

- Parameters

Dataset dataset - The dataset to search through.

Integer columnIndex - The index of the column to search through. Must be a column index of the provided dataset.

- Returns

Integer - The minimum value in that column.

Syntax

min(dataset, columnName)

- Parameters

Dataset dataset - The dataset to search through.

String columnName - The name of the column to search through. Must match a column name in the provided dataset.

- Returns

Integer - The minimum value in that column.

Syntax

min(value, [value, ...])

- Parameters

Integer /Float value - A number. Can be as many values as needed. Can be either a float or an integer.

- Returns

Integer - The minimum value in the list of values.

Examples

For example, suppose you had a table with this dataset in it:

ProductCode	Quantity	Weight
BAN_002	380	3.243
BAN_010	120	9.928
APL_000	125	1.287
FWL_220	322	7.889

Code Snippet

```
min({Root Container.Table.data}, 1) //... would return 120.
```

Code Snippet

```
min(0, 10/2, 3.14) //... would return 0.
```

Code Snippet

```
min({SomeValue}, 180) //This example is a great way to make sure a value never goes above 180.
```

minDate

This function is used by Ignition's Expression language.

Description

Finds and returns the minimum date in the given column of the given dataset, or the min value in a series of dates specified as arguments. When looking up the min date in a dataset, the column may be specified as an index or as a column name. Any null values in the column are ignored. If there are no rows in the dataset, null is returned.

Syntax

minDate(dataset, columnIndex)

- Parameter

Dataset dataset - The starting dataset to search.

Integer columnIndex - The index of the column to search for the max date. Must be a column index of the provided dataset.

- Returns

Date - The minimum date of the given date column in the given dataset.

Syntax

minDate(dataset, columnName)

- Parameter

Dataset dataset - The starting dataset to search.

String columnName - The name of the column to search for the max date. Must match a column name in the provided dataset.

- Returns

Date - The minimum date of the given date column in the given dataset.

Syntax

minDate(date, [date, ...])

- Parameter

Date date - A date. Can be as many dates as needed.

- Returns

Date - The minimum date of the given dates.

Examples

For example, suppose you had a table with this dataset in it:

AlarmTime	Path	Severity
2010-01-08 7:28:04	Tanks/Tank5/TempHiAlarm	4
2010-01-08 10:13:22	Tanks/Tank38/LoLevel	2
2010-01-08 13:02:56	Valves/Valve2/	2

```
minDate({Root Container.Table.data}, "AlarmTime") //You could use this expression to get the date and time for the oldest alarm.
```


stdDev

This function is used by Ignition's Expression language.

Description

Calculates the simple standard deviation of the values in the given column of the given dataset, or the standard deviation for a series of numbers specified as arguments. When looking up the standard deviation in a dataset, the column may be specified as an index or as a column name. Any null values in the column are ignored. If there are no rows in the dataset, zero is returned.

Syntax

stdDev(dataset, columnIndex)

- Parameters

Dataset dataset - The dataset to search through.

Integer columnIndex - The index of the column to search through. Must be a column index of the provided dataset.

- Returns

Integer/Float - The standard deviation of the values in that column.

Syntax

stdDev(dataset, columnName)

- Parameters

Dataset dataset - The dataset to search through.

String columnName - The name of the column to search through. Must match a column name in the provided dataset.

- Returns

Integer/Float - The standard deviation of the values in that column.

Syntax

stdDev(value, [value, ...])

- Parameters

Integer/Float value - A number. Can be as many values as needed. Can be either a float or an integer.

- Returns

Integer/Float - The standard deviation of the values in the list of values.

Examples

For example, suppose you had a table with this dataset in it:

ProductCode	Quantity	Weight
BAN_002	380	3.243
BAN_010	120	9.928
APL_000	125	1.287
FWL_220	322	7.889

Code Snippet

```
stdDev({Root Container.Table.data}, "Weight") //... would return 3.4687.
```

sum

This function is used by Ignition's Expression language.

Description

Calculates the sum of the values in the given column of the given dataset, or the sum for a series of numbers specified as arguments. When looking up the sum in a dataset, the column may be specified as an index or as a column name. Any null values in the column are ignored. If there are no rows in the dataset, zero is returned.

Syntax

sum(dataset, columnIndex)

- Parameters

Dataset dataset - The dataset to use.

Integer columnIndex - The index of the column to use. Must be a column index of the provided dataset.

- Returns

Integer/Float - The sum of the values in that column.

Syntax

sum(dataset, columnName)

- Parameters

Dataset dataset - The dataset to use.

String columnName - The name of the column to search through. Must match a column name in the provided dataset.

- Returns

Integer/Float - The sum of the values in that column.

Syntax

sum(value, [value, ...])

- Parameters

Integer/Float value - A number. Can be as many values as needed. Can be either a float or an integer.

- Returns

Integer/Float - The sum of the values.

Examples

For example, suppose you had a table with this dataset in it.

ProductCode	Quantity	Weight
BAN_002	380	3.243
BAN_010	120	9.928
APL_000	125	1.287
FWL_220	322	7.889

Code Snippet

```
sum({Root Container.Table.data}, 1) //... would return 947.
```

Code Snippet

```
sum(1,2,3) //... would return 6.
```

Alarming Expressions

Alarming Functions

The following functions allow you to view alarm status in expression bindings.

[In This Section ...](#)

isAlarmActive

This function is used by Ignition's Expression language.

Description

Returns whether there are active alarms that match the provided criteria. The alarm name is optional, and both the Tag path and alarm name support wildcards (*). For example, if only the Tag path was specified, this function would return whether any alarm on the Tag was active. The pollRate parameter is only applicable in the Client scope.

When calling this from the Gateway scope, the Tag Provider must be included in the path.

Syntax

isAlarmActive(tagPath, [alarmName], [pollRate])

- Parameters

String tagPath - The Tag path to search for active alarms. Supports the wildcard '*'.

String alarmName - The name of the alarm to search for. Supports the wildcard '*'. [optional]

Integer pollRate - The poll rate in milliseconds. Only applicable in the Client scope. [optional]

- Returns

Boolean - True if an alarm is active; False if no active alarms were found.

Examples

Code Snippet

```
isAlarmActive("[default]Tanks/Temp", "Tank_Temp_High") //When the Tank_Temp_High alarm is active then this expression returns true.
```

isAlarmActiveFiltered

This function is used by Ignition's Expression language.

Description

Returns whether there are active alarms that match the provided criteria. It is much more granular than isAlarmActive. The Tag path, alarm name, and display path all support wildcards ("*"). The min and max priority expect a number between 0 (diagnostic) and 4 (critical). The pollRate parameter is only applicable in the Client scope and is optional.

When calling this from the Gateway scope, the Tag Provider must be included in the path.

Syntax

`isAlarmActive(tagPath, alarmName, displayPath, minPriority, maxPriority, allowCleared, allowAcked, allowShelved, [pollRate])`

- Parameters

`String tagPath` - The Tag path to search for active alarms. Accepts the wildcard "*".

`String alarmName` - The alarm name to search for active alarms. Accepts the wildcard "*".

`String displayPath` - The display path to search for active alarms. Accepts the wildcard "*".

`Integer minPriority` - The minimum priority of alarms to accept: 0 is Diagnostic, 1 is Low, 2 is Medium, 3 is High, 4 is Critical.

`Integer maxPriority` - The maximum priority of alarms to accept: 0 is Diagnostic, 1 is Low, 2 is Medium, 3 is High, 4 is Critical.

`Boolean allowCleared` - A flag that indicates whether to accept cleared alarms.

`Boolean allowAcked` - A flag that indicates whether to accept acknowledged alarms.

`Boolean allowShelved` - A flag that indicates whether to accept shelved alarms.

`Integer pollRate` - The poll rate of the function in milliseconds. Only applicable in the Client scope. [optional]

- Results

`Boolean` - True if there are active alarms, False if there are not.

Examples

Code Snippet

```
isAlarmActiveFiltered("*, *, *, 0, 4, 0, 1, 0) //When any critical alarm is active, even if  
acknowledged, then this expression returns True.
```

Colors

Color Functions

The following functions allow you to modify or set color values in expression bindings.

[In This Section ...](#)

brighter

This function is used by Ignition's Expression language.

Description

Returns a color that is one shade brighter than the color given as an argument. Note that if you pass in a fully saturated color, like (255,0,0), it cannot be made brighter.

Syntax

brighter(color)

- Parameter
 - Color color - A color to make brighter. Can use the [color](#) function to create a color value.
- Results
 - Color - A color that is one shade brighter than the color passed in.

Examples

Code Snippet

```
brighter(color(100,150,250)) //Returns the color (142,214,255).
```

color

This function is used by Ignition's Expression language.

Description

Creates a color using the given red, green, and blue amounts, which are integers between 0-255. The optional alpha channel to the color controls transparency.

Note: This function was designed to return color objects to Vision bindings, and will not work with Perspective bindings. Instead, Perspective color properties can simply use string hex codes to derive a color from a binding, for example: "#00FF00".

Syntax

color(red, green, blue, [alpha])

- Parameter

Integer red - The intensity of red, between 0 - 255.

Integer green - The intensity of green, between 0 - 255.

Integer blue - The intensity of blue, between 0 - 255.

Integer alpha - The amount of transparency, between 0 - 255. [optional]

- Results

Color - Returns a color with the given RGB value.

Examples

There are no examples associated with this expression function.

darker

This function is used by Ignition's Expression language.

Description

Returns a color that is one shade darker than the color given as an argument.

Syntax

darker(color)

- Parameter
 - Color** color - A color to make darker. Can use the [color](#) function to create a color value.
- Results
 - Color** - A color that is one shade darker than the color passed in.

Examples

```
darker(color(100,150,250)) //Returns the color (70,105,175).
```

gradient

This function is used by Ignition's Expression language.

Description

Calculates a percentage given the three numeric arguments number, low, and high. Uses this percentage to create a color that is a mix between the two colors.

Syntax

`gradient(value, low, high, lowColor, highColor)`

- Parameter

`Integer` value - The value used to determine the percentage between the low and high values.

`Integer` low - The low value to use to calculate the percentage.

`Integer` high - The high value to use to calculate the percentage.

`Color` lowColor - The color that will match 0%.

`Color` highColor - The color that will match 100%.

- Results

`Color` - A color that is a mix of the two given colors based on the percentage.

Examples

code

```
gradient(0, 0, 100, toColor("red"), toColor("blue")) //Returns red.
```

Code Snippet

```
gradient(100, 0, 100, toColor("red"), toColor("blue")) //Returns blue.
```

Code Snippet

```
gradient(60, 0, 100, toColor("red"), toColor("blue")) //Returns a shade of purple.
```

Code Snippet

```
gradient({Root Container.Tank.value}, 0, 100, color(255,0,0), color(0,0,255)) //Returns a gradient from red to blue based on the level of a tank.
```

Date and Time

Date and Time Functions

The following functions allow you to check or modify time values in expression bindings.

[In This Section ...](#)

add*

This function is used by Ignition's Expression language.

Description

This function is a set of functions that include:

Function	Description
addMillis	Add or subtract an amount of milliseconds to a given date and time.
addSeconds	Add or subtract an amount of seconds to a given date and time.
addMinutes	Add or subtract an amount of minutes to a given date and time.
addHours	Add or subtract an amount of hours to a given date and time.
addDays	Add or subtract an amount of days to a given date and time.
addWeeks	Add or subtract an amount of weeks to a given date and time.
addMonths	Add or subtract an amount of months to a given date and time. This function is unique since each month can have a variable number of days. For example, if the date passed in is March 31st, and we add one month, April does not have a 31st day, so the returned date will be the proper number of months rounded down to the closest available day, in this case April 30th.
addYears	Add or subtract an amount of years to a given date and time.

Syntax

add*(date, value)

- Parameters

Date date - The starting date.

Integer value - The amount of units to change the date by, where the units is dependent on the function used.

- Results

Date - A new date that has been changed by the amount specified.

Code Examples

Code Snippet

```
addWeeks(now(), 2) //Adds 2 weeks to the current time.
```

Code Snippet

```
addDays(now(), -5) //Subtracts 5 days from the current time.
```

Code Snippet

```
addHours({Root Container.Calendar.date}, 5) //This example would add 5 hours to the date passed in from a calendar component.
```

dateArithmetic

This function is used by Ignition's Expression language.

Description

Adds or subtracts some amount of time from a date, returning the resulting date. The field argument must be a string, and must be one of these options:

- ms
- second
- sec
- minute
- hour
- hr
- day
- week
- month
- year

Syntax

dateArithmetic(date, value, field)

- Parameter

Date date - The starting date.

Integer value - The value to add or subtract from the given date.

String field - The units of the value.

- Results

Date - A new date, that has been altered by the amount of units specified.

Examples

Code Snippet

```
dateArithmetric(toDate("2010-01-04 8:00:00"), 5, "hour") //Returns the date '2010-01-04 13:00:00'.
```

Code Snippet

```
dateArithmetric({Root Container.DatePicker.date}, -8, "days") //Returns a date eight days before the date in a  
Popup Calendar component.
```

dateDiff

This function is used by Ignition's Expression language.

Description

Calculates the difference between the two dates, returning the result as a floating point value in the units specified by field, which must be a string matching one of these values:

- ms
- second
- sec
- minute
- min
- hour
- hr
- day
- week
- month
- year

The return value will be a floating point value, meaning that parts of units are considered. The exception to this rule is for the months and years fields, which will always return an integral difference. If the second date argument is after the first, the return value will be positive, otherwise it will be negative.

Syntax

dateDiff(date1, date2, field)

- Parameter

Date date1 - The first date.

Date date2 - The second date.

String field - The units that the difference will be specified in.

- Results

Integer/Float - The difference between two dates in the units specified.

Examples

Code Snippet

```
dateDiff(toDate("2008-2-24 8:00:00"), toDate("2008-2-24 8:15:30"), "minute") //Returns 15.5.
```

Code Snippet

```
dateDiff(toDate("2008-2-24 8:00:00"), toDate("2008-3-12 9:28:00"), "month") //Returns 1.
```

Code Snippet

```
dateDiff(toDate("2008-2-24 8:00:00"), toDate("2008-3-12 9:28:00"), "day") //returns 17.02
```

dateExtract

This function is used by Ignition's Expression language.

Description

Returns an integer value that is the value of the specified date field within the given date. The field must be a string, and must match one of these values:

- ms
- second
- sec
- minute
- min
- hour
- hr
- day
- week
- month
- year
- dayofweek
- dayofyear

Note: Months are returned zero-indexed. That is, January is month 0, February is month 1, and so on. To get a month index starting at 1, simply add 1 to the function result.

Syntax

dateExtract(date, field)

- Parameters

Date date - The given date.

String field - The field to extract the value from.

- Results

Integer - The value of the specified field within the given date.

Examples

Code Snippet

```
dateExtract(toDate("2003-9-14 8:00:00"), "year") //Returns 2003.
```

Code Snippet

```
dateExtract(toDate("2009-1-15 8:00:00"), "month") //Returns 0.
```

Code Snippet

```
dateExtract(toDate("2008-1-24 8:00:00"), "month") + 1 //Returns 1.
```

dateFormat

This function is used by **Ignition's Expression** language.

Description

Returns the given date as a string, formatted according to a pattern. The pattern is a format that is full of various placeholders that will display different parts of the date. These are case-sensitive! These placeholders can be repeated for a different effect. For example, M will give you 1-12, MM will give you 01-12, MMM will give you Jan-Dec, MMMM will give you January-December.

Refer to [Data Type Formatting Reference](#) for a table of the placeholders.



Expert Tip: This function uses the Java class `java.text.SimpleDateFormat` internally, and will accept any valid format string for that class.

Syntax

dateFormat(date, pattern)

- Parameter

`Date` date - The starting date.

`String` pattern - The pattern to format the given date to.

- Results

`String` - The given date formatted based on the given format pattern.

Examples

Code Snippet

```
dateFormat(toDate("2003-9-14 8:00:00"), "yyyy-MM-dd HH:mm:ss") //Returns the string "2003-09-14 08:00:00"  
This format is accepted in most databases.
```

Code Snippet

```
dateFormat(toDate("2003-9-14 8:00:00"), "yyyy-MM-dd h a") //Returns the string "2003-09-14 8 AM".
```

Code Snippet

```
dateFormat(toDate("2003-9-14 8:00:00"), "MMM d, yyyy") //Returns the string "Sep 14, 2003".
```

Code Snippet

```
dateFormat(now(), 'yyyy-MM-dd 00:00:00') //Returns the current date, but forces the time to 00:00:00.
```

dateIsAfter

This function is used by Ignition's Expression language.

Description

Compares two dates to see if date1 is after date2. This is *exclusive*, meaning if the dates are identical the result is always false.

Syntax

dateIsAfter(date1, date2)

- Parameters

Date date1 - The first date to compare.

Date date2 - The second date to compare.

- Results

Boolean - True if date1 is at or after date2; False if not.

Code Examples

Code Snippet

```
dateIsAfter(now(), toDate("2016-04-12 00:00:00"))
// Will be true if the current time is after April 12th, 2016 at midnight.
```

dateIsBefore

This function is used by Ignition's Expression language.

Description

Compares two dates to see if date1 is before date2. This is *exclusive*, meaning if the dates are identical the result is always false.

Syntax

dateIsBefore(date1, date2)

- Parameter

[Date](#) date1 - The first date to compare.

[Date](#) date2 - The second date to compare.

- Results

[Boolean](#) - True if date1 is at or before date2; False if not.

Code Examples

Code Snippet

```
dateIsBefore(now(), toDate("2016-04-12 00:00:00"))
// Will be true if the current time is before April 12th, 2016 at midnight.
```

dateIsBetween

This function is used by Ignition's Expression language.

Description

Compares two dates to see if a target date is between two other dates. This is *inclusive*, meaning if the targetDate is the same as the start or end date the result is true.

Syntax

dateIsBetween(targetDate, startDate, endDate)

- Parameters

Date targetDate - The date to compare.

Date startDate - The start of a date range.

Date endDate - The end of a date range. This date must be after the start date.

- Results

Boolean - True if the targetDate is at or between the startDate and endDate; false if not.

Code Examples

Code Snippet

```
dateIsBetween(now(), toDate("2016-06-12 00:00:00"), toDate("2016-06-19 00:00:00"))
// Will be true if the current time is between the 12th and 19th of June 2016.
```

dateIsDaylight

This function is used by Ignition's Expression language.

Description

Checks to see if the current time zone is using Daylight Saving Time during the date specified. Will use the current date if no date is specified.

Syntax

dateIsDaylight([date])

- Parameters

Date date - The date to use. Will use the current date if omitted. [optional]

- Results

Boolean - True if the current time zone is using Daylight Saving Time during the specified date.

Code Examples

Code Snippet

```
dateIsDaylight(toDate("2007-06-28 00:00:00")) // Will return True in the US/Pacific Timezone, due to that  
time zone's observation of Daylight Saving Time on the specified date.
```

*Between

This function is used by Ignition's Expression language.

Description

This function is a set of functions that include:

Function	Description
millisBetween	Calculates the number of whole milliseconds between two dates.
secondsBetween	Calculates the number of whole seconds between two dates.
minutesBetween	Calculates the number of whole minutes between two dates.
hoursBetween	Calculates the number of whole hours between two dates.
daysBetween	Calculates the number of whole days between two dates. Daylight savings changes are taken into account.
weeksBetween	Calculates the number of whole weeks between two dates.
monthsBetween	Calculates the number of whole months between two dates. Daylight savings changes are taken into account.
yearsBetween	Calculates the number of whole years between two dates. Daylight savings changes are taken into account.

Order does matter for the two dates passed in that we are calculating how much time has passed from date 1 to date 2. So, if date 2 is further in time than date 1, then a positive amount of time has passed. If date 2 is backwards in time from date 1, then a negative amount of time has passed.

Syntax

***between(date1, date2)**

- Parameter

Date date1 - The first date to compare.

Date date2 - The second date to compare.

- Results

Integer- The number of units between the two dates. The units is specified by the function used.

Code Examples

Code Snippet

```
daysBetween(toDate("2017-04-28 00:00:00"), toDate("2017-03-22 00:00:00")) //This will print -37.
```

Code Snippet

```
weeksBetween({Root Container.Calendar1.date}, {Root Container.Calendar2.date}) //Will grab the number of weeks between two dates of calendar components.
```

fromMillis

This function is used by Ignition's Expression language.

Description

Creates a date object given a time, in milliseconds, past Unix epoch (1 January 1970 at midnight UTC).

Syntax

fromMillis(milliseconds)

- Parameters

Integer millis - The number of milliseconds since epoch time.

- Results

Date - The date representing the given number of milliseconds since epoch time.

Code Examples

Code Snippet

```
fromMillis(1503092125000)//This example will print out the date "Fri Aug 18 14:35:25 PDT 2017".
```

get*

This function is used by Ignition's Expression language.

Description

This function is a set of functions that include:

Function	Description
getMillis	Extracts the milliseconds from a date, ranging from 0-999.
getSecond	Extracts the second from a date, ranging from 0-59.
getMinute	Extracts the minutes from a date, ranging from 0-59.
getHour12	Extracts the hour from a date. Uses a 12 hour clock, so noon and midnight are returned as 0.
getHour24	Extracts the hour from a date. Uses a 24 hour clock, so midnight is zero.
getDayOfWeek	Extracts the day of the week from a date. Sunday is day 1, Saturday is day 7.
getDayOfMonth	Extracts the day of the month from a date. The first day of the month is day 1.
getDayOfYear	Extracts the day of the year from a date. The first day of the year is day 1.
getMonth	Extracts the month from a date, where January is month 0.
getQuarter	Extracts the quarter from a date, ranging from 1-4.
getYear	Extracts the year from a date.
getAMorPM	Returns a 0 if the time is before noon, and a 1 if the time is equal to or after noon.

Syntax

get*(date)

- Parameters

Date date - The date to extract from.

- Results

Integer - The value of the units of the date specified. The units are determined by the function used.

Code Examples

Code Snippet

```
getMonth(now()) //This returns the current month.
```

Code Snippet

```
getQuarter(getDate(2017, 3, 15)) //The date, April 15th, is in the second quarter, so this returns 2.
```

Code Snippet

```
getDayOfWeek({Root Container.Calendar.date}) //Will return the day of the week of the selected date of the calendar component.
```

getDate

This function is used by Ignition's Expression language.

Description

Creates a new date object given a year, month and a day. The time will be set to midnight of that day. January is 0 and December is 11. The first day of the month is 1.

Syntax

getDate(year, month, day)

- Parameters

Integer year - The year that the date will be set to.

Integer month - The month that the date will be set to.

Integer day - The day that the date will be set to.

- Results

Date - The date created from the specified integers.

Code Examples

Code Snippet

```
getDate(2016, 11, 1) //This example will create a new date object set to December 1st, 2016.
```

getTimezone

This function is used by Ignition's Expression language.

Description

Returns the ID of the current timezone depending on the scope in which it is called. If run in a Vision client or Perspective session scope, the function will display the timezone of the client or session. If run in an expression in a global scope (e.g. on an expression tag), it will return the timezone of the gateway.

*This list is subject to change depending on the exact version of java that is installed.

Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
Africa/Asmera
Africa/Bamako
Africa/Bangui
Africa/Banjul
Africa/Bissau
Africa/Blantyre
Africa/Brazzaville
Africa/Bujumbura
Africa/Cairo
Africa/Casablanca
Africa/Ceuta
Africa/Conakry
Africa/Dakar
Africa/Dar_es_Salaam
Africa/Djibouti
Africa/Douala
Africa/El_Aaiun
Africa/Freetown
Africa/Gaborone
Africa/Harare
Africa/Johannesburg
Africa/Juba
Africa/Kampala
Africa/Khartoum
Africa/Kigali
Africa/Kinshasa
Africa/Lagos
Africa/Libreville
Africa/Lome
Africa/Luanda
Africa/Lubumbashi
Africa/Lusaka
Africa/Malabo
Africa/Maputo
Africa/Maseru
Africa/Mbabane
Africa/Mogadishu
Africa/Monrovia
Africa/Nairobi
Africa/Ndjamena
Africa/Niamey
Africa/Nouakchott
Africa/Ouagadougou
Africa/Porto-Novo
Africa/Sao_Tome
Africa/Timbuktu
Africa/Tripoli
Africa/Tunis
Africa/Windhoek
America/Adak
America/Anchorage
America/Anguilla
America/Antigua
America/Araguaina
America/Argentina/Buenos_Aires
America/Argentina/Catamarca
America/Argentina/ComodRivadavia
America/Argentina/Cordoba
America/Argentina/Jujuy

America/Argentina/La_Rioja
America/Argentina/Mendoza
America/Argentina/Rio_Gallegos
America/Argentina/Salta
America/Argentina/San_Juan
America/Argentina/San_Luis
America/Argentina/Tucuman
America/Argentina/Ushuaia
America/Aruba
America/Asuncion
America/Atikokan
America/Atka
America/Bahia
America/Bahia_Banderas
America/Barbados
America/Belem
America/Belize
America/Blanc-Sablon
America/Boa_Vista
America/Bogota
America/Boise
America/Buenos_Aires
America/Cambridge_Bay
America/Campo_Grande
America/Cancun
America/Caracas
America/Catamarca
America/Cayenne
America/Cayman
America/Chicago
America/Chihuahua
America/Coral_Harbour
America/Cordoba
America/Costa_Rica
America/Creston
America/Cuiaba
America/Curacao
America/Danmarkshavn
America/Dawson
America/Dawson_Creek
America/Denver
America/Detroit
America/Dominica
America/Edmonton
America/Eirunepe
America/El_Salvador
America/Ensenada
America/Fort_Wayne
America/Fortaleza
America/Glace_Bay
America/Godthab
America/Goose_Bay
America/Grand_Turk
America/Grenada
America/Guadeloupe
America/Guatemala
America/Guayaquil
America/Guyana
America/Halifax
America/Havana
America/Hermosillo
America/Indiana/Indianapolis
America/Indiana/Knox
America/Indiana/Marengo
America/Indiana/Petersburg
America/Indiana/Tell_City
America/Indiana/Vevay
America/Indiana/Vincennes
America/Indiana/Winamac
America/Indianapolis
America/Inuvik
America/Iqaluit
America/Jamaica
America/Jujuy
America/Juneau
America/Kentucky/Louisville
America/Kentucky/Monticello
America/Knox_IN
America/Kralendijk

America/La_Paz
America/Lima
America/Los_Angeles
America/Louisville
America/Lower_Princes
America/Maceio
America/Managua
America/Manaus
America/Marigot
America/Martinique
America/Matamoros
America/Mazatlan
America/Mendoza
America/Menominee
America/Merida
America/Metlakatla
America/Mexico_City
America/Miquelon
America/Moncton
America/Monterrey
America/Montevideo
America/Montreal
America/Montserrat
America/Nassau
America/New_York
America/Nipigon
America/Nome
America/Noronha
America/North_Dakota/Beulah
America/North_Dakota/Center
America/North_Dakota/New_Salem
America/Ojinaga
America/Panama
America/Pangnirtung
America/Paramaribo
America/Phoenix
America/Port-au-Prince
America/Port_of_Spain
America/Porto_Acre
America/Porto_Velho
America/Puerto_Rico
America/Rainy_River
America/Rankin_Inlet
America/Recife
America/Regina
America/Resolute
America/Rio_Branco
America/Rosario
America/Santa_Isabel
America/Santarem
America/Santiago
America/Santo_Domingo
America/Sao_Paulo
America/Scoresbysund
America/Shiprock
America/Sitka
America/St_Barthelemy
America/St_Johns
America/St_Kitts
America/St_Lucia
America/St_Thomas
America/St_Vincent
America/Swift_Current
America/Tegucigalpa
America/Thule
America/Thunder_Bay
America/Tijuana
America/Toronto
America/Tortola
America/Vancouver
America/Virgin
America/Whitehorse
America/Winnipeg
America/Yakutat
America/Yellowknife
Antarctica/Casey
Antarctica/Davis
Antarctica/DumontDUrville
Antarctica/Macquarie

Antarctica/Mawson
Antarctica/McMurdo
Antarctica/Palmer
Antarctica/Rothera
Antarctica/South_Pole
Antarctica/Syowa
Antarctica/Troll
Antarctica/Vostok
Arctic/Longyearbyen
Asia/Aden
Asia/Almaty
Asia/Amman
Asia/Anadyr
Asia/Aqtau
Asia/Aqtobe
Asia/Ashgabat
Asia/Ashkhabad
Asia/Baghdad
Asia/Bahrain
Asia/Baku
Asia/Bangkok
Asia/Beirut
Asia/Bishkek
Asia/Brunei
Asia/Calcutta
Asia/Chita
Asia/Choibalsan
Asia/Chongqing
Asia/Chungking
Asia/Colombo
Asia/Dacca
Asia/Damascus
Asia/Dhaka
Asia/Dili
Asia/Dubai
Asia/Dushanbe
Asia/Gaza
Asia/Harbin
Asia/Hebron
Asia/Ho_Chi_Minh
Asia/Hong_Kong
Asia/Hovd
Asia/Irkutsk
Asia/Istanbul
Asia/Jakarta
Asia/Jayapura
Asia/Jerusalem
Asia/Kabul
Asia/Kamchatka
Asia/Karachi
Asia/Kashgar
Asia/Kathmandu
Asia/Katmandu
Asia/Khandyga
Asia/Kolkata
Asia/Krasnoyarsk
Asia/Kuala_Lumpur
Asia/Kuching
Asia/Kuwait
Asia/Macao
Asia/Macau
Asia/Magadan
Asia/Makassar
Asia/Manila
Asia/Muscat
Asia/Nicosia
Asia/Novokuznetsk
Asia/Novosibirsk
Asia/Omsk
Asia/Oral
Asia/Phnom_Penh
Asia/Pontianak
Asia/Pyongyang
Asia/Qatar
Asia/Qyzylorda
Asia/Rangoon
Asia/Riyadh
Asia/Saigon
Asia/Sakhalin

Asia/Samarkand
Asia/Seoul
Asia/Shanghai
Asia/Singapore
Asia/Srednekolymsk
Asia/Taipei
Asia/Tashkent
Asia/Tbilisi
Asia/Tehran
Asia/Tel_Aviv
Asia/Thimbu
Asia/Thimphu
Asia/Tokyo
Asia/Ujung_Pandang
Asia/Ulaanbaatar
Asia/Ulan_Bator
Asia/Urumqi
Asia/Ust-Nera
Asia/Vientiane
Asia/Vladivostok
Asia/Yakutsk
Asia/Yekaterinburg
Asia/Yerevan
Atlantic/Azores
Atlantic/Bermuda
Atlantic/Canary
Atlantic/Cape_Verde
Atlantic/Faeroe
Atlantic/Faroe
Atlantic/Jan_Mayen
Atlantic/Madeira
Atlantic/Reykjavik
Atlantic/South_Georgia
Atlantic/St_Helena
Atlantic/Stanley
Australia/ACT
Australia/Adelaide
Australia/Brisbane
Australia/Broken_Hill
Australia/Canberra
Australia/Currie
Australia/Darwin
Australia/Eucla
Australia/Hobart
Australia/LHI
Australia/Lindeman
Australia/Lord_Howe
Australia/Melbourne
Australia/NSW
Australia/North
Australia/Perth
Australia/Queensland
Australia/South
Australia/Sydney
Australia/Tasmania
Australia/Victoria
Australia/West
Australia/Yancowinna
Brazil/Acre
Brazil/DeNoronha
Brazil/East
Brazil/West
CET
CST6CDT
Canada/Atlantic
Canada/Central
Canada/East-Saskatchewan
Canada/Eastern
Canada/Mountain
Canada/Newfoundland
Canada/Pacific
Canada/Saskatchewan
Canada/Yukon
Chile/Continental
Chile/EasterIsland
Cuba
EET
EST5EDT
Egypt

Eire
Etc/GMT
Etc/GMT+0
Etc/GMT+1
Etc/GMT+10
Etc/GMT+11
Etc/GMT+12
Etc/GMT+2
Etc/GMT+3
Etc/GMT+4
Etc/GMT+5
Etc/GMT+6
Etc/GMT+7
Etc/GMT+8
Etc/GMT+9
Etc/GMT-0
Etc/GMT-1
Etc/GMT-10
Etc/GMT-11
Etc/GMT-12
Etc/GMT-13
Etc/GMT-14
Etc/GMT-2
Etc/GMT-3
Etc/GMT-4
Etc/GMT-5
Etc/GMT-6
Etc/GMT-7
Etc/GMT-8
Etc/GMT-9
Etc/GMT0
Etc/Greenwich
Etc/UCT
Etc/UTC
Etc/Universal
Etc/Zulu
Europe/Amsterdam
Europe/Andorra
Europe/Athens
Europe/Belfast
Europe/Belgrade
Europe/Berlin
Europe/Bratislava
Europe/Brussels
Europe/Bucharest
Europe/Budapest
Europe/Busingen
Europe/Chisinau
Europe/Copenhagen
Europe/Dublin
Europe/Gibraltar
Europe/Guernsey
Europe/Helsinki
Europe/Isle_of_Man
Europe/Istanbul
Europe/Jersey
Europe/Kaliningrad
Europe/Kiev
Europe/Lisbon
Europe/Ljubljana
Europe/London
Europe/Luxembourg
Europe/Madrid
Europe/Malta
Europe/Mariehamn
Europe/Minsk
Europe/Monaco
Europe/Moscow
Europe/Nicosia
Europe/Oslo
Europe/Paris
Europe/Podgorica
Europe/Prague
Europe/Riga
Europe/Rome
Europe/Samara
Europe/San_Marino
Europe/Sarajevo
Europe/Simferopol

Europe/Skopje
Europe/Sofia
Europe/Stockholm
Europe/Tallinn
Europe/Tirane
Europe/Tiraspol
Europe/Uzhgorod
Europe/Vaduz
Europe/Vatican
Europe/Vienna
Europe/Vilnius
Europe/Volgograd
Europe/Warsaw
Europe/Zagreb
Europe/Zaporozhye
Europe/Zurich
GB
GB-Eire
GMT
GMT0
Greenwich
Hongkong
Iceland
Indian/Antananarivo
Indian/Chagos
Indian/Christmas
Indian/Cocos
Indian/Comoro
Indian/Kerguelen
Indian/Mahe
Indian/Maldives
Indian/Mauritius
Indian/Mayotte
Indian/Reunion
Iran
Israel
Jamaica
Japan
Kwajalein
Libya
MET
MST7MDT
Mexico/BajaNorte
Mexico/BajaSur
Mexico/General
NZ
NZ-CHAT
Navajo
PRC
PST8PDT
Pacific/Apia
Pacific/Auckland
Pacific/Bougainville
Pacific/Chatham
Pacific/Chuuk
Pacific/Easter
Pacific/Efate
Pacific/Enderbury
Pacific/Fakaofa
Pacific/Fiji
Pacific/Funafuti
Pacific/Galapagos
Pacific/Gambier
Pacific/Guadalcanal
Pacific/Guam
Pacific/Honolulu
Pacific/Johnston
Pacific/Kiritimati
Pacific/Kosrae
Pacific/Kwajalein
Pacific/Majuro
Pacific/Marquesas
Pacific/Midway
Pacific/Nauru
Pacific/Niue
Pacific/Norfolk
Pacific/Noumea
Pacific/Pago_Pago
Pacific/Palau

Pacific/Pitcairn
Pacific/Pohnpei
Pacific/Ponape
Pacific/Port_Moresby
Pacific/Rarotonga
Pacific/Saipan
Pacific/Samoa
Pacific/Tahiti
Pacific/Tarawa
Pacific/Tongatapu
Pacific/Truk
Pacific/Wake
Pacific/Wallis
Pacific/Yap
Poland
Portugal
ROK
Singapore
SystemV/AST4
SystemV/AST4ADT
SystemV/CST6
SystemV/CST6CDT
SystemV/EST5
SystemV/EST5EDT
SystemV/HST10
SystemV/MST7
SystemV/MST7MDT
SystemV/PST8
SystemV/PST8PDT
SystemV/YST9
SystemV/YST9YDT
Turkey
UCT
US/Alaska
US/Aleutian
US/Arizona
US/Central
US/East-Indiana
US/Eastern
US/Hawaii
US/Indiana-Starke
US/Michigan
US/Mountain
US/Pacific
US/Pacific-New
US/Samoa
UTC
Universal
W-SU
WET
Zulu
EST
HST
MST
ACT
AET
AGT
ART
AST
BET
BST
CAT
CNT
CST
CTT
EAT
ECT
IET
IST
JST
MIT
NET
NST
PLT
PNT
PRT
PST
SST
VST

Syntax

getTimezone()

- Parameters

Nothing

- Results

String - The current timezone ID.

Code Examples

There are no examples associated with this expression function.

getTimezoneOffset

This function is used by Ignition's Expression language.

Description

Returns the current time zone's offset versus UTC for a given instant, taking Daylight Saving Time into account.

Syntax

getTimezoneOffset([date])

- Parameters

Date date - A specified date to compare the current timezone to UTC. Will use the current time if left blank. [optional]

- Results

Float - The offset of the current time from UTC.

Code Examples

Code Snippet

```
getTimezoneOffset(getDate(2017, 1, 22)) //Returns -8.0, if you are in Pacific Time.
```

Code Snippet

```
getTimezoneOffset(getDate(2017, 6, 22)) //Returns -7.0, if you are in Pacific Time, since Daylight Saving  
Time would be in effect.
```

getTimezoneRawOffset

This function is used by Ignition's Expression language.

Description

Returns the current timezone's offset versus UTC, not taking daylight savings into account.

Syntax

getTimezoneRawOffset()

- Parameters

Nothing

- Results

Float - The offset of the current time from UTC.

Code Examples

Code Snippet

```
getTimezoneRawOffset() //Returns -8.0 if you are in the Pacific Timezone, regardless of time of year.
```

midnight

This function is used by Ignition's Expression language.

Description

Returns a copy of a date with the hour, minute, second, and millisecond fields set to zero.

Syntax

midnight(date)

- Parameters

Date date - The date to set to midnight.

- Results

Date - The new date set to midnight.

Code Examples

Code Snippet

```
midnight(now()) //This will take the current date and set the time to midnight.
```

now

This function is used by Ignition's Expression language.

Description

Returns the current time. The host computer's system clock is used, meaning that if this expression is being evaluated in a running client, the computer running the client's system clock is used.

This function is one of the few expression functions that will poll. If you do not specify a pollRate, it will default to 1,000ms. If you do not want this function to poll, use a poll rate of zero.

Syntax

`now([pollRate])`

- Parameters

`Integer pollRate` - The poll rate in milliseconds to update the time at. Default is 1000 ms. [optional]

- Results

`Date` - The current time.

Examples

Code Snippet

```
now() //Returns the current time, updates every second.
```

Code Snippet

```
now(13000) //Returns the current time, updates every 13 seconds.
```

Code Snippet

```
dateFormat(now(0), "MMM d, h:mm a") //Returns a string representing the current time, formatted like "Feb 12, 9:54 AM". Does not update.
```

setTime

This function is used by Ignition's Expression language.

Description

Takes in a date, and returns a copy of it with the time fields set as specified. Note that the millisecond field is not preserved.

Syntax

setTime(date, hour, minute, second)

- Parameters

Date date - A starting date.

Integer hour - The value to set the hour field to.

Integer minute - The value to set the minute field to.

Integer second - The value to set the second field to.

- Results

Date - The new date with the time set as specified.

Code Examples

Code Snippet

```
setTime({Root Container.Calendar.date}, 1, 37, 44) //This example will set the date object to the current date with the time set to 01:37:44.
```

timeBetween

This function is used by Ignition's Expression language.

Description

Checks to see if the given time is between the start and end times. The given times are expected as strings, and may include dates.

Note: Dates will be parsed according to the default system culture.

Syntax

timeBetween(date, startDate, endDate)

- Parameters

Date/String date - The date to compare. Can be either a date or a string.

Date/String startDate - The start date to compare to. Can be either a date or a string.

Date/String endDate - The end date to compare to. Can be either a date or a string.

- Results

Boolean - True if the date is between the start and end date; false if not.

Examples

Code Snippet

```
timeBetween(toDate("2003-9-14 12:00:00"), toDate("2003-9-14 8:00:00"), toDate("2003-9-14 18:00:00")) //Returns true.
```

Code Snippet

```
timeBetween("2:00:00 pm", "9:00:00 am", "5:00:00 pm") //Returns true.
```

Code Snippet

```
timeBetween(toDate("2003-9-14 20:00:00"), toDate("2003-9-14 18:00:00"), toDate("2003-9-15 2:00:00")) //Returns true.
```

toMillis

This function is used by Ignition's Expression language.

Description

Converts a date object to its millisecond value elapsed since January 1, 1970, 00:00:00 UTC (GMT).

Syntax

toMillis(date)

- Parameters

Date date - The date to convert to epoch time.

- Results

Integer - The number of milliseconds from epoch time of the give date.

Code Examples

Code Snippet

```
// This will take the date Aug 22, 2017 at 14:35:25 PST and convert it to milliseconds from epoch time which  
is 1,500,767,134,000.  
toMillis(setTime(getDate(2017, 6, 22), 16, 45, 34))
```

Identity Provider

Identity Provider Functions

The following functions allow you to test whether specified elements are present in an IdP collection object. They can be used only in the [Security Level Rules](#) and [User Attribute Mapping](#) sections of the Gateway webpage.

[In This Section ...](#)

containsAll

This function is used by [Ignition's Expression language](#).

Description



This function is only available for [Security Level Rules](#) and [User Attribute Mapping](#).

This function checks to see if all of the listed elements are present in the collection object. The function requires at least two arguments, a collection and an element.

Syntax

containsAll(collection, element0, [elementN])

- Parameters

[Object](#) collection - A collection of values. Typically from the {security-zone} object or the {idp-attribute:X} object.

[String](#) element - One or more comma-separated elements to look for.

- Results

[Boolean](#) - True if the collection object contained all of the listed elements; false if otherwise.

Examples

Code Snippet

```
// Returns true for a login attempt against an Ignition IdP, if the user has both Administrator and Operator roles.  
containsAll({attribute-source:idTokenClaims:roles}, 'Administrator', 'Operator')
```

Code Snippet

```
// Returns true for a login attempt if the login location is in all three of the specified security zones.  
containsAll({security-zones}, 'site 1', 'mill', 'offshore')
```

containsAny

This function is used by **Ignition's Expression** language.

Description



This function is only available for [Security Level Rules](#) and [User Attribute Mapping](#).

This function checks to see if any of the listed elements are present in the collection object. The function requires at least two arguments, a collection and an element.

Syntax

containsAny(collection, element0, [elementN])

- Parameters

Object collection - A collection of values. Typically from the {security-zone} object or the {idp-attribute:X} object.

String element - One or more comma separated elements to look for.

- Results

Boolean - True if the collection object contained any of the listed elements; false if otherwise.

Examples

Code Snippet

```
// Returns true for a login attempt against an Ignition IdP, if the user has either the Administrator or
Operator roles.
containsAny({attribute-source:idTokenClaims:roles}, 'Administrator', 'Operator')
```

Code Snippet

```
// Returns true for a login attempt if the login location is in at least one of the specified security zones.
containsAny({security-zones}, 'site 1', 'mill', 'offshore')
```

JSON

JSON Functions

The following functions allow you to manipulate JSON strings in expression bindings.

[In This Section ...](#)

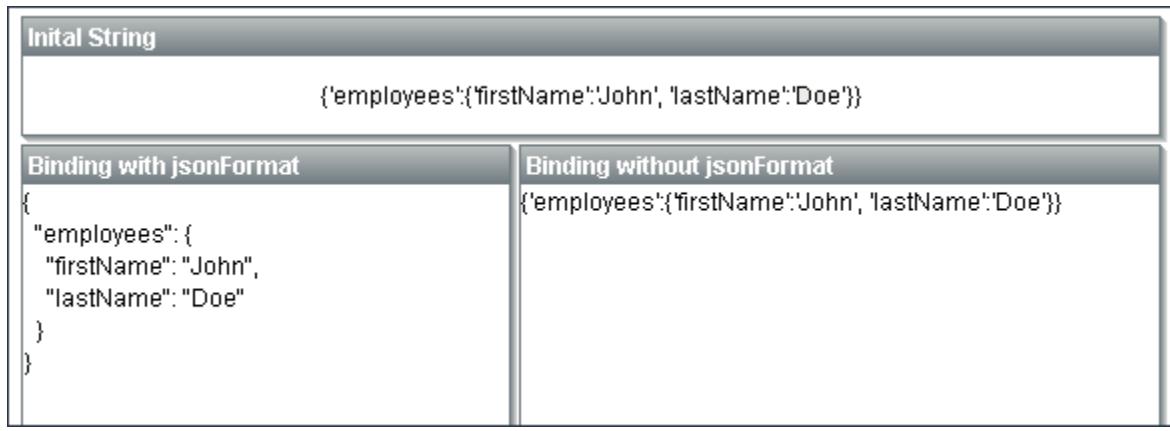
jsonFormat

This function is used by **Ignition's Expression** language.

Description

Takes a string and returns a prettyprints string, making the string easier to read by humans. Especially useful in cases where the string is displayed on components that respect carriage returns.

The image below shows a Label component with a JSON-friendly string. Below that are two Text Area components that are bound to the Label's text: one using the jsonFormat() function and the other without.



Syntax

jsonFormat(string)

- Parameters

String string - The string to format. The string must be in a JSON-friendly format; see examples below for possibilities.

- Results

String - A prettyprint string of the specified string.

Examples

Code Snippet

```
// This example builds a JSON friendly string, and returns a prettyprint version of the string.  
// Since the parameter passed is a string data type, the whole parameter must be wrapped in quotes.  
jsonFormat("{item1:10,item2:20}")
```

Code Snippet

```
// Another valid JSON format - you may optionally wrap inner strings in quotation marks.  
jsonFormat("[{'item1':'apples','item2':'bananas'},{'item1':'oranges','item2':'carrots'}]")
```

jsonGet

This function is used by Ignition's Expression language.

Description

Takes a JSON-friendly string and a path string, and returns the value of that path.

Syntax

jsonGet(json, path)

- Parameters

String json - The JSON string. The string must be in a JSON-friendly format.

String path - The path to look for in the JSON string.

- Results

Object - The value at the path.

Examples

Code Snippet

```
// This example takes a JSON friendly string and finds the value located at the path item.secondThing, which  
is 2.  
jsonGet("{'item':{'firstThing':1, 'secondThing':2}}", "item.secondThing")
```

Code Snippet

```
// This example takes a JSON friendly string and finds the value located at the path item, which is  
{"firstThing":1,"secondThing":2}.  
jsonGet("{'item':{'firstThing':1, 'secondThing':2}}", "item")
```

jsonSet

This function is used by **Ignition's Expression** language.

Description

Takes a JSON-friendly string, a path string, and value, and will return a new JSON-friendly string with the provided path set to the provided value. This is best used in conjunction with the [Derived Tags](#) writeback value.

Syntax

jsonSet(json, path, value)

- Parameters

String json - The JSON string. The string must be in a JSON friendly format.

String path - The path string.

Object value - The replacement for the value at the path.

- Results

String - A JSON-friendly string with a new value set at the specified path.

Examples

Code Snippet

```
// This example takes a JSON friendly string and sets the provided path to the given value. This would then
// return the string {'item':{'firstThing':1, 'secondThing':5}}.
jsonSet("{'item':{'firstThing':1, 'secondThing':2}}", "item.secondThing", 5)
```

Logic

Logic Functions

The following functions allow you to perform logic and evaluate values in expression bindings.

[In This Section ...](#)

binEnc

This function is used by Ignition's Expression language.

Description

This function, whose name stands for "binary encoder", takes a list of booleans and treats them like the bits in a binary number. It returns an integer representing the decimal value of the number. The digits go from least significant to most significant.

Syntax

binEnc(value, [value, ...])

- Parameters

Boolean value - A value that represents a bit. Can be either 0 or 1. Can be as many values as needed.

- Results

Integer - The integer representation of the binary value entered.

Examples

Code Snippet

```
binEnc(0,0,1,0) //Returns 4 (the value of 0100).
```

Code Snippet

```
binEnc(true,0,1,1,0) //Returns 13 (the value of 01101).
```

binEnum

This function is used by Ignition's Expression language.

Description

This function, whose name stands for "binary enumeration," takes a list of booleans, and returns the index (starting at 1) of the first parameter that evaluates to true.

Syntax

binEnum(value, [value, ...])

- Parameters

Integer value - Any number of values.

- Results

Integer - The index of the first true value.

Examples

Code Snippet

```
binEnum(0, 1, 0) //Returns 2.
```

Code Snippet

```
binEnum(0, false, 15, 0, 23) //Returns 3 (the index of the 15 - any non-zero number is "true").
```

case

This function is used by Ignition's Expression language.

Description

This function acts like the switch statement in C-like programming languages. It takes the value argument and compares it to each of the case1 through caseN expressions. If value is equal to caseX, then case returns valueX. If value is not equal to any of the case1..N, then returnDefault is returned.

Note that case() is similar in functionality to the [switch\(\)](#) expression function. The difference between the two is the order in which the parameters are passed.

Syntax

case(value, case, return, [case, return, ...], returnDefault)

- Parameters

Object value - A value of any type.

Object case - A case to match the value to.

Object return - The return if its pair case has been matched.

Object returnDefault - The default return if none of the case arguments were matched.

- Results

Object - The return value for the matched case, or the returnDefault value if no case was matched.

Examples

Code Snippet

```
//The following would return 46 because the value (15) matched case 3, so the third return (46) was returned.  
case(  
    15,           // value  
    1,            // case 1  
    44,           // return 1  
    24,           // case 2  
    45,           // return 2  
    15,           // case 3  
    46,           // return 3  
    -1)          // default
```

Code Snippet

```
//The following would return "Running".  
case(  
    1,           // value  
    0,            // case 1  
    "Off",        // return 1  
    1,            // case 2  
    "Running",    // return 2  
    2,            // case 3  
    "Fault",      // return 3  
    "BAD STATE!") // default
```

coalesce

This function is used by Ignition's Expression language.

Description

This function, which accepts any number of arguments, evaluates each in order, and returns the first non-null argument. A typical use case involves two arguments - the first being something dynamic, the second being a static value to use as a guard in case the dynamic value is null. The function itself detects its return type based on the type of the last argument.

Syntax

coalesce(value, [value, ...])

- Parameters
 - Object** value - Any number of values.
- Returns
 - Object** - The first non null argument.

Examples

Code Snippet

```
coalesce(null, "abc") //Would return "abc".
```

Code Snippet

```
coalesce("xyz", "abc") //Would return "xyz".
```

Code Snippet

```
coalesce({Root Container.MyDataSet}[0,"ColumnName"], 0) //Would return the value in the dataset if it isn't null, but 0 if it is null.
```

getBit

This function is used by Ignition's Expression language.

Description

This function returns the bit value (an integer, 0 or 1) in a number at a given position, according to its binary representation. The least significant bit in a number is position 0.

Syntax

getBit(number, position)

- Parameters

Integer number - The number value to start with.

Integer position - The bit position to check.

- Results

Integer - Returns a 0 or 1, depending on the bit at the position of the integer specified.

Examples

Code Snippet

```
getBit(0,0) //Would return 0.
```

Code Snippet

```
getBit(1,0) //Would return 1.
```

Code Snippet

```
getBit(8,2) //Would return 0.
```

hasChanged

This function is used by Ignition's Expression language.

Description

Note: This function is only available in [Expression Items](#) and [Expression Tags](#).

This function returns true if the given value has changed since the last time the Expression Item was run. Setting the optional boolean argument "include quality" to true means a quality change will make this function return true.

Syntax

hasChanged(value, [includeQuality], [pollRate])

- Parameters

Object value - The number value to check for change

Boolean includeQuality - A flag that indicates if a quality change will also trigger this Tag. [optional]

Integer pollRate - The poll rate in milliseconds. Only applicable on Expression Tags when the Execution Mode is set to Event Driven. All other Execution Modes will ignore this parameter. [optional]

Caution: The pollRate argument was not present in Ignition 7, but is required in Ignition 8.1 and higher for Event Driven Expression Tags. This means that any Expression Tags that are in the Event Driven Execution mode will either need to have the Execution mode changed or the poll rate added in on upgrade to 8.1 or higher, or the function will run once when the value changes and then will not work properly again.

- Results

Boolean - True if the value has changed since the last time the expression was evaluated; False if not.

Examples

Code Snippet

```
hasChanged({[default]Station 1/Status},True) //Would return true if the referenced Tag has changed in value or quality since the last group execution.
```

if

This function is used by Ignition's Expression language.

Description

This function evaluates the expression condition, and returns the value of trueReturn or falseReturn depending on the boolean value of condition.

Syntax

if(condition, trueReturn, falseReturn)

- Parameters

Object condition - The condition to evaluate.

Object trueReturn - The true return value.

Object falseReturn - The false return value.

- Results

Object - Returns the trueReturn if the condition is true, falseReturn if it is False.

Examples

Code Snippet

```
if(1, "Yes", "No") //Would return "Yes".
```

Code Snippet

```
if(0, "Yes", "No") //would return "No"
```

Code Snippet

```
if({Root Container.CheckBox.selected}, "Selected", "Not Selected") //Eould return with a description of the state of the checkbox.
```

Code Snippet

```
//Nests if functions to check the value of 2 different Tags, and return a message based on which ones are greater than 0.  
if({tag1} > 0, if({tag2} > 0, "Both Tags are positive.", "Tag 1 is positive."), if({tag2} > 0, "Tag 2 is positive.", "Neither Tag is positive."))
```

isGood

This function is used by Ignition's Expression language.

Description

Tests to see whether or not the given value is good quality.

Syntax

isGood(value)

- Parameters

Object value - A value to check if it is good.

- Results

Boolean - True if the value is good, False if it is not.

Examples

Code Snippet

```
isGood({path/to/myTag}) //Returns 1 if the value is good, 0 otherwise.
```

isNull

This function is used by Ignition's Expression language.

Description

Tests to see whether or not the argument value is null or not. You can also check for null by simply comparing the value to the null keyword. `isNull(x)` is the same as `x = null`.

Syntax

`isNull(value)`

- Parameters

`Object` `value` - A value to check if it is null.

- Results

`Boolean` - True if the value is null; False if it is not.

Examples

Code Snippet

```
//Returns "Value is Null" if the property is null, and the value otherwise.  
if(isNull({Root Container.MyProperty}), "Value is Null", {Root Container.MyProperty})
```

lookup

This function is used by Ignition's Expression language.

Description

This looks for lookupValue in the lookupColumn of dataset. If it finds a match, it will return the value from the resultColumn on the same row as the match. If no match is found, noMatchValue is returned.

Note: The type of the value returned will always be coerced to be the same type as the noMatchValue.

Syntax

lookup(dataset, lookupValue, noMatchValue, [lookupColumn], [resultColumn])

- Parameters

DataSet dataset - A dataset to search through.

Object lookupValue - The value to look for.

Object noMatchValue - The result value if no match.

Object lookupColumn - The column to lookup. Can either be the column index or the name of the column. Defaults to 0. [optional]

Object resultColumn - The column to pull the result value from. Can either be the column index or the name of the column. Defaults to 1. [optional]

- Results

Object - The value in the result column of the same row that the lookupValue was found, or the noMatchValue if a match was not found.
The data type of this object will always be coerced to match the type of the noMatchValue parameter.

Examples

The examples are based of a table that has the following data in it:

Product	Price	Category
"Apples"	1.99	"Fruit"
"Carrots"	3.5	"Vegetable"
"Walnuts"	6.25	"Nut"

Code Snippet

```
lookup({Root Container.Table.data}, "Carrots", -1.0) //Returns 3.50.
```

Code Snippet

```
lookup({Root Container.Table.data}, "Grapefruit", -1) //Returns -1, the noMatchValue.
```

Code Snippet

```
lookup({Root Container.Table.data}, "Walnuts", "Unknown", 0, "Category") //Returns "Nut".
```

Code Snippet

```
lookup({Root Container.Table.data}, "Pecans", "Unknown", 0, 2) //Returns "Unknown", the noMatchValue.
```

switch

This function is used by Ignition's Expression language.

Description

This function acts like the switch statement in C-like programming languages. It takes the value argument and compares it to each of the case1 through caseN expressions. If value is equal to caseX, then switch returns valueX. If value is not equal to any of the case1..N, then returnDefault is returned.

Note that switch() is similar in functionality to the [case\(\)](#) expression function. The difference between the two is the order in which the parameters are passed.

Syntax

switch(value, case, [caseN, ...], return, [returnN, ...], returnDefault)

- Parameter

Object value - The value to check against the case values.

Object case - A value to check against. Can be any number of case values.

Object return - A value to return for the matching case. Must be the same number of return values as case values.

Object returnDefault - The default return if no case is matched.

- Results

Object - The return value for the case that matched the value, or the returnDefault value if no matches were found.

Examples

Code Snippet

```
//The following would return 46 because the value (15) matched case 3, so the third return (46) was returned.  
switch(  
15, // value  
1, // case 1  
24, // case 2  
15, // case 3  
44, // return 1  
45, // return 2  
46, // return 3  
-1) // default
```

Code Snippet

```
//The following would return "Running".  
switch(  
1, // value  
0, 1, 2, // cases 1-3  
"Off", // return 1  
"Running", // return 2  
"Fault", // return 3  
"BAD STATE!") // default
```

try

This function is used by Ignition's Expression language.

Description

This expression is used to suppress errors caused by other expressions. The first expression will be executed, and if it executes successfully, its value will be used. However, if there is an error evaluating it, the value of failover will be used. When the failover is used, the data quality will be set by the failover value.

Syntax

try(expression, failover)

- Parameters

Object expression - An arbitrary expression.

Object failover - The value to use if there is an error in the expression parameter.

- Results

Object - The result of the expression or the failover value if there is an error.

Examples

Code Snippet

```
try(toInteger("boom"), -1) // Returns -1 with a quality code of 192 (good).
```

Code Snippet

```
// Fetch an integer value from the first row of a table. Return -1 if there are no rows.  
try({Root Container.Power Table.data}[0, 'Integer Column'], -1)
```

Math

Math Functions

The following functions allow you to perform math functions on values in expression bindings.

[In This Section ...](#)

abs

This function is used by Ignition's Expression language.

Description

Returns the absolute value of number.

Syntax

abs(number)

- Parameters

[Integer/Float](#) number - The number to get the absolute value of.

- Results

[Integer/Float](#) - The absolute value of the number provided.

Examples

Code Snippet

```
abs(-4) //Returns 4.
```

acos

This function is used by Ignition's Expression language.

Description

Returns the arc cosine of number, which must be a number between -1 and 1. The results will be an angle expressed in radians in the range of 0.0 through pi.

Syntax

acos(number)

- Parameters
 - Float** number - The number to get the arc cosine of. Must be a value between -1 and 1.
- Results
 - Float** - The arc cosine of the value provided.

Examples

Code Snippet

```
acos(.38) //Returns 1.181.
```

asin

This function is used by Ignition's Expression language.

Description

Returns the arc sine of number, which must be a number between -1 and 1. The results will be an angle expressed in radians in the range of -pi/2 through pi/2.

Syntax

asin(number)

- Parameters

Float number - The number to get the arc sine of. Must be between -1 and 1.

- Results

Float - The arc sine of the number provided.

Examples

Code Snippet

```
asin(.38) //Returns 0.3898.
```

atan

This function is used by Ignition's Expression language.

Description

Returns the arc tangent of number, which must be a number. The results will be an angle expressed in radians in the range of -pi/2 through pi/2

Syntax

atan(number)

- Parameters

Float number - The number to get the arc tangent of.

- Results

Float - The arc tangent of the number provided.

Examples

Code Snippet

```
atan(.38) //Returns 0.3631.
```

ceil

This function is used by Ignition's Expression language.

Description

Returns the smallest floating point value that is greater than or equal to the argument and is equal to a mathematical integer.

Syntax

ceil(number)

- Parameters

[Float](#) number - The number to get the ceiling of.

- Results

[Float](#) - The ceiling of the value provided..

Examples

Code Snippet

```
ceil(2.38) //Returns 3.0.
```

COS

This function is used by Ignition's Expression language.

Description

Returns the trigonometric cosine of number, which is interpreted as an angle expressed in radians. The results will be a floating point value.

Syntax

cos(number)

- Parameters

[Integer/Float](#) number - The number to get the cosine of.

- Results

[Float](#) - The cosine of the number provided.

Examples

Code Snippet

```
cos(1.89) //Returns -0.31381.
```

exp

This function is used by Ignition's Expression language.

Description

Returns Euler's number e raised to the power of the argument number, or e^{number} .

Syntax

exp(number)

- Parameters
 - [Integer/Float](#) number - The exponent value to raise e to the power of.
- Results
 - [Integer/Float](#) - The value of e to the power of the value provided.

Examples

Code Snippet

```
exp(5) //Returns 148.4.
```

floor

This function is used by Ignition's Expression language.

Description

Returns the largest floating point value that is less than or equal to the argument and is equal to a mathematical integer.

Syntax

floor(number)

- Parameters

Float number - The number to get the floor of.

- Results

Float - The floor of the number provided.

Examples

Code Snippet

```
floor(2.72) //Returns 2.0.
```

log

This function is used by Ignition's Expression language.

Description

Returns the natural logarithm (base e) of a number.

Syntax

log(number)

- Parameters

[Integer/Float](#) number - The number to get the log of.

- Results

[Float](#) - The log of the number provided.

Examples

Code Snippet

```
log(28) //Returns 3.332.
```

log10

This function is used by Ignition's Expression language.

Description

Returns the logarithm (base 10) of a number.

Syntax

log10(number)

- Parameters

[Integer/Float](#) number - The number to get the log base 10 of.

- Results

[Float](#) - The log base 10 of the number provided.

Examples

Code Snippet

```
log10(28) // Returns 1.447.
```

pow

This function is used by Ignition's Expression language.

Description

Returns a number raised to a power.

Syntax

pow(number, power)

- Parameters

[Integer/Float](#) number - The number to raise to the provided power.

[Integer/Float](#) power - The power value to raise the number value to.

- Results

[Integer/Float](#) - The result of the number provided raised to the power provided.

Examples

Code Snippet

```
pow(2,3) //Returns 8.
```

round

This function is used by Ignition's Expression language.

Description

Rounds a floating point number. If the decimals argument is omitted, then the number is rounded to the nearest integer value, and the result will be a long (64-bit integer). If a number of decimal places are specified, the result will be a double (64-bit floating point value), and the result will be rounded to the given number of decimal places.

Syntax

round(number, [decimals])

- Parameters

Float number - The number to round.

Integer decimals - The number of decimal places to round to. Defaults to 0. [optional]

- Results

Integer/Float - The value provided rounded to the specified decimal places.

Examples

Code Snippet

```
round(3.829839, 2) //Returns 3.83.
```

sin

This function is used by Ignition's Expression language.

Description

Returns the trigonometric sine of number, which is interpreted as an angle expressed in radians. The results will be a floating point value.

Syntax

sin(number)

- Parameters

[Integer/Float](#) number - The number to get the sine of.

- Results

[Integer/Float](#) - The sine of the number provided.

Examples

Code Snippet

```
sin(1.89) //Returns 0.9495.
```

sqrt

This function is used by Ignition's Expression language.

Description

Returns the square root of the argument number.

Syntax

sqrt(number)

- Parameters

[Integer/Float](#) number - The number to get the square root of.

- Results

[Float](#) - The square root of the number provided.

Examples

Code Snippet

```
sqrt(64) //Returns 8.0.
```

tan

This function is used by Ignition's Expression language.

Description

Returns the trigonometric tangent of number, which is interpreted as an angle expressed in radians. The results will be a floating point value.

Syntax

tan(number)

- Parameters

[Integer/Float](#) number - The number to get the tangent of.

- Results

[Float](#) - The tangent of the number provided.

Examples

Code Snippet

```
tan(1.89) //Returns -3.026.
```

todegrees

This function is used by Ignition's Expression language.

Description

Converts an angle measured in radians to an equivalent angle measured in degrees.

Syntax

todegrees(number)

- Parameters

[Integer/Float](#) number - The number radians.

- Results

[Integer/Float](#) - The degree equivalent of the radians provided.

Examples

Code Snippet

```
todegrees(3.14) //Returns 179.9088.
```

toradians

This function is used by Ignition's Expression language.

Description

Converts an angle measured in degrees to an equivalent angle measured in radians.

Syntax

toradians(number)

- Parameters

[Integer/Float](#) number - The number of degrees.

- Results

[Integer/Float](#) - The radian equivalent of the degrees provided.

Examples

Code Snippet

```
toradians(180) //Returns 3.141592653589793.
```

String

String Functions

The following functions allow you to search or modify string values in expression bindings.

[In This Section ...](#)

concat

This function is used by Ignition's Expression language.

Description

Concatenates all of the strings passed in as arguments together. A null string passed as an argument will be evaluated as the word null. Rarely used, as the + operator does the same thing.

Syntax

concat(string, [string, ...])

- Parameters

String string - Any number of string values to concatenate together.

- Results

String - A string that is all of the strings provided concatenated together.

Examples

Code Snippet

```
concat("The answer is: ", "42") //Returns "The answer is: 42".
```

escapeSQL

This function is used by Ignition's Expression language.

Description

Returns the given string with special SQL characters escaped. This function just replaces single quotes with two single quotes, and backslashes with two backslashes. See [system.db.runPrepUpdate](#) for a safer way to sanitize user input.

Syntax

escapeSQL(string)

- Parameters

String string - The starting string.

- Results

String - A string that has been formatted so that single quotes are replaced with two single quotes, and backslashes are replaced with two backslashes.

Examples

Code Snippet

```
"SELECT * FROM mytable WHERE option = '" + escapeSQL("Jim's Settings") + "'" // Returns SELECT * FROM mytable WHERE option ='Jim''s Settings'.
```

Code Snippet

```
"SELECT * FROM mytable WHERE option = '" + escapeSQL({Root Container.Text Field.text}) + "'" //Returns a query with sanitized user input from a text field.
```

escapeXML

This function is used by Ignition's Expression language.

Description

Returns the given string after being escaped to be valid for inclusion in XML. This means replacing XML special characters with their XML entity equivalents.

Syntax

escapeXML(string)

- Parameters

String string - The starting string.

- Results

String - A string that has been escaped for XML.

Examples

Code Snippet

```
escapeXML("Use Navigate > PB to get to the Pork&Beans section.") //Returns "Use Navigate &gt; PB to get to  
the Pork&amp;Beans section."
```

fromBinary

This function is used by Ignition's Expression language.

Description

Returns an integer value of the binary formatted string argument. Numbers outside of the range (-2³¹) - (2³¹-1), and strings that are not binary numbers, return null.

Syntax

fromBinary(string)

- Parameters

String string - A string representation of a binary.

- Results

Integer - The integer value of the specified binary.

Examples

Code Snippet

```
fromBinary("1111") //Returns 15.
```

Code Snippet

```
fromBinary("-1111") //Returns -15.
```

fromHex

This function is used by Ignition's Expression language.

Description

Returns an integer value of the hex formatted string argument. Numbers outside of the range (-2³¹) - (2³¹-1), and strings that are not hex numbers, return null.

Syntax

fromHex(string)

- Parameters

String string - A string representation of a hex value.

- Results

Integer - The integer of the hex value.

Examples

Code Snippet

```
fromHex("ff") //Returns 255.
```

Code Snippet

```
fromHex("0xff") //Returns 255.
```

Code Snippet

```
fromHex("-ff") //Returns -255.
```

fromOctal

This function is used by Ignition's Expression language.

Description

Returns an integer value of the octal formatted string argument. Numbers outside of the range (-2³¹) - (2³¹-1), and strings that are not octal numbers, return null.

Syntax

fromOctal(string)

- Parameters
 - `String` string - A string representation of an octal.
- Results
 - `Integer` - The integer of the octal value.

Examples

Code Snippet

```
fromOctal("77") //Returns 63.
```

Code Snippet

```
fromOctal("-77") //Returns -63.
```

indexOf

This function is used by Ignition's Expression language.

Description

Searches for the first occurrence of the substring inside of string. Returns the index of where substring was found, or -1 if it wasn't found. The first position in the string is position 0.

Syntax

indexOf(string, substring)

- Parameters

`String` string - The string to search through.

`String` substring - The string to search for.

- Results

`String` - The index where the substring was first found in the string.

Examples

Code Snippet

```
indexOf( "Hamburger" , "urge" ) //Returns 4.
```

Code Snippet

```
indexOf( "Test" , "" ) //returns 0
```

Code Snippet

```
indexOf( "Dysfunctional" , "fun" ) //returns 3
```

Code Snippet

```
indexOf( "Dysfunctional" , "marble" ) //returns -1
```

Code Snippet

```
indexOf( "banana" , "n" ) //returns 2
```

lastIndexOf

This function is used by Ignition's Expression language.

Description

Searches for the last occurrence of the substring inside of string. Returns the index of where substring was found, or -1 if it wasn't found. The first position in the string is position 0.

Syntax

lastIndexOf(string, substring)

- Parameters

`String` string - The string to search through.

`String` substring - The string to search for.

- Results

`String` - The index where the substring was last found in the string.

Examples

Code Snippet

```
lastIndexOf( "Hamburger" , "urge" ) //Returns 4.
```

Code Snippet

```
lastIndexOf( "Test" , "" ) //Returns 4.
```

Code Snippet

```
lastIndexOf( "Dysfunctional" , "fun" ) //Returns 3.
```

Code Snippet

```
lastIndexOf( "Dysfunctional" , "marble" ) //Returns -1.
```

Code Snippet

```
lastIndexOf( "banana" , "n" ) //Returns 4.
```

left

This function is used by Ignition's Expression language.

Description

Returns count characters from the left side of string, where count and string are the arguments to the function.

Syntax

left(string, charCount)

- Parameters

String string - The starting string.

Integer charCount - The number of characters to return.

- Results

String - A string that is the first charCount number of characters of the specified string.

Examples

Code Snippet

```
left("hello", 2) //Returns "he".
```

Code Snippet

```
left("hello", 0) //Returns "".
```

Code Snippet

```
left("hello", 5) //Returns "hello".
```

len

This function is used by Ignition's Expression language.

Description

Returns the length of the argument, which may be a string or a dataset. If the argument is a string, it returns the number of characters in the string. If the argument is a dataset, it returns the number of rows in the dataset. Will return zero if the argument is null.

Syntax

len(value)

- Parameters

Object value- The starting object.

- Results

Integer - The length of the provided object.

Examples

Code Snippet

```
len("Hello World") //Returns 11.
```

Code Snippet

```
len({Root Container.Table.data}) //Returns the number of rows in the table.
```

lower

This function is used by Ignition's Expression language.

Description

Takes a string and returns a lower-case version of it.

Syntax

lower(string)

- Parameters

String string - The string to make lowercase.

- Results

String - The starting string with all characters lowercase.

Examples

Code Snippet

```
lower("Hello World") // Returns "hello world".
```

numberFormat

This function is used by Ignition's Expression language.

Description

Returns a string version of the number argument, formatted as specified by the pattern string. This is commonly used to specify the number of decimal places to display, but can be used for more advanced formatting as well. The pattern string is a numeric format string, which may include any of these characters that instruct it how to format the number.

Refer to [Data Type Formatting Reference](#).

Syntax

numberFormat(number, pattern)

- Parameters

[Float](#) number - The number to format.

[String](#) pattern - The format pattern.

- Results

[String](#) - The string representation of the number formatted according to the pattern provided.

Examples

Code Snippet

```
numberFormat(34.8, "#0.00'%'") //Returns the string "34.80%".
```

repeat

This function is used by Ignition's Expression language.

Description

Repeats the given string some number of times.

Syntax

repeat(string, count)

- Parameters

String string - The string to repeat

Integer count - The number of times to repeat the string.

- Results

String - The given string repeated the given number of times.

Examples

Code Snippet

```
repeat("hello", 2) //Returns "hellohello".
```

Code Snippet

```
repeat("hello", 0) //Returns "".
```

replace

This function is used by Ignition's Expression language.

Description

Finds all occurrences of a substring inside of a source string, and replaces them with the replacement string. The first argument is the source, the second is the search string, and the third is the replacement.

Syntax

replace(string, substring, replacementString)

- Parameters

String string - The starting string.

String substring - The string to search for.

String replacementString - The string to replace any instances of the substring with.

- Results

String - The starting string with all instances of the substring replaced by the replacementString.

Examples

Code Snippet

```
replace("XYZ", "Y", "and") //Returns "XandZ".
```

Code Snippet

```
replace("bob and mary went to bob's house", "bob", "judith") //Returns "judith and mary went to judith's house".
```

right

This function is used by Ignition's Expression language.

Description

Returns count number of characters starting from the right side of string, where count and string are the arguments to the function.

Syntax

right(string, charCount)

- Parameters

String string - The starting string.

String charCount - The number of characters to return.

- Results

String - A string of the number of characters specified in the charCount from the specified string.

Examples

Code Snippet

```
right("hello", 2) //Returns "lo".
```

Code Snippet

```
right("filename.pdf", 3) //Returns "pdf".
```

Code Snippet

```
right("hello", 0) //Returns "".
```

split

This function is used by Ignition's Expression language.

Description

This function takes the string string and splits it into a bunch of substrings. The substrings are returned as a dataset with one column called "parts". The split occurs wherever the regular expression regex occurs. Don't be intimidated by the regular expression, this is normally just another string, like "," for comma separated lists.

The optional limit argument, if greater than zero, limits the number of times the regex pattern is applied to limit-1. Put another way, it limits the length of the resulting dataset to length limit. If limit is non-positive then the regex pattern will be applied as many times as possible and the returned dataset can have any length. If limit is zero (the default) then the pattern will be applied as many times as possible, the returned dataset can have any length, and trailing empty strings will be discarded.

Syntax

split(string, regex, [limit])

- Parameters

String string - The starting string.

String regex - The string to split on.

Integer limit - The max number of splits to make. Default 0 which is as many as possible. [optional]

- Results

Dataset - The split string, with a single column called parts, where each row is a new part of the string.

Examples

Code Snippet

```
split("hello,world", ",") //Returns dataset [[{"hello"}, {"world"}]].
```

Code Snippet

```
split("boo:and:foo", ":") //Returns dataset [[{"boo"}, {"and"}, {"foo"}]].
```

Code Snippet

```
split("boo:and:foo", ":", 2) //Returns dataset [[{"boo"}, {"and:foo"}]].
```

substring

This function is used by Ignition's Expression language.

Description

Substring will return the portion of the string from the startIndex (inclusive) to the endIndex (exclusive), or end of the string if endIndex is not specified. All indexes start at 0, so in the string "Test", "s" is at index 2. Indexes outside of the range of the string throw a StringIndexOutOfBoundsException.

Syntax

substring(string, startIndex, [endIndex])

- Parameters

String string - The starting string.

String startIndex - The index (inclusive) to start the substring at.

String endIndex - The end index (exclusive) of the substring. [optional]

- Results

String - The substring from the start to end indexes of the specified string.

Examples

Code Snippet

```
substring("unhappy", 2) //Returns "happy".
```

Code Snippet

```
substring("hamburger", 4, 8) //Returns "urge" - note 8 is the index of "r", but the end index is exclusive.
```

toBinary

This function is used by Ignition's Expression language.

Description

Returns an binary formatted string representing the unsigned integer argument. If the argument is negative, the binary string represents the value plus 2³².

Syntax

toBinary(number)

- Parameters

Integer number - The value to convert to binary.

- Results

String - The string form of the binary representation of the specified number.

Examples

Code Snippet

```
toBinary(255) //Returns "11111111".
```

Code Snippet

```
toBinary(-255) //Returns "111111111111111111111100000001".
```

toHex

This function is used by Ignition's Expression language.

Description

Returns a hex formatted string representing the unsigned integer argument. If the argument is negative, the hex string represents the value plus 2^{32} .

Syntax

toHex(number)

- Parameters

[Integer](#) number - The number to convert to hex.

- Results

[String](#) - A string that is the hex value of the specified value.

Examples

Code Snippet

```
toHex(255) //Returns "FF".
```

Code Snippet

```
toHex(-255) //Returns "FFFFFF01".
```

toOctal

This function is used by Ignition's Expression language.

Description

Returns an octal formatted string representing the unsigned integer argument. If the argument is negative, the octal string represents the value plus 2^{32} .

Syntax

toOctal(number)

- Parameters

[Integer](#) number - The value to convert to octal.

- Results

[String](#) - A string that is the octal of the specified value.

Examples

Code Snippet

```
toOctal(255) //Returns "377".
```

Code Snippet

```
toOctal(-255) //Returns "3777777401".
```

trim

This function is used by Ignition's Expression language.

Description

Takes the argument string and trims of any leading and/or trailing whitespace, returning the result.

Syntax

trim(string)

- Parameters

String string - The starting string.

- Results

String - The starting string with all whitespace removed.

Examples

Code Snippet

```
trim("Hello Dave   ") //Returns "Hello Dave".
```

Code Snippet

```
trim( " Goodbye." ) //Returns "Goodbye."
```

upper

This function is used by Ignition's Expression language.

Description

Takes a string and returns an uppercase version of it.

Syntax

upper(string)

- Parameters

String string - The string to make uppercase.

- Results

String - The starting string with all characters uppercase.

Examples

Code Snippet

```
upper("Hello World") //Returns "HELLO WORLD".
```

stringFormat

This function is used by Ignition's Expression language.

Description

This expression returns a formatted string using the specified format string and arguments. Mainly, this expression is used for building dynamic string objects.

Refer to [Data Type Formatting Reference](#) for formatting elements.

Syntax

stringFormat(format, [args, ...])

- Parameters

String format - The a string that contains formatting elements in it (%s, %d, %i).

String args - The arguments to use in the format. Must match the number of formatting elements in the string. [optional]

- Results

String - The new formatted string.

Examples

Code Snippet

```
stringFormat("The boolean value is: %b", null) //Returns The boolean value is: False.
```

Code Snippet

```
stringFormat("Hello %s", "world") //Returns "Hello world".
```

Code Snippet

```
stringFormat("%s, %s, %s", 1, 2, 3) //Returns "1, 2, 3".
```

Code Snippet

```
stringFormat("%d, %d, %d", 4, 5, 6) //Returns "4, 5, 6".
```

Code Snippet

```
stringFormat("Today is: %tA", now()) //Returns Today is: Tuesday.
```

Code Snippet

```
stringFormat("The current month is: %tB", now()) //Returns The current month is: June.
```

urlEncode

This function is used by Ignition's Expression language.

Description

The expression function urlEncode() enables users to create an HTTP binding's URL on the fly. It will be possible to freely bind different URL path parameters to different component properties on a view or window.

The first argument is the string that's being encoded, second parameter is a boolean arg that controls whether you use query parameter style escaping or URI fragment style escaping.

Syntax

urlEncode(string, [usePercentEscape])

- Parameters

String url- The URL to encode.

Boolean usePercentEscape - False or black indicates to use query parameter style escaping. True indicates to use URI fragment style escaping. [optional]

- Results

String- The encoded URL.

Examples

Code Snippet

```
urlEncode("Hello World") //Yields "Hello+World".
```

Code Snippet

```
urlEncode("Hello World", False) //Yields "Hello+World".
```

Code Snippet

```
urlEncode("Hello World", True) //yields "Hello%20World".
```

Example

Vision Example

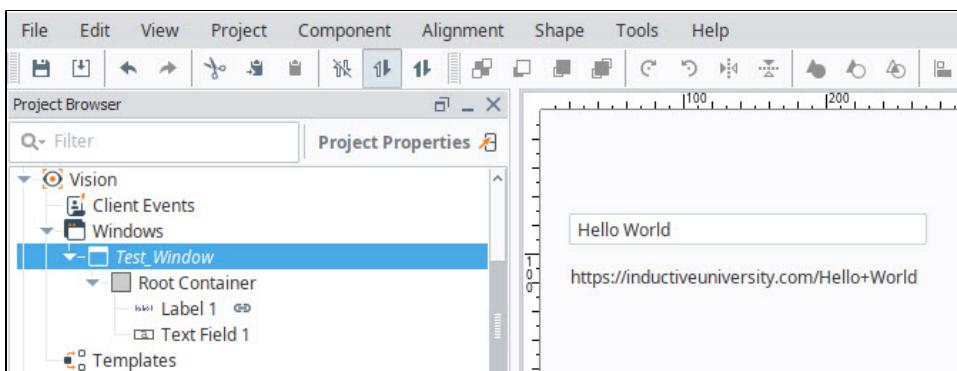
1. Drag a Text Field component and a Label component onto a window.
2. In the Vision Property Editor, enter a parameter for the URL you'll use. For this example, we used 'Hello World.'
3. Select the Label component, then click on the **Binding** icon for the label's Text property.
4. On the Property Binding screen, select **Expression**.
5. Enter the expression in the Configure Expression Binding section. In the example, we entered the URL plus the expression function, "urlEncode ()". We entered the oath to the Text Field's text property as the string.

Code Snippet

```
"https://inductiveuniversity.com/" + urlEncode({Root Container.Text Field 1.text})
```



6. Alternatively, you can get the path for the text field by putting the cursor inside the parentheses, clicking the **Property Value** icon, then selecting the text property from the Text Field component.
7. Click **OK** to save the binding. The encoded URL is now displayed in the Label component.



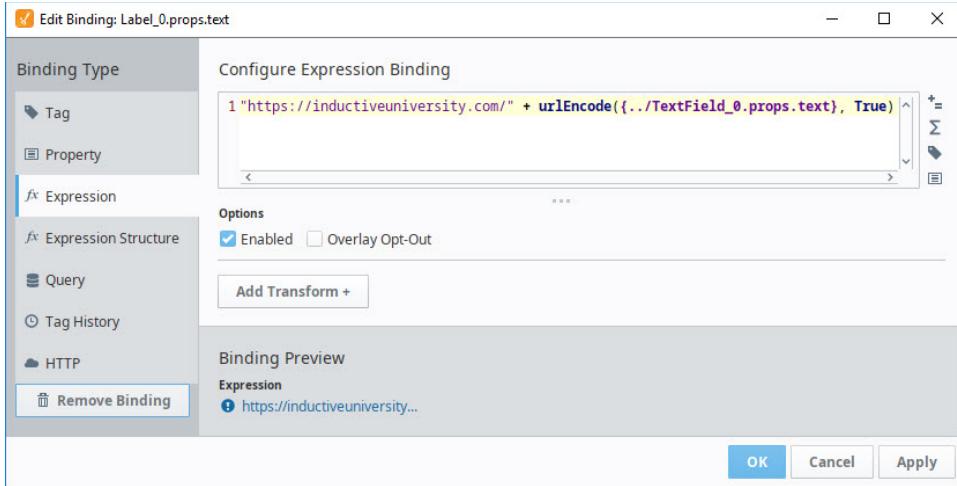
Example

Perspective Example

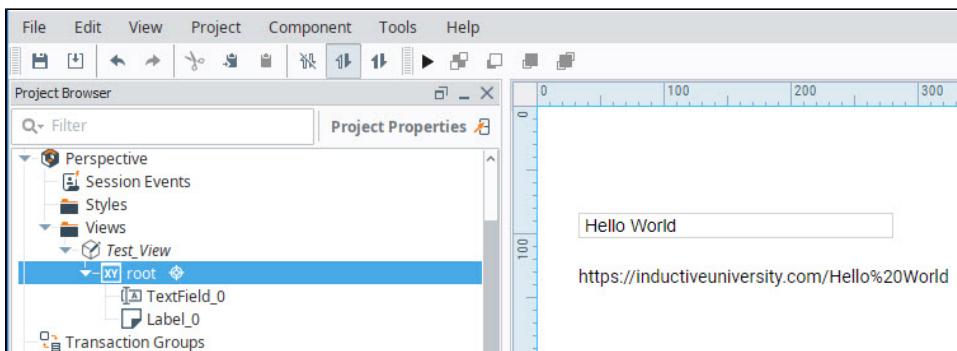
1. Drag a Text Field component and a Label component onto a view.
2. In the Perspective Property Editor, enter a parameter for the URL you'll use. For this example, we used 'Hello World.'
3. Select the Label component, then click on the **Binding** icon for the label's Text property.
4. On the Property Binding screen, select **Expression**.
5. Enter the expression in the Configure Expression Binding section. In the example, we entered the URL plus the expression function, "urlEncode ()". We entered the path to the Text Field's text property as the string and the parameter "True" to use URI fragment style escaping.

Code Snippet

```
"https://inductiveuniversity.com/" + urlEncode({../TextField_0.props.text}, True)
```



6. Alternatively, you can get the path for the text field by putting the cursor inside the parentheses, clicking the **Property Value** icon, then selecting the text property from the Text Field component.
7. Click **OK** to save the binding. The encoded URL is now displayed in the Label component.



Translation

Translation Functions

The following functions allow you to interact with the Translation system through expression bindings.

[In This Section ...](#)

translate

This function is used by Ignition's Expression language.

Description

Returns a translated string, based on the current locale. If the string does not exist in the global translations, the original string will be returned. This function exists in the Client and Gateway scopes.

Syntax

translate(stringKey)

- Parameters

String stringKey - The starting string to translate.

String languageString - The language or locale to use.

- Results

String - The starting string translated based on the current locale. If the translation does not exist, will return the specified value.

Examples

There are no examples associated with this expression function.

Type Casting

Type Casting Functions

The following functions allow you to change value types in expression bindings.

[In This Section ...](#)

toBoolean

This function is used by Ignition's Expression language.

Description

Tries to convert value to a boolean, according to these rules:

1. If value is a number, 0 is false and anything else is true.
2. If value is a string, then the strings (case insensitive) "on", "true", "t", "yes", "y" are all true. The strings (case insensitive) "off", "false", "f", "no", "n" are considered false. If the string represents a number, the first rule applies. All other strings fail type casting.
3. All other types fail type casting.

If type casting fails, an error is thrown, unless the failover argument is specified, in which case it will be used.

Syntax

toBoolean(value[, failover])

- Parameters

object value - The value to type cast.

object failover - *Optional*. The failover value if type casting fails.

- Results

Bool - The value type cast as a bool.

Examples

Code Snippet

```
toBoolean(1) //returns true
```

Code Snippet

```
toBoolean("abc", false) //returns false
```

toBorder

This function is used by Ignition's Expression language.

Description

This function is used specifically when binding a **Border** property on a component. Typically, this is used with a Container or Label component but can be used on any component that has a Border property.

This function takes a string and converts it into a border. The string must be a semi-colon separated list of values. The first value is the name of the border, and the other values depend on the type of border you use. The following table defines the border types and the arguments they accept.

Border Type	Options	Type	Style	Font Justification
bevel	bevelType	0 = Raised 1 = Lowered 1010 = Double		
button	Nothing			
etched	etchType	0 = Raised 1 = Lowered		
etchedtitled	title; style; fontJustification; fontPosition; fontColor; font		0 = Etched / Lowered 1 = Etched / Raised 2 = Beveled / Lowered 3 = Beveled / Raised 4 = Beveled / Double 5 = Standard	1 = Left 2 = Center 3 = Right 4 = Leading 5 = Trailing
field	Nothing			
line	color; thickness			
linetitled	title; width; lineColor; fontJustification; fontPosition; fontColor; font			1 = Left 2 = Center 3 = Right 4 = Leading 5 = Trailing
matte	color; topWidth, leftWidth; bottomWidth; rightWidth			
paneltitled	title; style; mainColor; bgColor, shadowSize, fontJustification; fontPosition; fontColor; font		1=Gradient / West-to-East 2=Gradient / North-to-South 3=Gradient / East-to-West 4=Solid	1 = Left 2 = Center 3 = Right 4 = Leading 5 = Trailing

To use this function, you need to include the border type and then any options you want to use in the correct order, for example:

```
toBorder("paneltitled; title; style; mainColor; bgColor, shadowSize, fontJustification; fontPosition; fontColor; font")
```

Syntax

toBorder(value, [failover])

- Parameters

String value - The value to type cast.

Object failover - The failover value if type casting fails. [optional]

- Results

Border - The value type cast as a border object.

Examples

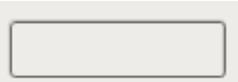
Code Snippet

```
toBorder("bevel;1010") //Returns this...
```



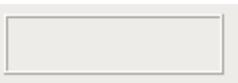
Code Snippet

```
toBorder("button") //Returns this...
```



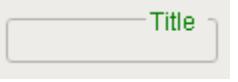
Code Snippet

```
toBorder("etched;0") //Returns this...
```



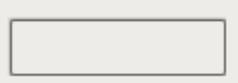
Code Snippet

```
toBorder("etchedtitled>Title;5;3;right:green;Arial") //Returns this...
```



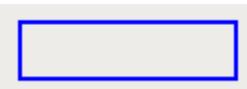
Code Snippet

```
toBorder("field") //Returns this...
```



Code Snippet

```
toBorder("line;blue;2")
```

**Code Snippet**

```
toBorder("linetitled>Title" ) //Returns this...
```

**Code Snippet**

```
toBorder("matte;red;10;1;1;1" ) //Returns this...
```

**Code Snippet**

```
toBorder("paneltitled;Options;1;grey;white;0;3;0;green;Dialog,bold,16" ) //Returns this...
```



toColor

This function is used by Ignition's Expression language.

Description

This function tries to convert value to a color. It assumes that value is a string. If you have integers representing Red, Green, and Blue values see the [color](#) expression. The string value is converted to a color according to these rules:

1. If value is a name of a color as defined in the table below, the corresponding color will be returned. Note that color names are case insensitive.
2. If value is a hex color string (with or without a leading "#", the color equivalent of that hex string will be used, for example: "#FF0000", "556B2F".
3. If value is a list of 3 or 4 integers, a color will be created that uses the first three integers as red, green, and blue values, and the optional fourth integer as an alpha channel value. All values should be between 0 and 255. The list is free-form, any non-digit characters may be used as delimiters between the digits. Examples: "(0,0,0)", "23-99-203", "[255,255,33,127]"

Note: This function was designed to return color objects to Vision bindings, and will not work with Perspective bindings. Instead, Perspective color properties can simply use string hex codes to derive a color from a binding, for example: "#00FF00".

Syntax

toColor(value, [failover])

- Parameters
 - String** value - The color value as a string.
 - Object** failover - The failover value if type casting fails. [optional]
- Results
 - Boolean** - The value type cast as a color object.

Examples

Code Snippet

```
//All of these expressions return the color red.  
toColor("red")  
toColor("#FF0000")  
toColor("255,0,0")
```

Code Snippet

```
//You can use the failover parameter to ensure that this expression returns something even if the input  
string may be bad:  
toColor({UserOptions/CustomColor}, "black")
```

Color Options

Note: Both "Grey" and "Gray" are accepted as valid colors, as well as the iterations of that color such as "DarkGrey" and "DarkGray."

AliceBlue	#F0F8FF	
AntiqueWhite	#FAEBD7	

Aqua	#00FFFF
Aquamarine	#7FFFAD
Azure	#F0FFFF
Beige	#F5F5DC
Bisque	#FFE4C4
Black	#000000
BlanchedAlmond	#FFEBBC
Blue	#0000FF
BlueViolet	#8A2BE2
Brown	#A52A2A
BurlyWood	#DEB887
CadetBlue	#5F9EA0
Chartreuse	#7FFF00
Chocolate	#D2691E
Clear	See Transparent
Coral	#FF7F50
CornflowerBlue	#6495ED
Cornsilk	#FFF8DC
Crimson	#DC143C
Cyan	#00FFFF
DarkBlue	#00008B
DarkCyan	#008B8B
DarkGoldenRod	#B8860B
DarkGray, DarkGrey	#A9A9A9
DarkGreen	#006400
DarkKhaki	#BDB76B
DarkMagenta	#8B008B
DarkOliveGreen	#556B2F
DarkOrange	#FF8C00
DarkOrchid	#9932CC
DarkRed	#8B0000
DarkSalmon	#E9967A
DarkSeaGreen	#8FBBC8F
DarkSlateBlue	#483D8B
DarkSlateGray, DarkSlateGrey	#2F4F4F
DarkTurquoise	#00CED1
DarkViolet	#9400D3
DeepPink	#FF1493
DeepSkyBlue	#00BFFF
DimGray, DimGrey	#696969
DodgerBlue	#1E90FF

Feldspar	#D19275
FireBrick	#B22222
FloralWhite	#FFF0F0
ForestGreen	#228B22
Fuchsia	#FF00FF
Gainsboro	#DCDCDC
GhostWhite	#F8F8FF
Gold	#FFD700
GoldenRod	#DAA520
Gray, Grey	#808080
Green	#008000
GreenYellow	#ADFF2F
HoneyDew	#F0FFF0
HotPink	#FF69B4
IndianRed	#CD5C5C
Indigo	#4B0082
Ivory	#FFFFFF0
Khaki	#F0E68C
Lavender	#E6E6FA
LavenderBlush	#FFF0F5
LawnGreen	#7CFC00
LemonChiffon	#FFFACD
LightBlue	#ADD8E6
LightCoral	#F08080
LightCyan	#E0FFFF
LightGoldenRodYellow	#FAFAD2
LightGreen	#90EE90
LightGray, LightGrey	#D3D3D3
LightPink	#FFB6C1
LightSalmon	#FFA07A
LightSeaGreen	#20B2AA
LightSkyBlue	#87CEFA
LightSlateBlue	#8470FF
LightSlateGray, LightSlateGrey	#778899
LightSteelBlue	#B0C4DE
LightYellow	#FFFFE0
Lime	#00FF00
LimeGreen	#32CD32
Linen	#FAF0E6
Magenta	#FF00FF
Maroon	#800000

MediumAquaMarine	#66CDAA
MediumBlue	#0000CD
MediumOrchid	#BA55D3
MediumPurple	#9370D8
MediumSeaGreen	#3CB371
MediumSlateBlue	#7B68EE
MediumSpringGreen	#00FA9A
MediumTurquoise	#48D1CC
MediumVioletRed	#C71585
MidnightBlue	#191970
MintCream	#F5FFFA
MistyRose	#FFE4E1
Moccasin	#FFE4B5
NavajoWhite	#FFDEAD
Navy	#000080
OldLace	#FDF5E6
Olive	#808000
OliveDrab	#6B8E23
Orange	#FFA500
OrangeRed	#FF4500
Orchid	#DA70D6
PaleGoldenRod	#EEE8AA
PaleGreen	#98FB98
PaleTurquoise	#AFEEEE
PaleVioletRed	#D87093
PapayaWhip	#FFEF5D
PeachPuff	#FFDAB9
Peru	#CD853F
Pink	#FFC0CB
Plum	#DDA0DD
PowderBlue	#B0E0E6
Purple	#800080
Red	#FF0000
RosyBrown	#BC8F8F
RoyalBlue	#4169E1
SaddleBrown	#8B4513
Salmon	#FA8072
SandyBrown	#F4A460
SeaGreen	#2E8B57
SeaShell	#FFF5EE
Sienna	#A0522D

Silver	#C0C0C0		
SkyBlue	#87CEEB		
SlateBlue	#6A5ACD		
SlateGray	#708090		
SlateGrey			
Snow	#FFFFFA		
SpringGreen	#00FF7F		
SteelBlue	#4682B4		
Tan	#D2B48C		
Teal	#008080		
Thistle	#D8bfd8		
Tomato	#FF6347		
Transparent	#FFFFFF		
Turquoise	#40E0D0		
Violet	#EE82EE		
VioletRed	#D02090		
Wheat	#F5DEB3		
White	#FFFFFF		
WhiteSmoke	#F5F5F5		
Yellow	#FFFF00		
YellowGreen	#9ACD32		

toDataSet

This function is used by Ignition's Expression language.

Description

Tries to coerce value into a dataset. Not many things can be coerced into datasets. Namely, only datasets and PyDatasets can be coerced into datasets. This is useful for the [runScript\(\)](#) expression, to convince the expression compiler to let you assign the return value of a scripting function to a dataset property.

Syntax

toDataSet(value[, failover])

- Parameters

Object value - The value to type cast, typically a dataset or PyDataset.

Object failover - The failover value if type casting fails. [optional]

- Results

Dataset - The value type cast as a dataset.

Examples

Code Snippet

```
toDataSet(runScript("app.funcs.runSomeFunction()")) //Coerces the value returned by the a project scripting function into a dataset.
```

toDate

This function is used by Ignition's Expression language.

Description

Tries to coerce value into a date. If value is a number or a string that represents a number, the number is treated as the number of milliseconds since the epoch, January 1, 1970, 00:00:00 GMT. If value is a string, it is parsed to see if it represents a date in one of the supported formats:

- 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
- yyyyMMdd.HHmmssSSSZ

If not, type casting fails. The failover value must be a number or string with the same restrictions.

Syntax

toDate(value, [failover])

- Parameters

Object value - The value to type cast into a date.

Object failover - The failover value if type casting fails. [optional]

- Results

Date - The value type cast as a date.

Examples

Code Snippet

```
toDate("2007-04-12 16:28:22") //Returns April 12th, 2007, 4:28:22 PM.
```

toDouble

This function is used by Ignition's Expression language.

Description

Tries to coerce value into a double (64-bit floating point value). If value is a number, the conversion is direct. If value is a string, it is parsed to see if it represents a double. If not, type casting fails.

Syntax

toDouble(value, [failover])

- Parameters

Object value - The value to type cast.

Object failover - The failover value if type casting fails. [optional]

- Results

Double - The value type cast as a double.

Examples

Code Snippet

```
toDouble("38.772") //Returns 38.772.
```

Code Snippet

```
toDouble({Root Container.Text Field.text}, 0.0) //Returns the value in the text box as a double, or 0.0 if the value doesn't represent an number.
```

toFloat

This function is used by Ignition's Expression language.

Description

Tries to coerce value into a float (32-bit floating point value). If value is a number, the conversion is direct. If value is a string, it is parsed to see if it represents a float. If not, type casting fails.

Syntax

toFloat(value[, failover])

- Parameters

Object value - The value to type cast.

Object failover - The failover value if type casting fails. [optional]

- Results

Float - The value type cast as a float.

Examples

Code Snippet

```
toFloat("38.772") //Returns 38.772.
```

Code Snippet

```
toFloat({Root Container.Text Field.text}, 0.0) //Returns the value in the text box as a float, or 0.0 if the value doesn't represent an number.
```

toFont

This function is used by Ignition's Expression language.

Description

Coerces a string into a font. The string must be in the format:

- 'fontName, fontType, fontSize)'
- fontName is the name of the font to use. Note that special care must be taken with fonts, because of the web-launched nature of the clients. You can only use font names that exist on the client machines. The following font names are known as logical fonts, meaning that they are guaranteed to exist on all systems, mapped to the most appropriate real, or physical font that exists on the host system:
 - Serif
 - SansSerif
 - Monospaced
 - Dialog
 - DialogInput
- fontType is a string, that should match one of these (case-insensitive):
 - Plain
 - Bold
 - Italic
 - BoldItalic
- fontSize is an integer that represent the font's point size.

Syntax

toFont(value, [failover])

- Parameters

String value - The value to type cast to a font.

Object failover - The failover value if type casting fails. [optional]

- Results

Font - The value type cast as a font.

Examples

Code Snippet

```
toFont("font(Dialog,Bold,12)") //Returns the standard font used in most clients.
```

toInt

This function is used by Ignition's Expression language.

Description

Tries to coerce value into an integer (32-bit integer). If value is a number, the conversion is direct (with possible loss of precision). If value is a string, it is parsed to see if it represents an integer. If not, type casting fails. Will round if appropriate.

Syntax

toInt(value, [failover])

- Parameters

Object value - The value to type cast.

Object failover - The failover value if type casting fails. [optional]

- Results

Integer - The value type cast as an int.

Examples

Code Snippet

```
toInt("38") //Returns 38.
```

Code Snippet

```
toInt("33.9") //Returns 34.
```

Code Snippet

```
toInt({Root Container.Text Field.text}, -1) //Returns the value in the text box as an int, or -1 if the value doesn't represent an number.
```

toInteger

This function is used by Ignition's Expression language.

Description

Identical to the [toInt](#) expression function.

Syntax

toInteger(value, [failover])

- Parameters

[Object](#) value - The value to type cast.

[Object](#) failover - The failover value if type casting fails. [optional]

- Results

[Integer](#) - The value type cast as an int.

Examples

There are no examples associated with this expression function.

toLong

This function is used by Ignition's Expression language.

Description

Tries to coerce value into a long (64-bit integer). If value is a number, the conversion is direct. If value is a string, it is parsed to see if it represents a long. If not, type casting fails. Will round if appropriate.

Syntax

toLong(value, [failover])

- Parameters

object value - The value to type cast.

object failover - The failover value if type casting fails. [optional]

- Results

Long - The value type cast as a long.

Examples

Code Snippet

```
toLong("38") //Returns 38.
```

Code Snippet

```
toLong("33.9") //Returns 34.
```

Code Snippet

```
toLong({Root Container.Text Field.text}, -1) //Returns the value in the text box as an long, or -1 if the value doesn't represent an number.
```

toStr

This function is used by Ignition's Expression language.

Description

Identical to the [toString](#) expression function.

Syntax

toStr(value, [failover])

- Parameters

[Object](#) value - The value to type cast, typically a Dataset or PyDataset.

[Object](#) failover - The failover value if type casting fails. [optional]

- Results

[string](#) - The value type cast as a string.

Examples

There are no examples associated with this expression function.

toString

This function is used by Ignition's Expression language.

Description

Represents the value as a string. Will succeed for any type of value.

Syntax

toString(value, [failover])

- Parameters

Object value - The value to type cast.

Object failover - The failover value if type casting fails. [optional]

- Results

String - The value type cast as a string.

Examples

Code Snippet

```
toString(1/3.0) //Returns "0.3333333333333333".
```

Code Snippet

```
toString({Root Container.Table.data}) //Returns something like: "Dataset [150R x 3C]".
```

Users

User Functions

The following functions allow you to interact with the User/Role system through expression bindings.

[In This Section ...](#)

hasRole

This function is used by Ignition's Expression language.

Description

Returns true if the user has the given role. The username and usersource parameters are optional in the Client scope, but required in the Gateway scope.

Syntax

hasRole(role[, username][, usersource])

- Parameters

String role - The name of a role.

String username - A username. Defaults to the current user. [optional]

String usersource - The usersource of the username. Defaults to the usersource of the current user. [optional]

- Results

Boolean - True if the specified user has the specified role, False if not.

Examples

Code Snippet

```
// This is an example using a username and userSource:  
hasRole("Administrator", "bob", "default")
```

Code Snippet

```
// This is an example using the current user and default userSource in the Client scope:  
hasRole("Administrator")
```

isAuthorized

This function is used by Ignition's Expression language.

Description

Returns a qualified value with a boolean value which is true if the user in the current Session is authorized, false otherwise. Scope is Perspective Sessions only.

Syntax

isAuthorized(isAllOf, securityLevel[, securityLevelN...])

- Parameters

boolean `isAllOf` - True if the current user must have all of the given security levels to be authorized, false if the current user must have at least one of the given security levels to be authorized

string `securityLevels` - One or more String paths to a security level node in the form "Path/To/Node". Each level in the tree is delimited by a forward slash character. Additional security level paths are simply added to the end of the parameter list. The Public node is never a part of the path.

- Results

Bool - Returns a qualified value with a boolean value which is true if the user in the current Session is authorized, false otherwise. The quality of the qualified value is the worst of the qualities of all the qualified values of each argument.

Examples

Code Snippet

```
// Returns true if the current user has both Administrator and Baz roles.  
// Returns false if they have only one or if they have neither.  
isAuthorized(true, 'Authenticated/Roles/Administrator', 'Foo/Bar/Baz')
```

Scripting Functions

The Ignition scripting API, which is available under the module name "system", is full of functions that are useful when designing projects in Ignition. From running database queries, manipulating components, to exporting data, scripting functions can help. Some of these functions only work in the Gateway scope, and other only work in the Client scope, while the rest will work in any scope.

Additional information on scripting Ignition can be found in the [Scripting](#) section.

In this section, we cover all of the built in scripting functions available inside of Ignition. Each page will have a banner at the top that looks like this:



This function is used in [Python Scripting](#).

This lets you know that you are looking at a function for the Python scripting language.

Keyboard Shortcut

A complete list of these functions (with their definitions) is available wherever you can add a script. Just type **system**. and then press **Ctrl+Space** to get a list of all the functions available. If you keep typing, the list will even be automatically narrowed down for you!

System Functions

You can see below there are many different categories of system functions available for your use. For an overview and syntax scripting, refer to the [Python Scripting](#) section.

system.alarm	system.groups	system.roster
system.bacnet	system.gui	system.secsgem
system.dataset	system.math	system.security
system.date	system.nav	system.serial
system.db	system.net	system.sfc
system.device	system.opc	system.tag
system.dnp3	system.opchda	system.twilio
system.eam	system.opcua	system.user
system.file	system.perspective	system.util
	system.print	
	system.report	

system.alarm

Alarm Functions

The following functions give you access to view and interact with the Alarm system in Ignition.

[In This Section ...](#)

system.alarm.acknowledge

This function is used in **Python Scripting**.

Description

Acknowledges any number of alarms, specified by their event ids. The event id is generated for an alarm when it becomes active, and it is used to identify a particular event from other events for the same source. The alarms will be acknowledged by the logged in user making the call. Additionally, acknowledgement notes may be included and will be stored along with the acknowledgement.

This function uses different parameters based on the scope of the script calling it. Both versions are listed below.

Client Permission Restrictions

Permission Type: Alarm Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax - Client Scripts

system.alarm.acknowledge(alarmIds, notes)

- Parameters

List[String] alarmIds - List of alarm event ids (UUIDs) to acknowledge.

String notes - A string that will be used as the Ack Note on each acknowledged alarm event. If set to **None**, then an Ack Note note will not be assigned to the alarm event.

- Returns

Nothing

- Scope

Vision Client

Syntax - Gateway Scripts

system.alarm.acknowledge(alarmIds, notes, username)

- Parameters

List[String] alarmIds - List of alarm event ids (UUIDs) to acknowledge.

String notes - A string that will be used as the Ack Note on each acknowledged alarm event. If set to **None**, then an Ack Note note will not be assigned to the alarm event.

String username - The user that acknowledged the alarm.

Note: This parameter is only used when called from a Gateway-scoped script. This parameter should be omitted from any Client-based scripts.

- Returns

Nothing

- Scope

Gateway, Perspective Session

Examples

Code Snippet - Acknowledging an Alarm in Client Scope

```
# This example shows the basic syntax for acknowledging an alarm from a Client-based script
system.alarm.acknowledge(['c27c06d8-698f-4814-af89-3c22944f58c5'],'Saw this alarm, did something about
it.')
```

Code Snippet - Acknowledging an Alarm in Gateway Scope

```
# This example shows the basic syntax for acknowledging an alarm from a Gateway-based script
system.alarm.acknowledge(['c27c06d8-698f-4814-af89-3c22944f58c5'],'Saw this alarm, did something about
it.', 'admin')
```

Code Snippet - Acknowledging Selected Alarms from a Table

```
# This code snippet could be used as a mouseReleased event handler on a Table component (not an Alarm
Status Table component)
# whose data was the return value of the system.alarm.queryStatus function.
# It presents a right-click menu to acknowledge the currently selected alarms (for more than one, the
# table must be set to allow multiple selection).
# This example does not ask for an ack message, and therefore might fail if the alarms we're attempting
# to acknowledge require notes.
# Also, note that the system will ignore any alarms that have already been acknowledged.

if event.button==3:
    rows = event.source.selectedRows
    data = event.source.data
    if len(rows)>0:
        uuids = [str(data.getValueAt(r,'EventId')) for r in rows]
        def ack(event, uuids=uuids):
            import system
            system.alarm.acknowledge(uuids, None)
        menu = system.gui.createPopupMenu({'Acknowledge':ack})
        menu.show(event)
```

Keywords

system alarm acknowledge, alarm.acknowledge

system.alarm.cancel

This function is used in **Python Scripting**.

Description

Cancels any number of alarms, specified by their event ids. The event id is generated for an alarm when it becomes active, and is used to identify a particular event from other events for the same source. The alarm will still be active, but will drop out of alarm pipelines.

Client Permission Restrictions

Permission Type: Alarm Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.alarm.cancel(alarmIds)

- Parameters

[List\[String\]](#) alarmIds - List of alarm event ids (UUIDs) to cancel.

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Examples

Code Snippet - Cancelling an Alarm

```
# This example shows the basic syntax for cancelling an alarm.  
  
system.alarm.cancel(['c27c06d8-698f-4814-af89-3c22944f58c5'])
```

Code Snippet - Cancelling All Currently Active Alarms

```
# To cancel all currently active alarms:  
  
ids = []  
results = system.alarm.queryStatus(state=["ActiveUnacked", "ActiveAcked"])  
for result in results:  
    id = result.getId()  
    ids.append(str(id))  
  
system.alarm.cancel(ids)
```

Keywords

system alarm cancel, alarm.cancel

system.alarm.createRoster

This function is used in **Python Scripting**.

Description

This function creates a new roster. Users may be added to the roster through the Gateway or the Roster Management component.

Client Permission Restrictions

Permission Type: Alarm Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.alarm.createRoster(name, description)

- Parameters

 String name - The name for the new roster.

 String description - An description for the new roster. Required, but can be blank.

- Returns

 Nothing

- Scope

 Gateway, Vision Client, Perspective Session

Code Examples

Code In Action - Creating a New Roster

```
# This example creates a new roster
name = 'MyRoster'
description = 'A roster created by scripting'
system.alarm.createRoster(name, description)
```

Keywords

system alarm createRoster, alarm.createRoster

system.alarm.getRosters

This function is used in [Python Scripting](#).

Description

This function returns a mapping of roster names to a list of usernames contained in the roster.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.alarm.getRosters()

- Parameters

Nothing

- Returns

[Dictionary\[String, List\[String\]\]](#) - A dictionary that maps roster names to a list of usernames in the roster. The list of usernames will be empty if no users have been added to the roster.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code In Action - Listing All the Users in a Roster

```
# This script will get all the rosters and list the users in them.  
rosters = system.alarm.getRosters()  
for key, values in rosters.iteritems():  
    # key is the roster name, values is a dict of usernames  
    print 'Roster', key, 'contains these users:'  
    for value in values:  
        print '  ', value
```

Output

```
Roster Admins contains these users:  
  admin  
Roster Supervisors contains these users:  
  asmith  
  jdoe
```

Keywords

system alarm getRosters, alarm.getRosters

system.alarm.getShelvedPaths

This function is used in [Python Scripting](#).

Description

Returns a list of the current shelved alarm paths. A "path" may be either a source path, or a display path.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.alarm.getShelvedPaths()

- Parameters
 - Nothing
- Returns
 - [List\[ShelvedPath\]](#) - A list of ShelvedPath objects. See [Scripting Object Reference](#).
- Scope
 - Gateway, Vision Client, Perspective Session

Examples

Code Snippet - Getting Paths for All Shelved Alarms

```
# The following code prints a list of the shelved alarms paths and prints them to the console.  
paths = system.alarm.getShelvedPaths()  
for p in paths:  
    print "Path: %s, Shelved by: %s, expires: %s, is expired? %s" % (p.getPath(), p.getUser(), p.  
getExpiration(), p.isExpired())
```

Keywords

system alarm getShelvedPaths, alarm.getShelvedPaths

system.alarm.listPipelines

This function is used in **Python Scripting**.

Description

Will return a list of the available Alarm Notification Pipelines in a project.

The legacy behavior of this function (7.9 and prior) did not have any parameters, and it would always check all projects for pipelines. See the [Upgrade Guide](#) for more details.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.alarm.listPipelines([projectName])

- Parameters

[String](#) projectName - The project to check alarm pipelines for. If omitted, will look for a project named "alarm-pipelines". [optional]

- Returns

[List\[String\]](#) - A list of pipeline names. The list will be empty if no pipelines exist. Unsaved name changes will not be reflected in the list.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code In Action - Listing the Alarm Pipelines in a Project

```
# This script will print a list of all alarm pipeline names in the "alarm-pipelines" project.
pipelines = system.alarm.listPipelines()
for pipeline in pipelines:
    print pipeline
```

Output

```
Emergency_Pipeline
Test
SMS_Pipeline
```

Keywords

system alarm listPipelines, alarm.listPipelines

system.alarm.queryJournal

This function is used in [Python Scripting](#).

Description

Queries the specified journal for historical alarm events. The result is a list of alarm events, which can be queried for individual properties.

[Click here](#) for more information on alarm properties

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

```
system.alarm.queryJournal([startDate], [endDate], [journalName], [priority], [state], [path], [source], [displaypath], [all_properties],  
[any_properties], [defined], [includeData], [includeSystem], [isSystem])
```

- Parameters

Date startDate - The start of the time range to query. Defaults to 8 hours previous to now if omitted. Time range is inclusive. [optional]

Date endDate - The end of the time range to query. Defaults to "now" if omitted. [optional]

String journalName - The journal name to query. If only one journal exists on the gateway, can be omitted. [optional]

List[String] priority - A list of possible priorities to match. Priorities can be specified by name or number, with the values: Diagnostic(0), Low(1), Medium(2), High(3), Critical(4). [optional]

List[String] state - A list of the event state types to match. Valid values are "ClearUnacked", "ClearAcked", "ActiveUnacked", and "ActiveAcked". [optional]

List[String] path - A list of possible source paths to search at. The wildcard "*" may be used. [optional]

List[String] source - A list of possible source paths to search at. The wildcard "*" may be used. [optional]

List[String] displaypath - A list of display paths to search at. Display paths are separated by "/", and if a path ends in "/*", everything below that path will be searched as well. [optional]

List[Tuple[String, String, Any]] all_properties - A set of property conditions, all of which must be met for the condition to pass. This parameter is a list of tuples, in the form ("propName", "condition", value). Valid condition values: "=", "!=",<,<=,>,>=". String values can only be compared using "=" and "!=" conditions. [optional]

List[Tuple[String, String, Any]] any_properties - A set of property conditions, any of which will cause the overall the condition to pass. This parameter is a list of tuples, in the form ("propName", "condition", value). Valid condition values: "=", "!=",<,<=,>,>=". String values can only be compared using "=" and "!=" conditions. [optional]

List[String] defined - A list of string property names, all of which must be present on an event for it to pass. [optional]

Boolean includeData - Whether or not event data should be included in the return. If this parameter is false, and if there are no conditions specified on associated data, the properties table will not be queried. [optional]

Boolean includeSystem - Specifies whether system events are included in the return. [optional]

Boolean isSystem - Specifies whether the returned event must or must not be a system event. [optional]

- Returns

AlarmQueryResult - The AlarmQueryResult object is a list of AlarmEvent objects with some additional helper methods. See [Scripting Object Reference](#).

Additionally, each AlarmEvent inside of the AlarmQueryResult object has several built-in methods to extract alarm information. More details on these methods can be found on the [Alarm Event Properties Reference](#) page.

Note: Each item in the resulting list is a separate alarm event: an alarm becoming active is one item, while the same alarm becoming acknowledged is a separate item. This differs from [system.alarm.queryStatus\(\)](#) which groups each event into a single item.

- Scope

Code Examples

Code Snippet - Querying the Alarm Journal

```
# This example shows the basic syntax for querying from the journal in a button's actionPerformed event,
with a date range selector ("Range"), storing the results back to a table called "Table":  
  
table = event.source.parent.getComponent("Table")
range= event.source.parent.getComponent("Range")  
  
results = system.alarm.queryJournal(journalName="Journal", startDate=range.startDate, endDate=range.
endDate)
table.data = results.getDataset()
```

Code Snippet - Querying the Alarm Journal With Filters

```
# This example extends the previous to only include non-acknowledged events of High or Critical severity,
who have associated data called "Department", set to "maintenance". It also excludes system events
(shelving notifications, etc):  
  
table = event.source.parent.getComponent("Table")
range= event.source.parent.getComponent("Range")  
  
results = system.alarm.queryJournal(journalName="Journal", startDate=range.startDate, endDate=range.
endDate, state=['ActiveUnacked', 'ClearUnacked'], all_properties=[("Department", "=", "maintenance")],
priority=["High", "Critical"], includeSystem=False)
table.data = results.getDataset()
```

Keywords

system alarm queryJournal, alarm.queryJournal

system.alarm.queryStatus

This function is used in [Python Scripting](#).

Description

Queries the current state of alarms. The result is a list of alarm events, which can be queried for individual properties.

Note: Depending on the number of alarm events in the system, this function can be fairly intensive and take a while to finish executing. Which can be problematic if the application is attempting to show the results on a component (such as using this function to retrieve a count of alarms). In these cases it's preferred to call this function in a gateway script of some sort (such as a timer script), and store the results in a tag, which can easily be accessed by a component binding.

The [Tag Properties](#) page contains more information on Alarm Properties

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

`system.alarm.queryStatus([priority], [state], [path], [source], [displaypath], [all_properties], [any_properties], [defined], [includeShelved])`

- Parameters

`List[String]` priority - A list of possible priorities to match. Priorities can be specified by name or number, with the values: Diagnostic(0), Low(1), Medium(2), High(3), Critical(4). [optional]

`List[String]` state - A list of states to allow. Valid values: "ClearUnacked", "ClearAcked", "ActiveUnacked", "ActiveAcked". [optional]

`List[String]` path - A list of possible source paths to search at. The wildcard "*" may be used. Works the same as the source argument, and either can be used. [optional]

`List[String]` source - A list of possible source paths to search at. The wildcard "*" may be used. Works the same as the path argument, and either can be used. [optional]

`List[String]` displaypath - A list of display paths to search at. Display paths are separated by "/", and if a path ends in "/*", everything below that path will be searched as well. [optional]

`List[Tuple[String, String, Any]]` all_properties - A set of property conditions, all of which must be met for the condition to pass. This parameter is a list of tuples, in the form ("propName", "condition", value). Valid condition values: "=", "!=",<,<=,>,>=". String values can only be compared using "=" and "!=" conditions. [optional]

`List[Tuple[String, String, Any]]` any_properties - A set of property conditions, any of which will cause the overall the condition to pass. This parameter is a list of tuples, in the form ("propName", "condition", value). Valid condition values: "=", "!=",<,<=,>,>=". String values can only be compared using "=" and "!=" conditions. [optional]

`List[String]` defined - A list of string property names, all of which must be present on an event for it to pass. [optional]

`Boolean` includeShelved - A flag indicating whether shelved events should be included in the results. Defaults to false. [optional]

- Returns

`AlarmQueryResult` - The AlarmQueryResult object is a list of AlarmEvent objects with some additional helper methods, see [Scripting Object Reference](#).

Additionally, each AlarmEvent inside of the AlarmQueryResult object has several built-in methods to extract alarm information. More details on these methods can be found on the [Alarm Event Properties Reference](#) page.

Note: Each item in the resulting list is a combination of each alarm event for the same alarm: details for when the alarm became active, acknowledged, and cleared are combined into a single item. This differs from [system.alarm.queryJournal\(\)](#) which splits these events into separate items

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Querying Alarm Status

```
# This example queries the state of all tags named "HiAlarm", and puts the results in a Vision table  
Component named "Table" (this assumes it's being run from a button on the same screen)  
# Note that this example is simple for the sake of brevity. Normally you'll want to use system.util.  
invokeAsynchronous to search for alarms in a separate thread, especially so if calling  
# this function from a component based script. See the next example for more information.  
  
table = event.source.parent.getComponent("Table")  
  
results = system.alarm.queryStatus(source=[ "*HiAlarm*" ])  
table.data = results.getDataset()
```

Code Snippet - Call queryStatus in a Separate Thread

```
# In this example we'll call system.alarm.queryStatus in a separate thread, and return the results to the  
data property on a Vision Table component. Similar to the example above.  
# What makes this example different is that it offers better performance when calling from a Vision  
component. Depending on the number of alarm events in the system, queryStatus  
# may take a significant amount of time to finish, which would lock up a Vision Client while the script  
is running in the GUI thread. Thus this example will use  
# system.util.invokeAsynchronous to call queryStatus in a separate thread, and then system.util.  
invokeLater make any changes to our components.  
  
# Define a function that will retrieve alarm data in a separate thread  
def getAlarms():  
    # Call queryStatus to retrieve the alarm data we're looking for, and store the results in a  
variable.  
    # In this case, we're looking for alarm events that contain the word "Sensor" in the source  
path.  
    results = system.alarm.queryStatus(source=[ "*Sensor*" ])  
  
    # From this same script, define a separate function that will later interact with the  
# GUI, allowing us to move our alarm data over to a component  
# We're also using the getDataset() function on the object returned by queryStatus,  
# since that will provide a dataset that our table component will expect.  
def setTheTable(alarms = results.getDataset()):  
  
    # Replace the property reference below with a path leading to whichever property  
    # you want to move the alarm data to.  
    event.source.parent.getComponent("Table").data = alarms  
  
    # The last thing we'll do in the separate thread is call invokeLater  
    # which will let our setTheTable function run in the GUI thread  
    system.util.invokeLater(setTheTable)  
  
# Call the getAlarms function in a separate thread, which starts the whole process  
system.util.invokeAsynchronous(getAlarms)
```

Code Snippet - Querying Alarm Status Using any_properties

```
# The any_properties parameter allows you to filter the results for specific properties. This is useful  
when searching for alarms that contain associated data.  
  
# Build a List of Tuples that represent the properties to search for. In this case, if our alarms have an  
Associated Data named 'Group', we can use  
# the following to search for potential values  
props = [ ("Group", "=", "value1"), ("Group", "=", "value2") ]  
state = [ "ActiveUnacked", "ActiveAcked" ]
```

```
alarms = system.alarm.queryStatus(any_properties = props, state = state)

# Here we're printing out the number of alarms that meet our criteria. We could replace this and further
# examine each individual alarm in a for-loop instead.
print len(alarms)
```

Keywords

system alarm queryStatus, alarm.queryStatus

system.alarm.shelve

This function is used in **Python Scripting**.

Description

This function shelves the specified alarms for the specified amount of time. The paths may be either source paths, or display paths. The time can be specified in minutes (timeoutMinutes) or seconds (timeoutSeconds). If an alarm is already shelved, this will overwrite the remaining time. If no timeout is specified, will default to 15 minutes.

Client Permission Restrictions

Permission Type: Alarm Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.alarm.shelve(path, [timeoutSeconds], [timeoutMinutes])

- Parameters

List[String] path - A list of possible source paths to search at. If a path ends in "*", the results will include anything below that path.

Integer timeoutSeconds - The amount of time to shelve the matching alarms for, specified in seconds. 0 indicates that matching alarm events should now be allowed to pass. [optional]

Integer timeoutMinutes - The amount of time to shelve the matching alarms for, specified in minutes. 0 indicates that matching alarm events should now be allowed to pass. [optional]

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code In Action - Shelving Alarms

```
# This example assumes that data has been loaded into a table ("Table") from system.alarm.queryStatus,
and it shelves the selected alarms for 5 minutes.
# It also assumes that it is being executed from a button's actionPerformed event.

table = event.source.parent.getComponent('Table')
rows = table.selectedRows
data = table.data
if len(rows)>0:
    sourcePaths = [str(data.getValueAt(r,'Source')) for r in rows]
    system.alarm.shelve(path=sourcePaths,timeoutMinutes=5)
```

Keywords

system alarm shelve, alarm.shelve

system.alarm.unshelve

This function is used in **Python Scripting**.

Client Permission Restrictions

Permission Type: Alarm Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.alarm.unshelve(path)

- Parameters

[List\[String\]](#) path - A list of possible source paths to search at. If a path ends in "/", the results will include anything below that path.

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system alarm unshelve, alarm.unshelve

system.bacnet

Functions

The following functions are used with the [BACnet driver](#) and a BACnet/IP device.

[In This Section ...](#)

system.bacnet.synchronizeTime

This function is used in [Python Scripting](#).

Description

Notifies the remote device of the correct current time, which is the system time (factoring in time zone and DST) of the server Ignition is running on.



To use this function, the [BACnet driver](#) must be installed.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.bacnet.synchronizeTime(deviceName)

- Parameters

`String deviceName`- The name of the configured BACnet/IP device instance to write from.

- Returns

Nothing

- Scope

Gateway

Code Examples

There are no examples associated with this scripting function.

Keywords

system bacnet synchronizeTime, bacnet.synchronizeTime

system.bacnet.synchronizeTimeUtc

This function is used in [Python Scripting](#).

Description

Notifies the remote device of the correct current time in UTC.



To use this function, the [BACnet driver](#) must be installed.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.bacnet.synchronizeTimeUtc(deviceName)

- Parameters

`String deviceName` - The name of the configured BACnet/IP device instance to write from.

- Returns

`Nothing`

- Scope

`Gateway`

Code Examples

There are no examples associated with this scripting function.

Keywords

`system bacnet synchronizeTimeUtc, bacnet.synchronizeTimeUtc`

system.bacnet.writeWithPriority

This function is used in **Python Scripting**.

Description

Write to the Present_Value attribute of an object with a custom priority level.



To use this function, the **BACnet** driver must be installed.

Client Permission Restrictions

This scripting function has no **Client Permission** restrictions.

Syntax

system.bacnet.writeWithPriority(deviceName, objectType, objectId, value, priority)

- Parameters

String deviceName - The name of the configured BACnet/IP device instance to write from.

Integer objectType - The numeric id of the objectType of the object instance being written to. See the objectType reference table below.

Integer objectId - The object instance number to write to.

Object value - The value to write. Clearing a value can be accomplished by writing a None value.

Integer priority - The priority level to write the value at. Must be in the range [1...16].

- Returns

Nothing

- Scope

Gateway

objectType Reference

Object	ID
Analog Input	0
Analog Output	1
Analog Value	2
Binary Input	3
Binary Output	4
Binary Value	5
Device	8
Large Analog Value	46
Multi-State Input	13
Multi-State Output	14
Multi-State Value	15

Code Examples

Example 1

```
# Write a value of 'True' to Binary Value 1 with a Priority of 7.

deviceName = 'BACnet Remote'
objectType = 5 # Binary Value
objectId = 1
value = True
priority = 7

system.bacnet.writeWithPriority(deviceName, objectType, objectId, value, priority)
```

Example 2

```
# Values can be cleared by writing a None value.

deviceName = 'BACnet Remote'
objectType = 5
objectId = 1
value = None
priority = 7

system.bacnet.writeWithPriority(deviceName, objectType, objectId, value, priority)
```

Keywords

```
system bacnet writeWithPriority, bacnet.writeWithPriority
```

system.dataset

Dataset Functions

The following functions give you access to view and interact with datasets.

[In This Section ...](#)

system.dataset.addColumn

This function is used in [Python Scripting](#).

Description

Takes a dataset and returns a new dataset with a new column added or inserted into it. If the columnIndex argument is omitted, the column will be appended to the end of the dataset.

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See [Altering a Dataset](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.addColumn(dataset, [colIndex], col, colName, colType)

- Parameters

Dataset dataset - The starting dataset. Please be aware that this dataset will not actually be modified (datasets are immutable), but rather will be the starting point for creating a new dataset.

Integer colIndex - The index (starting at 0) at which to insert the new column. Will throw an IndexError if less than zero or greater than the length of the dataset. If omitted, the new column will be appended to the end. [optional]

List[Any] col - A Python sequence representing the data for the new column. Its length must equal the number of rows in the dataset.

String colName - The name of the column.

Type colType - The type of the column. The type can be a Python builtin type, such as `str`, `int`, `float`, or a Java class, such as `java.util.Date`.

- Returns

Dataset - A new dataset with the new column inserted or appended.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will work on a Button component on a Vision window, given two Vision bar charts with
# default values.
# The script takes the dataset from Bar Chart 1, adds a column of integers called Center Area to the end
# of the existing data,
# and displays the new dataset in Bar Chart 2.

ds1 = event.source.parent.getComponent('Bar Chart 1').data
colCount = ds1.getColumnCount()
columnName = "Center Area"
columnData = []
for i in range(ds1.getRowCount()):
    columnData.append(i* 10)
```

```
ds2 = system.dataset.addColumn(ds1, colCount, columnData, columnName, int)
event.source.parent.getComponent('Bar Chart 2').data = ds2
```

Keywords

system dataset addColumn, dataset.addColumn

system.dataset.addRow

This function is used in [Python Scripting](#).

Description

Takes a dataset and returns a new dataset with a new row added or inserted into it. If the rowIndex argument is omitted, the row will be appended to the end of the dataset.

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See [Altering a Dataset](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.addRow(dataset, [rowIndex], row)

- Parameters

Dataset dataset - The starting dataset. Please be aware that this dataset will not actually be modified (datasets are immutable), but rather will be the starting point for creating a new dataset.

Integer rowIndex - The index (starting at 0) at which to insert the new row. Will throw an IndexError if less than zero or greater than the length of the dataset. If omitted, the new row will be appended to the end. [optional]

List[Any] row - A Python sequence representing the data for the new row. Its length must equal the number of columns in the dataset.

- Returns

Dataset - A new dataset with the new row inserted or appended.

- Scope

Gateway, Vision Client, Perspective Session

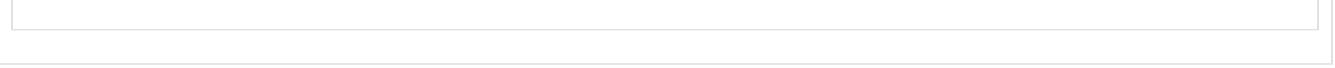
Code Examples

Code Snippet

```
# This example would add a new option to a Vision Dropdown List by adding a row to its underlying dataset.  
# Notice how the last line assigns the return value of the addRow function to the dropdown's data property.  
  
# This script should be on a button that is a sibling to a Dropdown List component named "Dropdown".  
dropdown = event.source.parent.getComponent("Dropdown")  
newRow = [5, "New Option"]  
dropdown.data = system.dataset.addRow(dropdown.data, newRow)
```

Code Snippet

```
# This snippet would add a new option into a Dropdown component just like above, but at the beginning:  
dropdown = event.source.parent.getComponent("Dropdown")  
newRow = [5, "New Option"]  
dropdown.data = system.dataset.addRow(dropdown.data, 0, newRow)
```



Keywords

system dataset addRow, dataset.addRow

system.dataset.addRows

This function is used in [Python Scripting](#).

Description

Takes a dataset and returns a new dataset with new rows added or inserted into it. If the rowIndex argument is omitted, the rows will be appended to the end of the dataset.

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See [Altering a Dataset](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.addRows(dataset, [rowIndex], rows)

- Parameters

Dataset dataset - The starting dataset. Please be aware that this dataset will not actually be modified (datasets are immutable), but rather will be the starting point for creating a new dataset.

Integer rowIndex - The index (starting at 0) at which to insert the new row. Will throw an IndexError if less than zero or greater than the length of the dataset. If omitted, the new row will be appended to the end. [optional]

List[Any] rows - A Python sequence of sequences representing the data for the new rows. The length of each sequence must equal the number of columns in the dataset.

- Returns

Dataset - A new dataset with the new rows inserted or appended.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example adds new options to a Vision Dropdown List by adding a row to its underlying dataset.  
# Note that the last line assigns the return value of the addRows function to the dropdown's data  
# property.  
  
dropdown = event.source.parent.getComponent("Dropdown")  
newRow = [[5, "New Option"], [6, "Another New Option"]]  
dropdown.data = system.dataset.addRows(dropdown.data, newRow)
```

Code Snippet

```
# This snippet adds new options into a Dropdown component just like above, but at the beginning:  
  
dropdown = event.source.parent.getComponent("Dropdown")  
newRow = [[5, "New Option"], [6, "Another New Option"]]  
dropdown.data = system.dataset.addRows(dropdown.data, 0, newRow)
```

Keywords

system dataset addRows, dataset.addRows

system.dataset.appendDataset

This function is used in [Python Scripting](#).

Description

Takes two different datasets and returns a new dataset with the second dataset appended to the first. Will throw an error if the number and types of columns do not match.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.appendDataset(dataset1, dataset2)

- Parameters

[Dataset](#) dataset1 - The dataset that will come first in the returned dataset.

[Dataset](#) dataset2 - The second dataset that will be appended to the end in the returned dataset.

- Returns

[Dataset](#) - A new dataset that is a combination of the original two datasets.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example pulls in the datasets from two different Vision table components, combines them, then
# writes the results to a third table.
# The two source datasets must have the same number and types of columns.

data1 = event.source.parent.getComponent('Table 1').data
data2 = event.source.parent.getComponent('Table 2').data
comboData = system.dataset.appendDataset(data1, data2)
event.source.parent.getComponent('Table 3').data = comboData
```

Keywords

system dataset appendDataset, dataset.appendDataset

system.dataset.clearDataset

This function is used in **Python Scripting**.

Description

Takes a dataset and returns a new dataset with all of the same column names and types, but no rows.

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See [Altering a Dataset](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.clearDataset(dataset)

- Parameters

Dataset dataset - The starting dataset.

- Returns

Dataset - A new dataset with no rows.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example pulls in the dataset from a Vision Table component, clears it, then writes the empty
# dataset back to the table.

data = event.source.parent.getComponent('Table').data
event.source.parent.getComponent('Table').data = system.dataset.clearDataset(data)
```

Keywords

system dataset clearDataset, dataset.clearDataset

system.dataset.dataSetToHTML

This function is used in [Python Scripting](#).

Description

Formats the contents of a dataset as an HTML page, returning the results as a string. Uses the <table> element to create a data table page.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.dataSetToHTML(showHeaders, dataset, title)

- Parameters

Boolean showHeaders - If true, the HTML table will include a header row.

Dataset dataset - The dataset to export

String title - The title for the HTML page.

- Returns

String - The HTML page as a string.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This snippet would run a SQL query against a database, and turn the results into a string containing
# HTML. It then writes the string to a file on the local hard drive.
```

```
results = system.db.runNamedQuery("Fetch Records",{})
html = system.dataset.dataSetToHTML(1, results, "Production Report")
filePath = "C:\\\\output\\\\results.html"
system.file.writeFile(filePath, html)
```

Keywords

system dataset dataSetToHTML, dataset.dataSetToHTML

system.dataset.deleteRow

This function is used in **Python Scripting**.

Description

Takes a dataset and returns a new dataset with a row removed.

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See [Altering a Dataset](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.deleteRow(dataset, rowIndex)

- Parameters

Dataset dataset - The starting dataset. Please be aware that this dataset will not actually be modified (datasets are immutable), but rather will be the starting point for creating a new dataset.

Integer rowIndex - The index (starting at 0) of the row to delete. Will throw an IndexError if less than zero or greater than the row count of the dataset -1.

- Returns

Dataset - A new dataset with the specified row removed.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example would remove the selected row from a Vision List component, by re-assigning the List's data property to the new dataset returned by the deleteRow function.
```

```
myList = event.source.parent.getComponent("List")
row = myList.selectedIndex
if row != -1: # make sure there is something selected
    myList.data = system.dataset.deleteRow(myList.data, row)
```

Keywords

system dataset deleteRow, dataset.deleteRow

system.dataset.deleteRows

This function is used in **Python Scripting**.

Description

Takes a dataset and returns a new dataset with one or more rows removed.

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See [Altering a Dataset](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.deleteRows(dataset, rowIndices)

- Parameters

Dataset dataset - The starting dataset. Please be aware that this dataset will not actually be modified (datasets are immutable), but rather will be the starting point for creating a new dataset.

List[Integer] rowIndices - The indices (starting at 0) of the rows to delete. Will throw an IndexError if any element is less than zero or greater than the number of rows in the dataset - 1.

- Returns

Dataset - A new dataset with the specified rows removed.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example would remove several rows from a Vision Table component, by re-assigning the Table's data property to the new dataset returned by the deleteRows function.
```

```
ds = event.source.parent.getComponent('Table').data
rows = [0,2,3,4]
ds = system.dataset.deleteRows(ds, rows)
event.source.parent.getComponent('Table').data = ds
```

Keywords

system dataset deleteRows, dataset.deleteRows

system.dataset.exportCSV

This function is used in [Python Scripting](#).

Description

Exports the contents of a dataset as a CSV file, prompting the user to save the file to disk. To write silently to a file, you cannot use the [dataset.export*](#) functions. Instead, use the [toCSV\(\)](#) function.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.exportCSV(filename, showHeaders, dataset)

- Parameters

String filename - A suggested filename to save as.

Boolean showHeaders - If true, the CSV file will include a header row.

Dataset dataset - The dataset to export.

- Returns

String - The path to the saved file, or None if the action was canceled by the user.

- Scope

Vision Client

Code Examples

Code Snippet

```
# This snippet would prompt the user to save the data currently displayed in a Table component to a CSV file, and would open the file (in an external program, presumably Excel) after a successful save.
```

```
table = event.source.parent.getComponent("Table")
filePath = system.dataset.exportCSV("data.csv", 1, table.data)
if filePath != None:
    system.net.openURL("file:///"+filePath.replace('\\','/'))
```

Keywords

system dataset exportCSV, dataset.exportCSV

system.dataset.exportExcel

This function is used in **Python Scripting**.

Description

Exports the contents of a dataset as an Excel spreadsheet, prompting the user to save the file to disk. Uses the same format as the [system.dataset.toExcel](#) function. To write silently to a file, you cannot use the [dataset.export*](#) functions. Instead, use the [toExcel\(\)](#) function.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.exportExcel(filename, showHeaders, dataset, [nullsEmpty])

- Parameters

String filename - A suggested filename to save as.

Boolean showHeaders - If true, the spreadsheet will include a header row.

List[Dataset] dataset - A sequence of datasets, one for each sheet in the resulting workbook.

Boolean nullsEmpty - If True, the spreadsheet will leave cells with NULL values empty, instead of allowing Excel to provide a default value like 0. Defaults to False. [optional]

- Returns

String - The path to the saved file, or None if the action was canceled by the user.

- Scope

Vision Client

Code Examples

Code Snippet

```
# This snippet prompts the user to save the data currently displayed in a Table component to an Excel-compatible spreadsheet file. It opens the file after a successful save.
```

```
table = event.source.parent.getComponent("Table")
filePath = system.dataset.exportExcel("data.xls", 1, table.data)
if filePath != None:
    system.net.openURL("file://" + filePath)
```

Keywords

system dataset exportExcel, dataset.exportExcel

system.dataset.exportHTML

This function is used in [Python Scripting](#).

Description

Exports the contents of a dataset to an HTML page. Prompts the user to save the file to disk.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.exportHTML(filename, showHeaders, dataset, title)

- Parameters

String filename - A suggested filename to save as.

Boolean showHeaders - If true, the HTML table will include a header row.

Dataset dataset - The dataset to export.

String title - The title for the HTML page.

- Returns

String - The path to the saved file, or None if the action was canceled by the user.

- Scope

Vision Client

Code Examples

Code Snippet

```
# This snippet prompts the user to save the data currently displayed in a Vision Table component to an HTML file, and opens the file in the default web browser after a successful save.
```

```
table = event.source.parent.getComponent("Table")
filePath = system.dataset.exportHTML("data.html", 1, table.data, "Production Report")
if filePath != None:
    system.net.openURL("file://"+filePath)
```

Keywords

system dataset exportHTML, dataset.exportHTML

system.dataset.filterColumns

This function is used in **Python Scripting**.

Description

Takes a dataset and returns a view of the dataset containing only the columns found within the given list of columns.

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See [Altering a Dataset](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.filterColumns(dataset, columns)

- Parameters

Dataset dataset - The starting dataset.

List[Integer] | List[String] columns - A list of columns to keep in the returned dataset. The columns may be in integer index form (starting at 0), or the name of the columns as strings.

- Returns

Dataset - A new dataset containing the filtered columns.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example takes the dataset from a four column Bar Chart and displays a subset of the data in two
separate tables.
# This is performed in a button component actionPerformed script.

chartData = event.source.parent.getComponent('Bar Chart').data

northSouth = [0, 1] # North Area, South Area cols
eastWest = ["East Area", "West Area"]

filteredData = system.dataset.filterColumns(chartData, northSouth)
event.source.parent.getComponent('NorthSouthTable').data = filteredData

filteredData = system.dataset.filterColumns(chartData, eastWest)
event.source.parent.getComponent('EastWestTable').data = filteredData
```

Keywords

system dataset filterColumns, dataset.filterColumns

system.dataset.formatDates

This function is used in [Python Scripting](#).

Description

Returns a new dataset with Date columns as strings formatted according to the dateFormat specified.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.formatDates(dataset, dateFormat, locale)

- Parameters

[Dataset](#) dataset - The starting dataset to format.

[String](#) dateFormat - A valid Java DateFormat string, representing how the date should be formatted. For example: "yyyy-MM-dd HH:mm:ss". Refer to [Data Type Formatting Reference](#) for more information on the valid characters.

[Locale](#) locale - The Locale to use for formatting. The Locale object is actually any of Java's Locales, which can be found [here](#). The java Locale library must be imported, and the Locale must be defined in all caps. See the second example below for an idea of how that works. If no Locale is specified, the default Locale will be used. [optional]

- Returns

[Dataset](#) - A new dataset, containing the formatted dates.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example takes the dataset from a table component and formats the dates to look like Fri, Jan 22, 2018.
```

```
data = event.source.parent.getComponent('Table').data
formattedData = system.dataset.formatDates(data, "EEE, MMM d, yyyy")
```

Code Snippet

```
# This example formats the date similarly to the last example, but uses the Italian Locale, which causes the dates to be formatted with the Locale.
```

```
from java.util import Locale

data = event.source.parent.getComponent('Table').data
locale = Locale.ITALY
formattedDataFr = system.dataset.formatDates(data, "yyyy-MM-dd HH:mm:ss", locale)
```

Keywords

system dataset formatDates, dataset.formatDates

system.dataset.fromCSV

This function is used in [Python Scripting](#).

Description

Converts a dataset stored in a CSV formatted string to a dataset that can be immediately assignable to a dataset property in your project. Usually this is used in conjunction with [system.file.readFileAsString](#) when reading in a CSV file that was exported using [system.dataset.toCSV](#). The CSV string must be formatted in a specific way:

```
#NAMES
Col 1,Col 2,Col 3
#TYPES
I,str,D
#ROWS,6
44,Test Row 2,1.8713151369491254
86,Test Row 3,97.4913421614675
0,Test Row 8,20.39722542161364
78,Test Row 9,34.57127071614745
20,Test Row 10,76.41114659745085
21,Test Row 13,13.880548366871926
```

The first line must be #NAMES

The second line must list the names of the columns of the dataset separated by commas

The third line must be #TYPES

The fourth line must list the type of each column of the dataset in order, separated by commas

```
Integer = I
String = str
Double = D
Date = date
Long = L
Short = S
Float = F
Boolean = B
```

The fifth line must be #ROWS followed by a comma and then the number of rows of data (i.e. #ROWS, 6)

The following lines will be your data, each column value separated by a comma; each row on a separate line. The number of rows must match what was specified on line 5

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.fromCSV(csv)

- Parameters
 - String** **csv** - A string holding a CSV dataset.
- Returns
 - Dataset** - A new dataset.
- Scope

Code Examples

Code Snippet

```
# In this example it is assumed that the CSV file being read was a dataset  
#that was previously exported using system.dataset.toCSV:  
# Specify file path.  
file_path = "C:\\my_dataset.csv"  
# Read in the file as a string.  
data_string = system.file.readFileAsString(file_path)  
# Convert the string to a dataset and store in a variable.  
data = system.dataset.fromCSV(data_string)  
# Assign the dataset to a table.  
event.source.parent.getComponent('Table').data = data
```

Keywords

system dataset fromCSV, dataset.fromCSV

system.dataset.getColumnHeaders

This function is used in [Python Scripting](#).

Description

Takes in a dataset and returns the headers as a Python list.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.getColumnHeaders(dataset)

- Parameters
 - [Dataset](#) dataset - The input dataset.
- Returns
 - [List](#) - A list of column header strings.
- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example fetches the dataset from a Vision table, and prints the table headers as a list.  
# Fetch data from table component.  
data = event.source.parent.getComponent('Table').data  
# Print dataset headers.  
print system.dataset.getColumnHeaders(data)
```

Keywords

system dataset getColumnHeaders, dataset.getColumnHeaders

system.dataset.setValue

This function is used in **Python Scripting**.

Description

Takes a dataset and returns a new dataset with one value altered.

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See [Altering a Dataset](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.setValue(dataset, rowIndex, columnName, value)

- Parameters

Dataset dataset - The starting dataset. Will not be modified (datasets are immutable), but acts as the basis for the returned dataset.

Integer rowIndex - The index of the row to set the value at (starting at 0).

String columnName - The name of the column to set the value at. Case insensitive.

Any value - The new value for the specified row/column.

- Returns

Dataset - A new dataset, with the new value set at the given location.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This snippet could be used for a Vision Button's actionPerformed event to change the selected cell's
# value in a Table component to zero.
# Fetch table reference
table = event.source.parent.getComponent("Table")
# Fetch selected row and column
selRow = table.getSelectedRow()
selCol = table.getSelectedColumn()
# If row and column have been selected, update value in table to 0.
if selRow != -1 and selCol != -1:
    newData = system.dataset.setValue(table.data, selRow, selCol, 0.0)
    table.data = newData
```

Keywords

system dataset setValue, dataset.setValue

system.dataset.sort

This function is used in **Python Scripting**.

Description

Takes a dataset and returns a sorted version of dataset. The sort order is determined by a single column. This works on numeric, as well as alphanumeric columns. When sorting alphanumeric, contiguous numbers are treated as a single number: you may recognize this as a "natural sort".

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See [Altering a Dataset](#).

Alphanumeric Sort

The table below represents an example of how alphanumeric values are sorted by the function. Where **Raw Column Values** represents an initial set of values, and the **Sorted** columns show how the function sorts in **Ascending** and **Descending** order.

Raw Column Values	Sorted - Ascending	Sorted - Descending
a1	a1	Z3
a22	A1	z3
Z3	a4	a77z99
z3	a7z9	a77z4
a4	a22	a22
a77z4	a77z4	a7z9
a77z99	a77z99	a4
a7z9	Z3	a1
A1	z3	A1

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.sort(dataset, keyColumn, [ascending])

- Parameters

[Dataset](#) dataset - The dataset to sort.

[Integer](#) keyColumn - The index of the column to sort on.

[Boolean](#) ascending - True for ascending order, False for descending order. If omitted, ascending order will be used. [optional]

- Returns

[Dataset](#) - A new sorted dataset.

- Scope

Gateway, Vision Client, Perspective Session

Syntax

system.dataset.sort(dataset, keyColumn, [ascending])

- Parameters

[Dataset](#) dataset - The dataset to sort.

[String](#) keyColumn - The column name of the column to sort on.

[Boolean](#) ascending - True for ascending order, False for descending order. If omitted, ascending order will be used. [optional]

- Returns

[Dataset](#) - A new sorted dataset.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code will take the data in a Vision Table component, sort it based on the column with index 1,  
# and then reinsert the sorted data into the same Table.
```

```
data = event.source.parent.getComponent('Table').data  
newData = system.dataset.sort(data, 1)  
event.source.parent.getComponent('Table').data = newData
```

Code Snippet

```
# This code will create a dataset in scripting, and then sort it based on the name of one of the columns.  
# It then inserts the sorted dataset into a table component.
```

```
# Initialize column headers and empty data list  
headers = ["City", "Population", "Timezone", "GMTOffset"]  
data = []  
# Add rows, one by one, into data list  
data.append(["New York", 8363710, "EST", -5])  
data.append(["Los Angeles", 3833995, "PST", -8])  
data.append(["Chicago", 2853114, "CST", -6])  
data.append(["Houston", 2242193, "CST", -6])  
data.append(["Phoenix", 1567924, "MST", -7])  
# Convert headers and data lists into dataset  
cities = system.dataset.toDataSet(headers, data)  
# Sort the resulting dataset by city name  
newData = system.dataset.sort(cities, "City")  
# Write final dataset to a table  
event.source.parent.getComponent('Table').data = newData
```

Keywords

system dataset sort, dataset.sort

system.dataset.toCSV

This function is used in [Python Scripting](#).

Description

Formats the contents of a dataset as CSV (comma separated values), returning the resulting CSV as a string. If the "forExport" flag is set, then the format will be appropriate for parsing using the [system.dataset.fromCSV](#) function.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.toCSV(dataset, showHeaders, forExport, localized)

- Parameters

[Dataset](#) dataset - The dataset to export to CSV.

[Boolean](#) showHeaders - If set to true, a header row will be present in the CSV. Default is true.

[Boolean](#) forExport - If set to true, extra header information will be present in the CSV data which is necessary for the CSV to be compatible with the fromCSV method. Overrides showHeaders. Default is false.

[Boolean](#) localized - If set to true, the string representations of the values in the CSV data will be localized. Default is false.

- Returns

[String](#) - The CSV data as a string.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This snippet runs a SQL query against a database and turns the results into a CSV string. It would then store the resulting CSV to a file on the local hard drive.
```

```
results = system.db.runNamedQuery("Fetch Records",{})
csv = system.dataset.toCSV(dataset = results, showHeaders = True, forExport = False)
filePath = "C:\\\\output\\\\results.csv"
system.file.writeFile(filePath, csv)
```

Keywords

system dataset toCSV, dataset.toCSV

system.dataset.toDataSet

This function is used in **Python Scripting**.

Description

This function is used to convert PyDataSets to DataSets, and to create new datasets from raw Python lists. When creating a new dataset, headers should have unique names.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.toDataSet(dataset)

- Parameters

[PyDataset](#) dataset - A PyDataset object to convert.

- Returns

[Dataset](#) - The newly created dataset.

- Scope

Gateway, Vision Client, Perspective Session

Syntax

system.dataset.toDataSet(headers, data)

- Parameters

[List\[String\]](#) headers - The column names for the dataset to create.

[List\[Any\]](#) data - A list of rows for the new dataset. Each row must have the same length as the headers list, and each value in a column must be the same type.

- Returns

[Dataset](#) - The newly created dataset.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example create a single column dataset.  
header = ['myColumn']  
rows = [[1], [2]]  
dataset = system.dataset.toDataSet(header, rows)
```

Code Snippet

```
# This second example shows how this function can be used to convert from a PyDataSet (which is what  
system.db.runQuery returns) to a normal DataSet, which is the data type of a Table component's data
```

```
property.

pyDataSet = system.db.runQuery("SELECT * FROM example1 LIMIT 100")
table = event.source.parent.getComponent("Table")
normalDataSet = system.dataset.toDataSet(pyDataSet)
table.data = normalDataSet
```

Code Snippet

```
# This third example shows how to use this function to create a new dataset out of a Python sequence that
you have filled in. In this case, the sequence is created via a for loop appending rows to a list.

# Generate the Rows.
rows = []
for x in range(10):
    oneRow = ["Row " + str(x), x]
    rows.append(oneRow)

# Generate the Dataset.
headers = ["Row Name", "Value"]
data = system.dataset.toDataSet(headers, rows)

# Use our new dataset to fill in a Table.
table = event.source.parent.getComponent("Table")
table.data = data
```

Keywords

system dataset toDataSet, dataset.toDataSet

system.dataset.toExcel

This function is used in **Python Scripting**.

Description

Formats the contents of one or more datasets as an Excel spreadsheet, returning the results as a byte array. Each dataset specified will be added as a worksheet in the Excel workbook.

This function replaces the deprecated [system.dataset.dataSetToExcel](#) function.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.toExcel(showHeaders, dataset, [nullsEmpty], [sheetNames])

- Parameters

Boolean showHeaders - If True, the spreadsheet will include a header row. If False, the header row will be omitted.

List[Dataset] dataset - A sequence of one or more datasets, one for each sheet in the resulting workbook.

Boolean nullsEmpty - If True, the spreadsheet will leave cells with NULL values empty, instead of allowing Excel to provide a default value like 0. Defaults to False. [optional]

List sheetNames - Expects a list of strings, where each string is a name for one of the datasets. When used, there must be an equal number of string names in `sheetName` as there are datasets in the `dataset` parameter. Names provided in this parameter may be sanitized into acceptable Excel sheet names. [optional]

- Returns

Array - A byte array representing an Excel workbook.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This snippet would run a SQL query against a database, and turn the results into a string that is XML  
# that Excel can open.  
# It then writes the string to a file on the local hard drive.  
  
results = system.db.runNamedQuery("Fetch Records",{})  
spreadsheet = system.dataset.toExcel(True, [results])  
filePath = "C:\\\\output\\\\results.xlsx"  
system.writeFile(filePath, spreadsheet)
```

Keywords

system dataset toExcel, dataset.toExcel

system.dataset.toPyDataSet

This function is used in [Python Scripting](#).

Description

This function converts from a normal dataSet to a PyDataset, which is a wrapper class which makes working with datasets more Python-esque. For more information on datasets and PyDatasets, see the [Datasets](#) page.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.toPyDataSet(dataset)

- Parameters

[Dataset](#) dataset - A dataset object to convert into a PyDataset.

- Returns

[PyDataset](#) - The newly created PyDataset.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example script would be added to a button that is in the same container as the table you are
working with.
# It grabs the data of the Table component, and adds up the values in the column named "Value",
displaying the result to the user.

# Get a Table component's data.
table = event.source.parent.getComponent("Table")
data = system.dataset.toPyDataSet(table.data)

# Loop through the data, summing the Value column.
value = 0.0
for row in data:
    value += row["Value"]

# Show the user the sum of the Value column.
system.gui.messageBox("The value is: %f" % value)
```

Keywords

system dataset toPyDataSet, dataset.toPyDataSet

system.dataset.updateRow

This function is used in [Python Scripting](#).

Description

Takes a dataset and returns a new dataset with a one row altered. To alter the row, this function takes a Python dictionary to represent the changes to make to the specified row. The keys in the dictionary are used to find the columns to alter.

Note: Datasets are immutable, which means they cannot be directly modified once created. Instead, this scripting function returns a new dataset with some modification applied, which must be assigned to a variable to be used. See [Altering a Dataset](#).

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.dataset.updateRow(dataset, rowIndex, changes)

- Parameters

Dataset `dataset` - The starting dataset. Will not be modified (datasets are immutable), but acts as the basis for the returned dataset.

Integer `rowIndex` - The index of the row to update (starting at 0).

Dictionary[String, Any] `changes` - A dictionary of changes to make. Their keys in the dictionary should match column names in the dataset, and their values will be used to update the row.

- Returns

Dataset - A new dataset with the values at the specified row updated according to the values in the dictionary.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example could be used to dynamically change the data that an Easy Chart displays.  
# In this simple example, we assume that the chart is always configured to display a single tank's level.  
# This script updates the pen being displayed using a dynamic tank number.  
  
# Generate new tag name and tag path.  
tankNumber = 5  
newName = "Tank %d Level" % tankNumber  
newPath = "Tanks/Tank %d/Level" % tankNumber  
  
# Consolidate changes into a dictionary.  
updates = { "NAME": newName, "TAG_PATH": newPath}  
  
# Update the Easy Chart.  
chart = event.source.parent.getComponent("Easy Chart")  
newPens = system.dataset.updateRow(chart.tagPens, 0, updates)  
chart.tagPens = newPens
```

Keywords

system dataset updateRow, dataset.updateRow

system.date

Date Functions

The following functions give you access to test and modify dates.

In This Section ...

system.date.*Between Functions	system.date.get* Functions
<code>system.date.millisBetween</code> <code>system.date.secondsBetween</code> <code>system.date.minutesBetween</code> <code>system.date.hoursBetween</code> <code>system.date.daysBetween</code> <code>system.date.weeksBetween</code> <code>system.date.monthsBetween</code> <code>system.date.yearsBetween</code>	<code>system.date.getMillis</code> <code>system.date.getSecond</code> <code>system.date.getMinute</code> <code>system.date.getHour12</code> <code>system.date.getHour24</code> <code>system.date.getDayOfWeek</code> <code>system.date.getDayOfMonth</code> <code>system.date.getDayOfYear</code> <code>system.date.getMonth</code> <code>system.date.getQuarter</code> <code>system.date.getYear</code> <code>system.date.getAMorPM</code>
system.date.add* Functions	
<code>system.date.addMillis</code> <code>system.date.addSeconds</code> <code>system.date.addMinutes</code> <code>system.date.addHours</code> <code>system.date.addDays</code> <code>system.date.addWeeks</code> <code>system.date.addMonths</code> <code>system.date.addYears</code>	

system.date.*Between

This function is used in **Python Scripting**.

Description

This function is a set of functions that include:

Function	Description
system.date.millisBetween	Calculates the number of whole milliseconds between two dates.
system.date.secondsBetween	Calculates the number of whole seconds between two dates.
system.date.minutesBetween	Calculates the number of whole minutes between two dates.
system.date.hoursBetween	Calculates the number of whole hours between two dates.
system.date.daysBetween	Calculates the number of whole days between two dates. Daylight savings changes are taken into account.
system.date.weeksBetween	Calculates the number of whole weeks between two dates.
system.date.monthsBetween	Calculates the number of whole months between two dates. Daylight savings changes are taken into account.
system.date.yearsBetween	Calculates the number of whole years between two dates. Daylight savings changes are taken into account.

Order does matter for the two dates passed in that we are calculating how much time has passed from date 1 to date 2. So, if date 2 is further in time than date 1, then a positive amount of time has passed. If date 2 is backwards in time from date 1, then a negative amount of time has passed.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.date.*between(date_1, date_2)

- Parameters

Date date_1 - The first date to use.

Date date_2 - The second date to use.

- Returns

Integer - An integer that is representative of the difference between two dates.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example would grab the current time, and add 119 minutes to it, then calculate the number
# of hours between the two dates.

first = system.date.now()
second = system.date.addMinutes(first, 119)
print system.date.hoursBetween(first, second) # This would print 1 since it is only 1 whole hour.
```

Code Snippet

```
# This example would create two date objects, one on the 28th of May,  
# and one on the 22nd of April, both in 2020. Because the second date is  
# before the first date, a negative number will be returned.  
  
first = system.date.getDate(2020, 4, 28)  
second = system.date.getDate(2020, 3, 22)  
print system.date.daysBetween(first, second) # This will print -36
```

Code Snippet

```
# This example can be placed on the action performed event of a button.  
# It will then grab the week difference of two calendar components,  
# and enter the value returned into a numeric text field.  
  
first = event.source.parent.getComponent('Start Date Calendar').date  
second = event.source.parent.getComponent('End Date Calendar').date  
event.source.parent.getComponent('Numeric Text Field').intValue = system.date.weeksBetween(first, second)
```

Keywords

```
system date *Between, date.*Between, date millisBetween, date.millisBetween, date secondsBetween, date.secondsBetween, date.minutesBetween,  
system.date.minutesBetween, date hoursBetween, date.hoursBetween, date weeksBetween, date.weeksBetween, date monthsBetween, date.  
monthsBetween, date yearsBetween, date.yearsBetween
```

system.date.add*

This function is used in **Python Scripting**.

Description

This function is a set of functions to add and subtract time that include:

Function	Description
system.date.addMillis	Add or subtract an amount of milliseconds to a given date and time.
system.date.addSeconds	Add or subtract an amount of seconds to a given date and time.
system.date.addMinutes	Add or subtract an amount of minutes to a given date and time.
system.date.addHours	Add or subtract an amount of hours to a given date and time.
system.date.addDays	Add or subtract an amount of days to a given date and time.
system.date.addWeeks	Add or subtract an amount of weeks to a given date and time.
system.date.addMonths	Add or subtract an amount of months to a given date and time. This function is unique since each month can have a variable number of days. For example, if the date passed in is March 31st, and we add one month, April does not have a 31st day, so the returned date will be the proper number of months rounded down to the closest available day, in this case April 30th.
system.date.addYears	Add or subtract an amount of years to a given date and time.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.date.add*(date, value)

- Parameters

Date date - The starting date.

Integer value - The number of units to add, or subtract if the value is negative.

- Returns

Date - A new date object offset by the integer passed to the function.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example would add two days to the date passed in.
```

```
today = system.date.now()
twoDaysFromToday = system.date.addDays(today, 2)
```

Code Snippet

```
# This example would subtract 20 minutes from a date object we create.
# Even though our original date starts on the 28th, it starts at midnight,
# so subtracting 20 minutes puts it at the previous day.
```

```
date = system.date.getDate(2020, 5, 25) # This would print out like Mon May 25 00:00:00 PDT 2020
print system.date.addMinutes(date, -20) # This will print Sun May 24 23:40:00 PDT 2020
```

Code Snippet

```
# This example can be placed on the property change script of one Vision Calendar component.
# It will then automatically set a second calendar component two weeks in advanced of the first calendars
selected date.
if event.propertyName == "date":
    date = event.newValue
    event.source.parent.getComponent('End Date Calendar').date = system.date.addWeeks(date, 2)
```

Keywords

```
system date add*, date.add*, date addMillis, date.addMillis, date addSeconds, date.addSeconds, date addMinutes, date.addMinutes, date
addHours, date.addHours, date addDays, date.addDays, date addWeeks, date.addWeeks, date addMonths, date.addMonths, date addYears
```

system.date.format

This function is used in **Python Scripting**.

Description

Returns the given date as a string and formatted according to a pattern. The pattern is a format that is full of various placeholders that display different parts of the date. These are case-sensitive. These placeholders can be repeated for a different effect. For example, M will give you 1-12, MM will give you 01-12, MMM will give you Jan-Dec, MMMM will give you January-December.

Refer to [Data Type Formatting Reference](#) for a table of placeholders.

Note: This function uses the Java class [java.text.SimpleDateFormat](#) internally, and it will accept any valid format string for that class.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.date.format(date, format)

- Parameters

[Date](#) date - The date to format.

[String](#) format - A format string such as "yyyy-MM-dd HH:mm:ss". The format argument is optional. The default is "yyyy-MM-dd HH:mm:ss".

- Returns

[String](#) - A string representing the formatted datetime.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example would format the current date to look like "01/01/01"

today = system.date.now()
print system.date.format(today, "yy/MM/dd")
#This printed 16/04/01
```

Code Snippet

```
# This example would format the current date to look like "2001-01-31 16:59:59"
# This is a standard format that all databases recognize.

today = system.date.now()
print system.date.format(today, "yyyy-MM-dd HH:mm:ss")
```

Keywords

system date format, date.format

system.date.fromMillis

This function is used in [Python Scripting](#).

Description

Creates a date object given a millisecond value.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.date.fromMillis(millis)

- Parameters

[Long](#) millis- The number of milliseconds elapsed since January 1, 1970, 00:00:00 UTC (GMT).

- Returns

[Date](#) - A new date object.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will print out the date "Fri Aug 18 14:35:25 PDT 2017"  
print system.date.fromMillis(1503092125000)
```

Keywords

system date fromMillis, date.fromMillis

system.date.get*

This function is used in **Python Scripting**.

Description

This function is a set of functions that include:

Function	Description
system.date.getMillis	Extracts the milliseconds from a date, ranging from 0-999.
system.date.getSecond	Extracts the second from a date, ranging from 0-59.
system.date.getMinute	Extracts the minutes from a date, ranging from 0-59.
system.date.getHour12	Extracts the hour from a date. Uses a 12 hour clock, so noon and midnight are returned as 0.
system.date.getHour24	Extracts the hour from a date. Uses a 24 hour clock, so midnight is zero.
system.date.getDayOfWeek	Extracts the day of the week from a date. Sunday is day 1, Saturday is day 7.
system.date.getDayOfMonth	Extracts the day of the month from a date. The first day of the month is day 1.
system.date.getDayOfYear	Extracts the day of the year from a date. The first day of the year is day 1.
system.date.getMonth	Extracts the month from a date, where January is month 0.
system.date.getQuarter	Extracts the quarter from a date, ranging from 1-4.
system.date.getYear	Extracts the year from a date.
system.date.getAMorPM	Returns a 0 if the time is before noon, and a 1 if the time is after noon.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.date.get*(date)

- Parameters

Date date - The date to use.

- Returns

Integer - An integer that represents the extracted value.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example would grab the current time and print the current month.

date = system.date.now()
print system.date.getMonth(date) #This would print the current month.
```

Code Snippet

```
# This example would create a date object and print out the quarter of that date.  
  
date = system.date.getDate(2017, 3, 15) #This would print "Mon April 15 00:00:00 PDT 2016"  
print system.date.getQuarter(date) # This will print 2
```

Code Snippet

```
# This example can be placed on the action performed event of a button.  
# It will then grab the day of the week of the calendar component,  
# and enter the value returned into a numeric text field.  
  
date = event.source.parent.getComponent('Calendar').date  
event.source.parent.getComponent('Numeric Text Field').intValue = system.date.getDayOfWeek(date)
```

Keywords

```
system date get*, date.get*, date.getMillis, date.getMillis, date.getSeconds, date.getSeconds, date.getMinutes, date.getMinutes, date.getHours12, date.getHours12, date.getHours24, date.getHours24, date.getDayOfWeek, date.getDayOfWeek, date.getDayOfMonth, date.getDayOfMonth, date.getDayOfYear, date.getDayOfYear, date.getMonth, date.getMonth, date.getQuarter, date.getQuarter, date.getYear, date.getYear, date.getAMorPM, date.getAMorPM
```

system.date.getDate

This function is used in [Python Scripting](#).

Description

Creates a new Date object given a year, month and a day. The time will be set to midnight of that day.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.date.getDate(year, month, day)

- Parameters

[Integer](#) year - The year for the new date.

[Integer](#) month - The month of the new date. January is month 0.

[Integer](#) day - The day of the month for the new date. The first day of the month is day 1.

- Returns

[Date](#) - A new date, set to midnight of that day.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example creates a new date object set to January 1st, 2021.  
  
date = system.date.getDate(2021, 0, 1)  
print date
```

Keywords

system date getDate, date.getDate

system.date.getTimezone

This function is used in **Python Scripting**.

Description

Returns the ID of the current timezone.

*This list is subject to change depending on the exact version of java that is installed.

Africa/Abidjan
Africa/Accra
Africa/Addis_Ababa
Africa/Algiers
Africa/Asmara
Africa/Asmera
Africa/Bamako
Africa/Bangui
Africa/Banjul
Africa/Bissau
Africa/Blantyre
Africa/Brazzaville
Africa/Bujumbura
Africa/Cairo
Africa/Casablanca
Africa/Ceuta
Africa/Conakry
Africa/Dakar
Africa/Dar_es_Salaam
Africa/Djibouti
Africa/Douala
Africa/El_Aaiun
Africa/Freetown
Africa/Gaborone
Africa/Harare
Africa/Johannesburg
Africa/Juba
Africa/Kampala
Africa/Khartoum
Africa/Kigali
Africa/Kinshasa
Africa/Lagos
Africa/Libreville
Africa/Lome
Africa/Luanda
Africa/Lubumbashi
Africa/Lusaka
Africa/Malabo
Africa/Maputo
Africa/Maseru
Africa/Mbabane
Africa/Mogadishu
Africa/Monrovia
Africa/Nairobi
Africa/Ndjamena
Africa/Niamey
Africa/Nouakchott
Africa/Ouagadougou
Africa/Porto-Novo
Africa/Sao_Tome
Africa/Timbuktu
Africa/Tripoli
Africa/Tunis
Africa/Windhoek
America/Adak
America/Anchorage
America/Anguilla
America/Antigua
America/Araguaina
America/Argentina/Buenos_Aires
America/Argentina/Catamarca
America/Argentina/ComodRivadavia
America/Argentina/Cordoba
America/Argentina/Jujuy

America/Argentina/La_Rioja
America/Argentina/Mendoza
America/Argentina/Rio_Gallegos
America/Argentina/Salta
America/Argentina/San_Juan
America/Argentina/San_Luis
America/Argentina/Tucuman
America/Argentina/Ushuaia
America/Aruba
America/Asuncion
America/Atikokan
America/Atka
America/Bahia
America/Bahia_Banderas
America/Barbados
America/Belem
America/Belize
America/Blanc-Sablon
America/Boa_Vista
America/Bogota
America/Boise
America/Buenos_Aires
America/Cambridge_Bay
America/Campo_Grande
America/Cancun
America/Caracas
America/Catamarca
America/Cayenne
America/Cayman
America/Chicago
America/Chihuahua
America/Coral_Harbour
America/Cordoba
America/Costa_Rica
America/Creston
America/Cuiaba
America/Curacao
America/Danmarkshavn
America/Dawson
America/Dawson_Creek
America/Denver
America/Detroit
America/Dominica
America/Edmonton
America/Eirunepe
America/El_Salvador
America/Ensenada
America/Fort_Wayne
America/Fortaleza
America/Glace_Bay
America/Godthab
America/Goose_Bay
America/Grand_Turk
America/Grenada
America/Guadeloupe
America/Guatemala
America/Guayaquil
America/Guyana
America/Halifax
America/Havana
America/Hermosillo
America/Indiana/Indianapolis
America/Indiana/Knox
America/Indiana/Marengo
America/Indiana/Petersburg
America/Indiana/Tell_City
America/Indiana/Vevay
America/Indiana/Vincennes
America/Indiana/Winamac
America/Indianapolis
America/Inuvik
America/Iqaluit
America/Jamaica
America/Jujuy
America/Juneau
America/Kentucky/Louisville
America/Kentucky/Monticello
America/Knox_IN
America/Kralendijk

America/La_Paz
America/Lima
America/Los_Angeles
America/Louisville
America/Lower_Princes
America/Maceio
America/Managua
America/Manaus
America/Marigot
America/Martinique
America/Matamoros
America/Mazatlan
America/Mendoza
America/Menominee
America/Merida
America/Metlakatla
America/Mexico_City
America/Miquelon
America/Moncton
America/Monterrey
America/Montevideo
America/Montreal
America/Montserrat
America/Nassau
America/New_York
America/Nipigon
America/Nome
America/Noronha
America/North_Dakota/Beulah
America/North_Dakota/Center
America/North_Dakota/New_Salem
America/Ojinaga
America/Panama
America/Pangnirtung
America/Paramaribo
America/Phoenix
America/Port-au-Prince
America/Port_of_Spain
America/Porto_Acre
America/Porto_Velho
America/Puerto_Rico
America/Rainy_River
America/Rankin_Inlet
America/Recife
America/Regina
America/Resolute
America/Rio_Branco
America/Rosario
America/Santa_Isabel
America/Santarem
America/Santiago
America/Santo_Domingo
America/Sao_Paulo
America/Scoresbysund
America/Shiprock
America/Sitka
America/St_Barthelemy
America/St_Johns
America/St_Kitts
America/St_Lucia
America/St_Thomas
America/St_Vincent
America/Swift_Current
America/Tegucigalpa
America/Thule
America/Thunder_Bay
America/Tijuana
America/Toronto
America/Tortola
America/Vancouver
America/Virgin
America/Whitehorse
America/Winnipeg
America/Yakutat
America/Yellowknife
Antarctica/Casey
Antarctica/Davis
Antarctica/DumontDUrville
Antarctica/Macquarie

Antarctica/Mawson
Antarctica/McMurdo
Antarctica/Palmer
Antarctica/Rothera
Antarctica/South_Pole
Antarctica/Syowa
Antarctica/Troll
Antarctica/Vostok
Arctic/Longyearbyen
Asia/Aden
Asia/Almaty
Asia/Amman
Asia/Anadyr
Asia/Aqtau
Asia/Aqtobe
Asia/Ashgabat
Asia/Ashkhabad
Asia/Baghdad
Asia/Bahrain
Asia/Baku
Asia/Bangkok
Asia/Beirut
Asia/Bishkek
Asia/Brunei
Asia/Calcutta
Asia/Chita
Asia/Choibalsan
Asia/Chongqing
Asia/Chungking
Asia/Colombo
Asia/Dacca
Asia/Damascus
Asia/Dhaka
Asia/Dili
Asia/Dubai
Asia/Dushanbe
Asia/Gaza
Asia/Harbin
Asia/Hebron
Asia/Ho_Chi_Minh
Asia/Hong_Kong
Asia/Hovd
Asia/Irkutsk
Asia/Istanbul
Asia/Jakarta
Asia/Jayapura
Asia/Jerusalem
Asia/Kabul
Asia/Kamchatka
Asia/Karachi
Asia/Kashgar
Asia/Kathmandu
Asia/Katmandu
Asia/Khandyga
Asia/Kolkata
Asia/Krasnoyarsk
Asia/Kuala_Lumpur
Asia/Kuching
Asia/Kuwait
Asia/Macao
Asia/Macau
Asia/Magadan
Asia/Makassar
Asia/Manila
Asia/Muscat
Asia/Nicosia
Asia/Novokuznetsk
Asia/Novosibirsk
Asia/Omsk
Asia/Oral
Asia/Phnom_Penh
Asia/Pontianak
Asia/Pyongyang
Asia/Qatar
Asia/Qyzylorda
Asia/Rangoon
Asia/Riyadh
Asia/Saigon
Asia/Sakhalin

Asia/Samarkand
Asia/Seoul
Asia/Shanghai
Asia/Singapore
Asia/Srednekolymsk
Asia/Taipei
Asia/Tashkent
Asia/Tbilisi
Asia/Tehran
Asia/Tel_Aviv
Asia/Thimbu
Asia/Thimphu
Asia/Tokyo
Asia/Ujung_Pandang
Asia/Ulaanbaatar
Asia/Ulan_Bator
Asia/Urumqi
Asia/Ust-Nera
Asia/Vientiane
Asia/Vladivostok
Asia/Yakutsk
Asia/Yekaterinburg
Asia/Yerevan
Atlantic/Azores
Atlantic/Bermuda
Atlantic/Canary
Atlantic/Cape_Verde
Atlantic/Faeroe
Atlantic/Faroe
Atlantic/Jan_Mayen
Atlantic/Madeira
Atlantic/Reykjavik
Atlantic/South_Georgia
Atlantic/St_Helena
Atlantic/Stanley
Australia/ACT
Australia/Adelaide
Australia/Brisbane
Australia/Broken_Hill
Australia/Canberra
Australia/Currie
Australia/Darwin
Australia/Eucla
Australia/Hobart
Australia/LHI
Australia/Lindeman
Australia/Lord_Howe
Australia/Melbourne
Australia/NSW
Australia/North
Australia/Perth
Australia/Queensland
Australia/South
Australia/Sydney
Australia/Tasmania
Australia/Victoria
Australia/West
Australia/Yancowinna
Brazil/Acre
Brazil/DeNoronha
Brazil/East
Brazil/West
CET
CST6CDT
Canada/Atlantic
Canada/Central
Canada/East-Saskatchewan
Canada/Eastern
Canada/Mountain
Canada/Newfoundland
Canada/Pacific
Canada/Saskatchewan
Canada/Yukon
Chile/Continental
Chile/EasterIsland
Cuba
EET
EST5EDT
Egypt

Eire
Etc/GMT
Etc/GMT+0
Etc/GMT+1
Etc/GMT+10
Etc/GMT+11
Etc/GMT+12
Etc/GMT+2
Etc/GMT+3
Etc/GMT+4
Etc/GMT+5
Etc/GMT+6
Etc/GMT+7
Etc/GMT+8
Etc/GMT+9
Etc/GMT-0
Etc/GMT-1
Etc/GMT-10
Etc/GMT-11
Etc/GMT-12
Etc/GMT-13
Etc/GMT-14
Etc/GMT-2
Etc/GMT-3
Etc/GMT-4
Etc/GMT-5
Etc/GMT-6
Etc/GMT-7
Etc/GMT-8
Etc/GMT-9
Etc/GMT0
Etc/Greenwich
Etc/UCT
Etc/UTC
Etc/Universal
Etc/Zulu
Europe/Amsterdam
Europe/Andorra
Europe/Athens
Europe/Belfast
Europe/Belgrade
Europe/Berlin
Europe/Bratislava
Europe/Brussels
Europe/Bucharest
Europe/Budapest
Europe/Busingen
Europe/Chisinau
Europe/Copenhagen
Europe/Dublin
Europe/Gibraltar
Europe/Guernsey
Europe/Helsinki
Europe/Isle_of_Man
Europe/Istanbul
Europe/Jersey
Europe/Kaliningrad
Europe/Kiev
Europe/Lisbon
Europe/Ljubljana
Europe/London
Europe/Luxembourg
Europe/Madrid
Europe/Malta
Europe/Mariehamn
Europe/Minsk
Europe/Monaco
Europe/Moscow
Europe/Nicosia
Europe/Oslo
Europe/Paris
Europe/Podgorica
Europe/Prague
Europe/Riga
Europe/Rome
Europe/Samara
Europe/San_Marino
Europe/Sarajevo
Europe/Simferopol

Europe/Skopje
Europe/Sofia
Europe/Stockholm
Europe/Tallinn
Europe/Tirane
Europe/Tiraspol
Europe/Uzhgorod
Europe/Vaduz
Europe/Vatican
Europe/Vienna
Europe/Vilnius
Europe/Volgograd
Europe/Warsaw
Europe/Zagreb
Europe/Zaporozhye
Europe/Zurich
GB
GB-Eire
GMT
GMT0
Greenwich
Hongkong
Iceland
Indian/Antananarivo
Indian/Chagos
Indian/Christmas
Indian/Cocos
Indian/Comoro
Indian/Kerguelen
Indian/Mahe
Indian/Maldives
Indian/Mauritius
Indian/Mayotte
Indian/Reunion
Iran
Israel
Jamaica
Japan
Kwajalein
Libya
MET
MST7MDT
Mexico/BajaNorte
Mexico/BajaSur
Mexico/General
NZ
NZ-CHAT
Navajo
PRC
PST8PDT
Pacific/Apia
Pacific/Auckland
Pacific/Bougainville
Pacific/Chatham
Pacific/Chuuk
Pacific/Easter
Pacific/Efate
Pacific/Enderbury
Pacific/Fakaofa
Pacific/Fiji
Pacific/Funafuti
Pacific/Galapagos
Pacific/Gambier
Pacific/Guadalcanal
Pacific/Guam
Pacific/Honolulu
Pacific/Johnston
Pacific/Kiritimati
Pacific/Kosrae
Pacific/Kwajalein
Pacific/Majuro
Pacific/Marquesas
Pacific/Midway
Pacific/Nauru
Pacific/Niue
Pacific/Norfolk
Pacific/Noumea
Pacific/Pago_Pago
Pacific/Palau

Pacific/Pitcairn
Pacific/Pohnpei
Pacific/Ponape
Pacific/Port_Moresby
Pacific/Rarotonga
Pacific/Saipan
Pacific/Samoa
Pacific/Tahiti
Pacific/Tarawa
Pacific/Tongatapu
Pacific/Truk
Pacific/Wake
Pacific/Wallis
Pacific/Yap
Poland
Portugal
ROK
Singapore
SystemV/AST4
SystemV/AST4ADT
SystemV/CST6
SystemV/CST6CDT
SystemV/EST5
SystemV/EST5EDT
SystemV/HST10
SystemV/MST7
SystemV/MST7MDT
SystemV/PST8
SystemV/PST8PDT
SystemV/YST9
SystemV/YST9YDT
Turkey
UCT
US/Alaska
US/Aleutian
US/Arizona
US/Central
US/East-Indiana
US/Eastern
US/Hawaii
US/Indiana-Starke
US/Michigan
US/Mountain
US/Pacific
US/Pacific-New
US/Samoa
UTC
Universal
W-SU
WET
Zulu
EST
HST
MST
ACT
AET
AGT
ART
AST
BET
BST
CAT
CNT
CST
CTT
EAT
ECT
IET
IST
JST
MIT
NET
NST
PLT
PNT
PRT
PST
SST
VST

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.date.getTimezone()

- Parameters

Nothing

- Returns

[String](#) - A representation of the current time zone.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will print out your current Timezone ID.  
# If your Client and Gateway are in different time zones, the returned value will be  
# dependent on where this script is run.  
# IE: in a button on a client will return the Client time zone. On a Gateway script will  
# return the Gateway time zone.  
  
print system.date.getTimezone()
```

Keywords

system date getTimezone, date.getTimezone

system.date.getTimezoneOffset

This function is used in **Python Scripting**.

Description

Returns the current time zone's offset versus UTC for a given instant, taking Daylight Saving Time into account.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.date.getTimezoneOffset([date])

- Parameters
 - Date** date- The instant in time for which to calculate the offset. Uses now() if omitted. [optional]
- Returns
 - Double** - The time zone offset compared to UTC, in hours.
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will give the time zone offset using the date February 22, 2021
# and the computers current time zone.

date = system.date.getDate(2021, 1, 22)
print system.date.getTimezoneOffset(date) # returns -8.0 (if you are in Pacific Daylight Time)
```

Keywords

system date getTimezoneOffset, date.getTimezoneOffset

system.date.getTimezoneRawOffset

This function is used in [Python Scripting](#).

Description

Returns the current time zone offset versus UTC, not taking Daylight Saving Time into account.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.date.getTimezoneRawOffset()

- Parameters
 - Nothing
- Returns
 - [Double](#) - The time zone offset.
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example gives the raw timezone offset (ignoring Daylight Saving Time) for the computers current time zone.

print system.date.getTimezoneRawOffset() # returns -8.0 (if you are in the Pacific time zone)
```

Keywords

system date getTimezoneRawOffset, date.getTimezoneRawOffset

system.date.isAfter

This function is used in [Python Scripting](#).

Description

Compares two dates to see if date_1 is after date_2.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.date.isAfter(date_1, date_2)

- Parameters

[Date](#) date_1 - The first date.

[Date](#) date_2 - The second date.

- Returns

[Boolean](#) - True if date_1 is after date_2, false otherwise.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This checks to see if the first date is after the second date, which it is.

first = system.date.getDate(2018, 4, 28)
second = system.date.getDate(2018, 3, 22)
print system.date.isAfter(first, second) #Will print true.
```

Keywords

system date isAfter, date.isAfter

system.date.isBefore

This function is used in [Python Scripting](#).

Description

Compares two dates to see if date_1 is before date_2.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.date.isBefore(date_1, date_2)

- Parameters

[Date](#) date_1 - The first date.

[Date](#) date_2 - The second date.

- Returns

[Boolean](#) - True if date_1 is before date_2, false otherwise.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This checks to see if the first date is before the second date, which it is not.  
  
first = system.date.getDate(2018, 4, 28)  
second = system.date.getDate(2018, 3, 22)  
print system.date.isBefore(first, second) #Will print false.
```

Keywords

system date isBefore, date.isBefore

system.date.isBetween

This function is used in **Python Scripting**.

Description

Compares a target date with two other dates; checkes to see if the target date is between the other two dates.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.date.isBetween(target_date, start_date, end_date)

- Parameters

Date target_date - The date to compare.

Date start_date - The start of a date range.

Date end_date - The end of a date range. This date must be after the start date.

- Returns

Boolean - True if target_date is \geq start_date and target_date \leq end_date, false otherwise.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This compares if the first date is between the other dates, which it is not.  
# Note that if the end date is before the start date, this function will always return false.  
  
target = system.date.getDate(2017, 4, 28)  
start = system.date.getDate(2017, 3, 22)  
end = system.date.getDate(2017, 4, 22)  
print system.date.isBetween(target, start, end) #Will print false.
```

Keywords

system date isBetween, date.isBetween

system.date.isDaylightTime

This function is used in **Python Scripting**.

Description

Checks to see if the current time zone is using Daylight Saving Time during the date specified.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.date.isDaylightTime([date])

- Parameters
 - Date** date - The date you want to check if the current time zone is observing Daylight Saving Time. Uses now() if omitted. [optional]
- Returns
 - Boolean** - True if date is observing Daylight Saving Time in the current time zone; false otherwise.
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
date = system.date.getDate(2018, 6, 28)
print system.date.isDaylightTime(date) #Will print True in the US Pacific time zone.
```

Keywords

system date isDaylightTime, date.isDaylightTime

system.date.midnight

This function is used in [Python Scripting](#).

Description

Returns a copy of a date with the hour, minute, second, and millisecond fields set to zero.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.date.midnight(date)

- Parameters
 - Date** date- The starting date.
- Returns
 - Date** - A new date, set to midnight of the day provided.
- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example prints out the current date with the time set to midnight.

date = system.date.now()
print system.date.midnight(date)
```

Keywords

system date midnight, date.midnight

system.date.now

This function is used in **Python Scripting**.

Description

Returns a java.util.Date object that represents the current time according to the local system clock.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.date.now()

- Parameters
Nothing
- Returns
[Date](#) - A new date, set to the current date and time.
- Scope
Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will set a calendar component to the current date and time.  
event.source.parent.getComponent('Calendar').date = system.date.now()
```

Keywords

system date now, date.now

system.date.parse

This function is used in **Python Scripting**.

Description

Attempts to parse a string and create a date. Causes ParseException if the date dateString parameter is in an unrecognized format.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.date.parse(dateString[, formatString][, locale])

- Parameters

[String](#) dateString - The string to parse into a date.

[String](#) formatString - Format string used by the parser. Default is "yyyy-MM-dd HH:mm:ss". Refer to [Data Type Formatting Reference](#). [optional]

[Locale](#) | [String](#) locale - Locale used for parsing. Can be the locale name such as 'fr', or the Java Locale such as 'Locale.French'. Default is 'Locale.English'. Refer to [Java Locale](#). [optional]

- Returns

[Date](#) - The parsed date.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will return a date object set to the given date and time, May 28th, 1992 at 4:22am.  
print system.date.parse('May 28, 1992 4:22', 'MMMM dd, yyyy hh:mm')
```

Code Example - Using the Locale Parameter

```
# This example demonstrates the locale parameter to parse a French date  
print system.date.parse("juillet 15, 2015 10:32:15", "MMMM dd, yyyy hh:mm:ss", "fr")  
  
# If using the Java Locale, then you must import from the Locale class. The following example parses a German  
date.  
from java.util import Locale  
print system.date.parse('21-Februar-2017 04:22:00', 'dd-MMMM-yyyy HH:mm:ss', Locale.GERMAN)
```

Keywords

system date parse, date.parse

system.date.setTime

This function is used in **Python Scripting**.

Description

Takes in a date and returns a copy of it with the time fields set as specified.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.date.setTime(date, hour, minute, second)

- Parameters

Date date - The starting date.

Integer hour - The hours (0-23) to set.

Integer minute - The minutes (0-59) to set.

Integer second - The seconds (0-59) to set.

- Returns

Date - A new date, set to the appropriate time.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example will set the date object to the current date with the time set to 01:37 in the morning and 44 seconds.

date = system.date.getDate(2018, 6, 29) #getDate is zero based, so a month parameter of 6 will return July.
print system.date.setTime(date, 1, 37, 44) #This will print Fri July 29 01:37:44 PDT 2018
```

Keywords

system date setTime, date.setTime

system.date.toMillis

This function is used in **Python Scripting**.

Description

Converts a date object to its millisecond value elapsed since January 1, 1970, 00:00:00 UTC (GMT).

Syntax

system.date.toMillis(date)

- Parameters

Date date - The date object to convert.

- Returns

Integer- An 8-byte integer representing the number of millisecond elapsed since January 1, 1970, 00:00:00 UTC (GMT).

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example takes the date Fri Aug 18 14:35:25 PDT 2017,  
# and prints it out 1503092125000.  
  
date = system.date.getDate(2017, 7, 18)  
datetime = system.date.setTime(date, 14, 35, 25)  
print system.date.toMillis(datetime)
```

Keywords

system date toMillis, date.toMillis

system.db

Database Functions

The following functions give you access to view and modify data in the database.

[In This Section ...](#)

system.db.addDatasource

This function is used in **Python Scripting**.

Description

Adds a new database connection in Ignition.

Client Permission Restrictions

Permission Type: Datasource Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

```
system.db.addDatasource(jdbcDriver, name, description, [connectUrl], [username], [password], [props], [validationQuery], [maxConnections])
```

- Parameters

String jdbcDriver - The name of the JDBC driver configuration to use. Available options are based off the JDBC driver configurations on the the Gateway.

String name - The datasource name.

String description - Description of the datasource. [optional]

String connectUrl - Default is the connect URL for JDBC driver. [optional]

String username - Username to login to the datasource with. [optional]

String password - Password for the login. [optional]

String props - The extra connection parameters. [optional]

String validationQuery - Default is the validation query for the JDBC driver. [optional]

Integer maxConnections - Default is 8. [optional]

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Adding a MySQL Database to a Gateway

```
system.db.addDatasource(jdbcDriver="myJDBCDriver", name="NewDatabase",
connectURL="jdbc:mysql://localhost:3306/test", username="root",
password="password", props="zeroDateTimeBehavior=convertToNull;")
```

Keywords

system db addDatasource, db.addDatasource

system.db.beginNamedQueryTransaction

This function is used in [Python Scripting](#).

Description

Begins a new database transaction using [Named Queries](#). Database transactions are used to execute multiple queries in an atomic fashion. After executing queries, you must either commit the transaction to have your changes take effect or rollback the transaction, which will make all operations since the last commit not take place. The transaction is given a new unique string code, which is then returned. You can then use this code as the tx argument for other `system.db.*` function calls to execute various types of queries using this transaction.

An open transaction consumes one database connection until it is closed. Because leaving connections open indefinitely would exhaust the connection pool, each transaction is given a timeout. Each time the transaction is used, the timeout timer is reset. For example, if you make a transaction with a timeout of one minute, you must complete that transaction within a minute. If a transaction is detected to have timed out, it will be automatically closed and its transaction id will no longer be valid.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax - Vision

`system.db.beginNamedQueryTransaction([database], [isolationLevel], [timeout])`

- Parameters

`String` database - The name of the database connection to create a transaction in. If omitted, uses the project's default connection.

`Integer` isolationLevel - The transaction isolation level to use. Use one of the four constants: system.db.READ_COMMITTED, system.db.READ_UNCOMMITTED, system.db.REPEATABLE_READ, or system.db.SERIALIZABLE. If omitted, uses system.db.READ_COMMITTED. [optional]

`Integer` timeout - The amount of time, in milliseconds, that this connection is allowed to remain open without being used. Timeout counter is reset any time a query or call is executed against the transaction, or when committed or rolled-back. If omitted, defaults to 30,000. [optional]

- Returns

`String` - The new transaction ID. You'll use this ID as the "tx" argument for all other calls to have them execute against this transaction.

- Scope

Vision Client

Syntax - Perspective and Gateway

`system.db.beginNamedQueryTransaction(project, database, [isolationLevel], [timeout])`

- Parameters

`String` project - The name of the project that contains the named query.

`String` database - The name of the database connection to create a transaction in.

`Integer` isolationLevel - The transaction isolation level to use. Use one of the four constants: system.db.READ_COMMITTED, system.db.READ_UNCOMMITTED, system.db.REPEATABLE_READ, or system.db.SERIALIZABLE. If omitted, uses system.db.READ_COMMITTED. [optional]

`Integer` timeout - The amount of time, in milliseconds, that this connection is allowed to remain open without being used. Timeout counter is reset any time a query or call is executed against the transaction, or when committed or rolled-back. If omitted, defaults to 30,000. [optional]`Integer` timeout - The amount of time, in milliseconds, that this connection is allowed to remain open without being used. Timeout counter is reset any time a query or call is executed against the transaction, or when committed or rolled-back. If omitted, defaults to 30,000. [optional]

- Returns

`String` - The new transaction ID. You'll use this ID as the "tx" argument for all other calls to have them execute against this transaction.

- Scope

Isolation Level Values

The following table lists each value of the isolationLevel parameter and its associated level. Either the integer value or constant may be passed. Note that some JDBC drivers only support some levels, so the driver's documentation should be consulted. Isolation levels are well documented online, but the following link is a great starting point: [Data Concurrency and Consistency](#)

Isolation Level	Int Value	Constant
Read Uncommitted	1	system.db.READ_UNCOMMITTED
Read Committed	2	system.db.READ_COMMITTED
Repeatable Read	4	system.db.REPEATABLE_READ
Serializable	8	system.db.SERIALIZABLE

Code Examples**Code Snippet - Running Named Query Using Named Query Transactions**

```
# This example starts a transaction and checks a screen to see if the transaction should be completed or
reversed (rolled back).
# The example assumes you have several components on screen and a Named Query that takes in an ID and a
string value.

# Get details from the screen: Numeric Text Field, Text Field, Checkbox
idEntry = event.source.parent.getComponent('ID Field').intValue
valueEntry = event.source.parent.getComponent('Value Field').text
shouldRollback = event.source.parent.getComponent('CheckBox').selected

# Begin the transaction.
datasource = "MYSQL"
isolationLevel = system.db.READ_COMMITTED
timeout = 60000
txNumber = system.db.beginNamedQueryTransaction(datasource, isolationLevel, timeout)

# Start by running a Named Query against the transaction.
namedQueryPath = "InsertQueries/AddValues"
params = {"id":idEntry, "value":valueEntry}
system.db.runNamedQuery(namedQueryPath, params, txNumber)

# Check the window to see if the user selected to cancel the transaction.
if shouldRollback:
    # cancel the transaction
    system.db.rollbackTransaction(txNumber)
    print "Transaction rolled back"
else:
    # complete the transaction
    system.db.commitTransaction(txNumber)
    print "Transaction committed"

# Close the transaction now that we are done.
system.db.closeTransaction(txNumber)
```

Keywords

system db beginNamedQueryTransaction, db.beginNamedQueryTransaction

system.db.beginTransaction

This function is used in **Python Scripting**.

Description

Begins a new database transaction for using run* and runPrep* queries. Database transactions are used to execute multiple queries in an atomic fashion. After executing queries, you must either commit the transaction to have your changes take effect, or rollback the transaction which will make all operations since the last commit not take place. The transaction is given a new unique string code, which is then returned. You can then use this code as the tx argument for other system.db.* function calls to execute various types of queries using this transaction.

An open transaction consumes one database connection until it is closed. Because leaving connections open indefinitely would exhaust the connection pool, each transaction is given a timeout. Each time the transaction is used, the timeout timer is reset. For example, if you make a transaction with a timeout of one minute, you must use that transaction at least once a minute. If a transaction is detected to have timed out, it will be automatically closed and its transaction id will no longer be valid.

Client Permission Restrictions

Permission Type: Legacy Database Access

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.db.beginTransaction(database, isolationLevel, timeout)

- Parameters

String database - The name of the database connection to create a transaction in. Use "" for the project's default connection.

Integer isolationLevel - The transaction isolation level to use. Use one of the four constants: system.db.READ_COMMITTED, system.db.READ_UNCOMMITTED, system.db.REPEATABLE_READ, or system.db.SERIALIZABLE.

Integer timeout - The amount of time, in milliseconds, that this connection is allowed to remain open without being used. Timeout counter is reset any time a query or call is executed against the transaction, or when committed or rolled-back.

- Returns

String - The new transaction ID. You'll use this ID as the "tx" argument for all other calls to have them execute against this transaction.

- Scope

Gateway, Vision Client, Perspective Session

Isolation Level Values

The following table lists each value of the isolationLevel parameter and its associated level. Either the integer value or constant may be passed. Note that some JDBC drivers only support some levels, so the driver's documentation should be consulted. Isolation levels are well documented online, but the following link is a great starting point: [Data Concurrency and Consistency](#).

Isolation Level	Int Value	Constant
Read Uncommitted	1	system.db.READ_UNCOMMITTED
Read Committed	2	system.db.READ_COMMITTED
Repeatable Read	4	system.db.REPEATABLE_READ
Serializable	8	system.db.SERIALIZABLE

Code Examples

Code Snippet - Running a Query Using Query Transactions

```
# This example starts a transaction with a 5 second timeout against the project's default database, using
# the default isolation level. Then it executes a series of update calls, and commits and closes the
# transaction.

txId = system.db.beginTransaction(timeout=5000)
status=2

for machineId in range(8):
    system.db.runPrepUpdate("UPDATE MachineStatus SET status=? WHERE ID=?",
                           args=[status, machineId], tx=txId)

system.db.commitTransaction(txId)
system.db.closeTransaction(txId)
```

Keywords

system db beginTransaction, db.beginTransaction

system.db.clearAllNamedQueryCaches

This function is used in **Python Scripting**.

Description

This clears the caches of all Named Queries in a project. If called from the shared scope (i.e., Tag Event Scripts, Alarm Pipelines, etc.), then the name of the project must be passed as a parameter.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax - Project Scope

system.db.clearAllNamedQueryCaches()

- Parameters
 - Nothing
- Returns
 - Nothing
- Scope
 - Vision Clients

Syntax - Shared Scope

system.db.clearAllNamedQueryCaches(project)

- Parameters
 - String** project - The project that contains the Named Query whose cache needs to be cleared.
- Returns
 - Nothing
- Scope
 - Gateway, Perspective Session

Code Examples

Example - Clear All Named Query Cache for a Specific Project

```
# Calling this simply clears all Named Query caches.  
# This is assumed to run in the Shared Scope, so the name of the project must be included.  
system.db.clearAllNamedQueryCaches( "myProjectName" )
```

Example - Clear All Named Query Cache

```
# If multiple Named Queries with varying parameters are called in a single script, then  
clearAllNamedQueryCaches can be used to free up the memory used by all of the newly created caches.  
# This example is assumed to run in the Project Scope, so the project parameter may be omitted.  
  
# This creates one cache.
```

```
params = {"param1": "A"}  
system.db.runNamedQuery("myUpdateQuery", params)  
  
# This creates a separate cache.  
params = {"param1": "B"}  
system.db.runNamedQuery("anotherUpdateQuery", params)  
  
# Clear all of the caches from the current project. Note that all caches are cleared, including those  
generated from elsewhere on the Gateway.  
system.db.clearAllNamedQueryCaches()
```

Keywords

system db clearAllNamedQueryCaches, db.clearAllNamedQueryCaches

system.db.clearNamedQueryCache

This function is used in [Python Scripting](#).

Description

This clears the cache of a Named Query. If called from the shared scope (i.e., Tag Event Scripts, Alarm Pipelines, etc.) then the name of the project must be passed as a parameter.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax - Project Scope

system.db.clearNamedQueryCache(path)

- Parameters

String path - The path to the Named Query for which you want to clear the cache.

- Returns

Nothing

- Scope

Vision Client, Perspective Session

Syntax - Shared Scope

system.db.clearNamedQueryCache(project, path)

- Parameters

String project - The project that contains the Named Query whose cache needs to be cleared.

String path - The path to the Named Query for which you want to clear the cache.

- Returns

Nothing

- Scope

Gateway

Code Examples

Example - Clear Named Query Cache for a Specific Project

```
# Calling this simply clears all Named Query caches.  
# This example is being called from the shared scope. If called from the project scope, the projectName  
parameter should be omitted.
```

```
projectName = "myProject"  
namedQueryPath = "folder/selectFromInventory"  
  
system.db.clearNamedQueryCache(projectName, namedQueryPath)
```

Example - Clear Named Query Cache

```
# If the same Named Query is called multiple times with different parameters in a single script, then we
can clear the caches once we're done with the following.
# This example assumes the script is running in the project scope. If called from the Shared Scope, the
name of the project would need to be included.

namedQueryPath = "myUpdateQuery"

# This creates one cache.
params = {"param1":"A"}
system.db.runNamedQuery(namedQueryPath, params)

# This creates a separate cache.
params = {"param1":"B"}
system.db.runNamedQuery(namedQueryPath, params)

# Clear all of the caches from the specified Named Query. Note that all caches are cleared, including
those generated from elsewhere on the Gateway.
system.db.clearNamedQueryCache(namedQueryPath)
```

Keywords

system db clearNamedQueryCache, db.clearNamedQueryCache

system.db.closeTransaction

This function is used in [Python Scripting](#).

Description

Closes the transaction with the given ID. You must commit or rollback the transaction before you close it. Closing the transaction will return its database connection to the pool. The transaction ID will no longer be valid.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.db.closeTransaction(tx)

- Parameters

 String tx - The transaction ID.

- Returns

 Nothing

- Scope

 Gateway, Vision Client, Perspective Session

Code Examples

There are no examples available for this function.

Keywords

system db closeTransaction, db.closeTransaction

system.db.commitTransaction

This function is used in [Python Scripting](#).

Description

Performs a commit for the given transaction. This will make all statements executed against the transaction since its beginning or since the last commit or rollback take effect in the database. Until you commit a transaction, any changes that the transaction makes will not be visible to other connections.

Note: If you are done with the transaction, you must close it after you commit it

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.db.commitTransaction(tx)

- Parameters
 - String tx - The transaction ID.
- Returns
 - Nothing
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system db commitTransaction, db.commitTransaction

system.db.createSProcCall

This function is used in [Python Scripting](#).

Description

Creates an SProcCall object, which is a stored procedure call context. This is an object that is used to configure a call to a stored procedure. Once configured, you'd use [system.db.execSProcCall](#) to call the stored procedure. The call context object then holds any results from the stored procedure. The SProcCall object has the following functions used for registering parameters:

`SPRocCall.registerInParam(index OR name, typeCode, value)`

`SPRocCall.registerOutParam(index OR name, typeCode)`

`SPRocCall.registerReturnParam(typeCode)`

These functions are used to register any in/out parameters for the stored procedure. Parameters can be referenced by index (starting at 1, not 0), or by name. To register an in/out parameter, you simply register it twice - once as an input parameter with the value you'd like to pass to the stored procedure, and once as an output parameter. Note that not all JDBC drivers support named procedure parameters. If your function returns a value, you must use registerReturnParam to specify the data type of the returned value. Also be aware that this is different from stored procedures that return a result set, which doesn't require any setup on the SProcCall object. Some database systems call stored procedures that return a value of "functions" instead of "procedures". For all of these functions, you'll need to specify a type code. These are codes defined by the JDBC specification. For your convenience, the codes exist as constants in the system.db namespace. Each type code will be mapped to a database-specific type by the JDBC driver. Not all type codes will be recognized by all JDBC drivers. The following type code constants are available for use in createSProcCall:

BIT	REAL	LONGVARCHAR	LONGVARBINARY
TINYINT	DOUBLE	DATE	NULL
SMALLINT	NUMERIC	TIME	ROWID
INTEGER	DECIMAL	TIMESTAMP	CLOB
BIGINT	CHAR	BINARY	NCLOB
FLOAT	VARCHAR	VARBINARY	BLOB
NCHAR	NVARCHAR	LONGNVARCHAR	BOOLEAN

The following type code constants are available for other uses, but are not supported by createSProcCall:

ORACLE_CURSOR	DISTINCT	STRUCT	REF
JAVA_OBJECT	SQLXML	ARRAY	DATALINK
OTHER			

Once the call context has been executed, you can retrieve the result set, return value, and output parameter values (if applicable) by calling the following functions:

`SProcCall.getResultSet()` - returns a dataset that is the resulting data of the stored procedure, if any.

`SProcCall.getUpdateCount()` - returns the number of rows modified by the stored procedure, or -1 if not applicable.

`SProcCall.getReturnValue()` - returns the return value, if registerReturnParam had been called.

`SProcCall.getOutParamValue(index OR name)` - returns the value of the previously registered out-parameter.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

`system.db.createSProcCall(procedureName, [database], [tx], [skipAudit])`

- Parameters

String procedureName - The named of the stored procedure to call.

String database - The name of the database connection to execute against. If omitted or "", the project's default database connection will be used. [optional]

String tx - A transaction identifier. If omitted, the call will be executed in its own transaction. [optional]

Boolean skipAudit - A flag which, if set to true, will cause the procedure call to skip the audit system. Useful for some queries that have fields which won't fit into the audit log. [optional]

- Returns

SProcCall - A stored procedure call context, which can be configured and then used as the argument to system.db.execSProcCall.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Creating Stored Procedure Call

```
# This example calls a stored procedure named "start_batch" against the current project's default database connection that had no input or output parameters, and did not return any values or results:
```

```
call = system.db.createSProcCall("start_batch")
system.db.execSProcCall(call)
```

Code Snippet - Creating Stored Procedure Call

```
# This example would call a stored procedure "get_shift_workers" with no arguments, which returned a result set of employees for the current shift. It then pushes the resulting dataset into a Table component:
```

```
call = system.db.createSProcCall("get_shift_workers")
system.db.execSProcCall(call)

results = call.getResultSet()
table = event.source.parent.getComponent("Table")
table.data = results
```

Code Snippet - Creating Stored Procedure Call With Stored Procedure Parameters

```
# This example would call a stored procedure that took two arguments, the first an integer and the second a string. It also is configured to return an integer value.
```

```
call = system.db.createSProcCall("perform_calculation")
call.registerReturnParam(system.db.INTEGER)
call.registerInParam(1, system.db.INTEGER, 42)
call.registerInParam(2, system.db.VARCHAR, "DC-MODE")

system.db.execSProcCall(call)

# Print the result to the console
print call.getReturnValue()
```

Code Snippet - Creating Stored Procedure Call With Stored Procedure Parameters

```
# This example would do the same as the one above, except for a stored procedure that returned its value  
using an out-parameter. It also uses named argument names instead of indexed arguments.  
  
call = system.db.createSProcCall("perform_calculation")  
call.registerInParam("arg_one", system.db.INTEGER, 42)  
call.registerInParam("arg_two", system.db.VARCHAR, "DC-MODE")  
call.registerOutParam("output_arg", system.db.INTEGER)  
  
system.db.execSProcCall(call)  
  
# Print the result to the console  
print call.getOutParamValue("output_arg")
```

Keywords

system db createSProcCall, db.createSProcCall

system.db.dateFormat

This function is used in [Python Scripting](#).

Description

This function is used to format dates nicely as strings. It uses a format string to guide its formatting behavior. Learn more about date formatting in [Dates](#).

Note:

This function uses the Java class [java.text.SimpleDateFormat](#) internally, and will accept any valid format string for that class.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.db.dateFormat(date, formatPattern)

- Parameters

Date date - The Date object that you'd like to format.

String formatPattern - A format pattern string to apply.

- Returns

String - The date as a string formatted according to the format pattern.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This example displays a message box on a button press that displays the selected date (without the time)
# from a Calendar component, in a format like "Feb 3, 2009"
date = event.source.parent.getComponent("Calendar").latchedDate
toDisplay = system.db.dateFormat(date, "MMM d, yyyy")
system.gui.messageBox("The date you selected is: %s" % toDisplay)
```

Code Snippet

```
# This example does the same as the one above, but also displays the time, in a format like: "Feb 3, 2009
8:01pm"
date = event.source.parent.getComponent("Calendar").latchedDate
toDisplay = system.db.dateFormat(date, "MMM d, yyyy hh:mm a")
system.gui.messageBox("The date you selected is: %s" % toDisplay)
```

Code Snippet

```
# This example takes two dates from two Popup Calendar components, formats them in a manner that the
database understands,
```

```
# and then uses them in a SQL query to limit the results to a certain date range.  
startDate = event.source.parent.getComponent("StartDate").date  
endDate = event.source.parent.getComponent("EndDate").date  
startDate = system.db.dateFormat(startDate, "yyyy-MM-dd HH:mm:ss")  
endDate = system.db.dateFormat(endDate, "yyyy-MM-dd HH:mm:ss")  
query = ("SELECT * FROM mytable WHERE t_stamp >= '%s' AND t_stamp <= '%s'" % (startDate, endDate))  
results = system.db.runQuery(query)  
event.source.parent.getComponent("Table").data = results
```

Keywords

system db dateFormat, db.dateFormat

system.db.execSProcCall

This function is used in **Python Scripting**.

Description

Executes a stored procedure call. The one parameter to this function is an SProcCall - a stored procedure call context. See the description of [system.db.createSProcCall](#) for more information and examples.

This feature is new in Ignition version **8.1.2**
[Click here](#) to check out the other new features

Client Permission Restrictions

Restrictions Prior to 8.1.2

This scripting function has no [Client Permission](#) restrictions on Ignition versions earlier than 8.1.2.

Restrictions for 8.1.2 and beyond

[Permission Type](#) : Legacy Database Access

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.db.execSProcCall(callContext)

- Parameters

[SProcCall](#) callContext - A stored procedure call context, with any input, output, and/or return value parameters correctly configured.
Use [system.db.createSProcCall](#) to create a call context.

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system db execSProcCall, db.execSProcCall

system.db.getConnectionInfo

This function is used in [Python Scripting](#).

Description

Returns a dataset of information about a single database connection, as specified by the name argument.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.db.getConnectionInfo(name)

- Parameters

[String](#) name - The name of the database connection to find information about.

- Returns

[Dataset](#) - A dataset containing information about the named database connection, or an empty dataset if the connection wasn't found.

- Scope

Gateway, Vision Client, Perspective Session

Note: The database connection used when called from the Gateway scope is the connection configured on the Gateway scripting project.

Code Examples

Code Snippet - Getting Database Connection Information

```
# This example checks the database connection type and selects a query format that matches.

connectionInfo = system.db.getConnectionInfo()
dbType = connectionInfo.getValueAt(0, "DBType")
if dbType == "MYSQL":
    # mysql format for a column with a space in the name
    query = "SELECT `amps value` FROM pumps"
else:
    # mssql format for a column with a space in the name
    query = "SELECT [amps value] FROM pumps"
```

Keywords

system db getConnectionInfo, db.getConnectionInfo

system.db.getConnections

This function is used in [Python Scripting](#).

Description

Returns a dataset of information about each configured database connection. Each row represents a single connection.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.db.getConnections()

- Parameters
 - Nothing
- Returns
 - [Dataset](#) - A dataset, where each row represents a database connection.
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

There are no code examples for this function.

Keywords

system db getConnections, db.getConnections

system.db.refresh

This function is used in **Python Scripting**.

Description

This function will cause a Vision component binding to execute immediately. This is most often used for bindings that are set to Polling - Off. In this way, you cause a binding to execute on demand, when you know that the results of its query will return a new result. To use it, you simply specify the component and name of the property on whose binding you'd like to refresh.

Even though the function includes "db" in the name, the function can update all types of Vision component bindings, including Property and Expression bindings.

Note: This function will only work within the Vision module. To manually execute bindings in Perspective, use the [refreshBinding](#) component method.

Client Permission Restrictions

Permission Type: Legacy Database Access

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.db.refresh(component, propertyName)

- Parameters

[JComponent](#) component - The component whose property you want to refresh.

[String](#) propertyName - The name of the property that has a binding that needs to be refreshed.

- Returns

[Boolean](#) - True if the property was found and refreshed successfully.

- Scope

Vision Client

Code Examples

Code Snippet - Refreshing a Table's Data Property

```
# This example could be placed in the actionPerformed event of a Button, to be used to refresh the data
# of a Table.
# Remember to use the scripting name of the property that you're trying to refresh, and that the property
# names are case-sensitive.

table = event.source.parent.getComponent("Table")
system.db.refresh(table, "data")
```

Keywords

system db refresh, db.refresh

system.db.removeDatasource

This function is used in **Python Scripting**.

Description

Removes a database connection from Ignition.

Client Permission Restrictions

Permission Type: Datasource Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.db.removeDatasource(name)

- Parameters
 - String name - The name of the database connection in Ignition.
- Returns
 - Nothing
- Scope
 - Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Removing Database from Gateway

```
# This results in the connection named MySQL being removed.  
system.db.removeDatasource("MySQL")
```

Keywords

system db removeDatasource, db.removeDatasource

system.db.rollbackTransaction

This function is used in [Python Scripting](#).

Description

Performs a rollback on the given connection. This will make all statements executed against this transaction since its beginning or since the last commit or rollback undone.

If you are done with the transaction, you must also close it after you do a rollback on it.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.db.rollbackTransaction(tx)

- Parameters

String tx - The transaction ID.

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Python - Rollback a Transaction on an Exception

```
# This example uses a for-loop to run multiple queries in a single Transaction, and rollback if an error occurs.

# Create some variables for use later.
txId = system.db.beginTransaction(timeout=5000)
status=2
query = "UPDATE MachineStatus SET status=? WHERE ID=?"
errors = False          # A flag to denote if we ran into a problem with a query during the transaction.

for machineId in range(8):
    try:
        system.db.runPrepUpdate(query,           args=[status, machineId], tx=txId)
    except:
        errors = True
        break
# If we encountered an error...
if errors:
    # ...then rollback the transaction
    system.db.rollbackTransaction(txId)
else:
    # Otherwise, commit it.
    system.db.commitTransaction(txId)
# In either case, close the transaction when we're done.
system.db.closeTransaction(txId)
```

Keywords

system db rollbackTransaction, db.rollbackTransaction

system.db.runNamedQuery

This function is used in **Python Scripting**.

Description

Runs a named query and returns the results. Note that the number of parameters in the function is determined by scope. Both versions of the function are listed below.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Project Scope Syntax

system.db.runNamedQuery(path, parameters, [tx], [getKey])

- Parameters

String path - The path to the named query to run. Note that this is the full path to the query, including any folders.

Dictionary[String, Any] parameters - A Python dictionary of parameters for the named query to use.

String tx - An transaction ID, obtained from [beginNamedQueryTransaction](#). If blank, will not be part of a transaction. [optional]

Boolean getKey - Only used for Update Query types. A flag indicating whether or not the result should be the number of rows affected (getKey=0) or the newly generated key value that was created as a result of the update (getKey=1). Not all databases support automatic retrieval of generated keys. [optional]

- Returns

Any - The results of the query. The exact object returned depends on the Query Type property of the Named Query: typically either a dataset when set to **Query**, an integer representing the number of rows affected when set to **Update Query**, or an object matching the data type of the value returned by a **Scalar Query**.

- Scope

Vision Client, Perspective Session

Gateway Scope Syntax

system.db.runNamedQuery(project, path, parameters, [tx], [getKey])

- Parameters

String project - The project name the query exists in.

String path - The path to the named query to run. Note that this is the full path to the query, including any folders.

Dictionary[String, Any] parameters - A Python dictionary of parameters for the named query to use.

String tx - An optional transaction ID, obtained from [beginNamedQueryTransaction](#). If blank, will not be part of a transaction. [optional]

Boolean getKey - Only used for Update Query types. A flag indicating whether or not the result should be the number of rows affected (getKey=0) or the newly generated key value that was created as a result of the update (getKey=1). Not all databases support automatic retrieval of generated keys. [optional]

- Returns

Any - The results of the query. The exact object returned depends on the Query Type property of the Named Query: typically either a dataset when set to **Query**, an integer representing the number of rows affected when set to **Update Query**, or an object matching the data type of the value returned by a **Scalar Query**.

- Scope

Gateway

Code Examples

Simple Example - Without Parameters

```
# This example runs a Named Query without any parameters in the Project scope.  
# The second argument in the function is NOT optional, so Named Queries that do not require a parameter  
must still pass an empty dictionary as an argument.  
  
# Request the Named Query with an empty dictionary as the second parameter.  
system.db.runNamedQuery("folderName/myNamedQuery", {})
```

Gateway Scope Example

```
# This example runs a Named Query without any parameters in the Gateway scope.  
# The last argument in the function is NOT optional, so Named Queries that do not require a parameter  
must still pass an empty dictionary as an argument.  
  
# Request the Named Query to execute.  
system.db.runNamedQuery("ProjectName", "folderName/myNamedQuery", {})
```

Simple Example - With Parameters

```
# This example runs a Named Query while passing some parameters in the Project scope.  
# The Named Query is assumed to have two parameters already defined on the Named Query:  
# param1 : A string  
# param2 : An integer  
  
# Create a Python dictionary of parameters to pass.  
params = {"param1": "my string", "param2": 10}  
  
# Run the Named Query.  
system.db.runNamedQuery("myUpdateQuery", params)
```

Keywords

system db runNamedQuery, db.runNamedQuery

system.db.runPrepQuery

This function is used in **Python Scripting**.

Description

Runs a prepared statement against the database, returning the results in a PyDataSet. Prepared statements differ from regular queries in that they can use a special placeholder, the question-mark character (?), in the query where any dynamic arguments would go, and then use an array of values to provide real information for those arguments. Make sure that the length of your argument array matches the number of question-mark placeholders in your query.

This call should be used for SELECT queries. This is a useful alternative to [system.db.runQuery](#) because it allows values in the WHERE clause, JOIN clause, and other clauses to be specified without having to turn those values into strings. This is safer because it protects against a problem known as a [SQL injection attack](#), where a user can input data that affects the query's semantics.

Note: The "?" placeholder refers to variables of the query statement that help the statement return the correct information. The "?" placeholder cannot reference column names, table names, or the underlying syntax of the query. This is because the SQL standard for handling the "?" placeholder excludes these items.

Client Permission Restrictions

Permission Type: Legacy Database Access

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.db.runPrepQuery(query, args, [database], [tx])

- Parameters

String query - A query (typically a SELECT) to run as a prepared statement with placeholders (?) denoting where the arguments go.

Object[] args - A list of arguments. Will be used in order to match each placeholder (?) found in the query.

String database - The name of the database connection to execute against. If omitted or "", the project's default database connection will be used. [optional]

String tx - A transaction identifier. If omitted, the query will be executed in its own transaction. [optional]

- Returns

PyDataset - The results of the query as a PyDataset.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Running Prepared Query With Query Parameter

```
# This example searches for all records in a LogEntry table where the message contained a user-entered search term.

search = event.source.parent.getComponent("SearchFor").text
# Wrap the term in % signs for LIKE-style matching
search = '%' + search + '%'
```

```
results= system.db.runPrepQuery("SELECT * FROM LogEntry WHERE EntryText LIKE ?", [search])
event.source.parent.getComponent("Table").data = results
```

Keywords

system db runPrepQuery, db.runPrepQuery

system.db.runPrepUpdate

This function is used in **Python Scripting**.

Description

Runs a prepared statement against the database, returning the number of rows that were affected. Prepared statements differ from regular queries in that they can use a special placeholder, the question-mark character (?), in the query where any dynamic arguments would go, and then use an array of values to provide real information for those arguments. Make sure that the length of your argument array matches the number of question-mark placeholders in your query. This call should be used for UPDATE, INSERT, and DELETE queries.

This is extremely useful for two purposes:

- This method avoids the problematic technique of concatenating user input inside of a query, which can lead to syntax errors, or worse, a nasty security problem called a [SQL injection attack](#). For example, if you have a user-supplied string that is used in a WHERE clause, you use single-quotes to enclose the string to make the query valid. What happens in the user has a single-quote in their text? Your query will fail. Prepared statements are immune to this problem.
- This is the only way to write an INSERT or UPDATE query that has binary or BLOB data. Using BLOBS can be very hand for storing images or reports in the database, where all clients have access to them.

Note:

The "?" placeholder refers to variables of the query statement that help the statement return the correct information. The "?" placeholder cannot reference column names, table names, or the underlying syntax of the query. This is because the SQL standard for handling the "?" placeholder excludes these items.

Client Permission Restrictions

Permission Type: Legacy Database Access

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.db.runPrepUpdate(query, args, [database], [tx], [getKey], [skipAudit])

- Parameters

String query - A query (typically an UPDATE, INSERT, or DELETE) to run as a prepared statement with placeholders (?) denoting where the arguments go.

List[Any] args - A list of arguments. Will be used in order to match each placeholder (?) found in the query.

String database - The name of the database connection to execute against. If omitted or "", the project's default database connection will be used. [optional]

String tx - A transaction identifier. If omitted, the update will be executed in its own transaction. [optional]

Boolean getKey - A flag indicating whether or not the result should be the number of rows affected(getKey=0) or the newly generated key value that was created as a result of the update (getKey=1). Not all databases support automatic retrieval of generated keys. [optional]

Boolean skipAudit - A flag which, if set to true, will cause the prep update to skip the audit system. Useful for some queries that have fields which won't fit into the audit log. [optional]

- Returns

Integer - The number of rows affected by the query, or the key value that was generated, depending on the value of the getKey flag.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Inserting Data Into Database

```
# This example gathers some user entered text and inserts it into the database.

userText = event.source.parent.getComponent("TextArea").text
userName = system.security.getUsername()
system.db.runPrepUpdate("INSERT INTO Comments (Name, UserComment) VALUES (?,?)", [userName, userText])
```

Code Snippet - Inserting Data Into Database

```
# This example gathers some user entered text and inserts it into the database.
# Unlike the previous example, this example is explicitly declaring which database connection to run the
query against.
# Sometimes you need to run a query against a database connection that is not the default connection.

userText = event.source.parent.getComponent("TextArea").text
userName = system.security.getUsername()
databaseConnection = "AlternateDatabase"
system.db.runPrepUpdate("INSERT INTO Comments (Name, UserComment) VALUES (?,?)", [userName, userText],
databaseConnection)
```

Code Snippet - Reading File as Bytes and Inserting Bytes Into Database

```
# This code reads a file and uploads it to the database.

filename = system.file.openFile() # Ask the user to open a file
if filename != None:
    filedata = system.file.readFileAsBytes(filename)
    system.db.runPrepUpdate("INSERT INTO Files (file_data) VALUES (?)", [filedata])
```

Code Snippet - Inserting Data and Retrieving the Number of Affected Rows Using getKey Parameter

```
# This example inserts a new user and gives them the 'admin' role. Demonstrates the ability to retrieve a
newly created key value.

# Get the username/password
name = event.source.parent.getComponent('Name').text
desc = event.source.parent.getComponent('Description').text
building = event.source.parent.getComponent('Building').selectedValue

# Insert the value.
id = system.db.runPrepUpdate("INSERT INTO machines (machine_name, description) VALUES (?, ?)", [name,
desc], getKey=1)

# Add a row to the user role mapping table.
system.db.runPrepUpdate("INSERT INTO machine_building_mapping (machine_id, building) VALUES (?, ?)", [id,
building])
```

Code Snippet - Inserting Data From a Table Component

```
# This example takes a dataset from a table component and inserts new records into the database, one row
at a time

# Read the contents of the table
tableData = event.source.parent.getComponent('Table').data

# Convert it to a PyDataset. This is mostly for convenience, as they're easier to iterate through.
pyData = system.dataset.toPyDataSet(tableData)
```

```
# Build the query we'll use. You could easily modify the line to accommodate the table you're trying to
# insert into.
query = "INSERT INTO my_table (col1, col2) VALUES (?, ?)"

# Iterate.
for row in pyData:

    # Build an arguments list based on the current row. Using indexing here, so 'row[0]' is the 1st
    # column, 'row[1]' is the 2nd column, etc.
    args = [row[0], row[1]]

    # Add a row to the database. Optionally, you could check the contents of the row first and add an
    # if-statement to prevent the record based on some criteria.
    system.db.runPrepUpdate(query, args)
```

Keywords

system db runPrepUpdate, db.runPrepUpdate

system.db.runQuery

This function is used in **Python Scripting**.

Description

Runs a SQL query, usually a SELECT query, against a database, returning the results as a dataset. If no database is specified, or the database is the empty-string "", then the current project's default database connection will be used. The results are returned as a PyDataSet, which is a wrapper around the standard dataset that is convenient for scripting.

Client Permission Restrictions

Permission Type: Legacy Database Access

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.db.runQuery(query, [database], [tx])

- Parameters

String query - A SQL query, usually a SELECT query, to run.

String database - The name of the database connection to execute against. If omitted or "", the project's default database connection will be used. [optional]

String tx - A transaction identifier. If omitted, the query will be executed in its own transaction. [optional]

- Returns

PyDataset - The results of the query as a PyDataset.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Assuming the following dataset:

ID	Value
1	3.55
2	67.2
3	9.87

If you executed the following code:

Code Snippet

```
table = system.db.runQuery( "SELECT * FROM TEST" )
```

Table[2] would access the third row (rows are zero-indexed), and both table[2][0] and table[2]["ID"] would access the ID value of the third row.

As further example of how to use the results of runQuery, here are seven different ways to print out the table, and their results follow. Note that some of the later methods exercise some more advanced Jython concepts such as list comprehensions and string formatting, but their intent should be obvious. Generally speaking, the more concise Jython code becomes, the more readable it is.

Code Snippet - Executing Query and Printing Its Results

```
table = system.db.runQuery("SELECT * FROM Test")

print "Printing TEST Method 1..."
for row in table:
    for col in row:
        print col,
    print ""
print ""

print "Printing TEST Method 2..."
for row in table:
    print row[0], row[1]
print ""

print "Printing TEST Method 3..."
for row in table:
    print row["ID"], row["VALUE"]
print ""

print "Printing TEST Method 4..."
for rowIdx in range(len(table)):
    print "Row ",str(rowIdx)+": ", table[rowIdx][0], table[rowIdx][1]
print ""

print "Printing TEST Method 5..."
print [str(row[0])+", "+ str(row[1]) for row in table]
print ""

print "Printing TEST Method 6..."
print ["%s, %s" % (row["ID"],row["VALUE"]) for row in table]
print ""

print "Printing TEST Method 7..."
print [[col for col in row] for row in table]
print ""
```

The result would be:

Printing TEST Method 1...

0 3.55

1 67.2

2 9.87

Printing TEST Method 2...

0 3.55

1 67.2

2 9.87

Printing TEST Method 3...

0 3.55

1 67.2

2 9.87

Printing TEST Method 4...

Row 0: 0 3.55

Row 1: 1 67.2

Row 2: 2 9.87

Printing TEST Method 5...

[0, 3.55', '1, 67.2', '2, 9.87']

Printing TEST Method 6...

[0, 3.55', '1, 67.2', '2, 9.87']

Printing TEST Method 7...

[[0, 3.55], [1, 67.2], [2, 9.87]]

Keywords

system db runQuery, db.runQuery

system.db.runScalarPrepQuery

This function is used in **Python Scripting**.

Description

Runs a prepared statement against a database connection just like the runPrepQuery function, but only returns the value from the first row and column. If no results are returned from the query, the special value None is returned.

Client Permission Restrictions

Permission Type: Legacy Database Access

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.db.runScalarPrepQuery(query, args, [database], [tx])

- Parameters

String query - A SQL query (typically a SELECT) to run as a prepared statement with placeholders (?) denoting where the arguments go, that should be designed to return one row and one column.

List[Any] args - A list of arguments. Will be used in order to match each placeholder (?) found in the query.

String database - The name of the database connection to execute against. If omitted or "", the project's default database connection will be used. [optional]

String tx - A transaction identifier. If omitted, the query will be executed in its own transaction. [optional]

- Returns

Any - The value from the first row and first column of the results. Returns None if no rows were returned.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Executing Query

```
# This example searches for the user id of someone based on a typed in username.  
name = event.source.parent.getComponent("User Search").text  
  
result = system.db.runScalarPrepQuery("SELECT user_id FROM users WHERE username = ?" , [name])  
event.source.parent.getComponent("Text Field").data = result
```

Keywords

system db runScalarPrepQuery, db.runScalarPrepQuery

system.db.runScalarQuery

This function is used in **Python Scripting**.

Description

Runs a query against a database connection just like the runQuery function, but only returns the value from the first row and column. If no results are returned from the query, the special value None is returned.

Client Permission Restrictions

Permission Type: Legacy Database Access

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.db.runScalarQuery(query, [database], [tx])

- Parameters

String query - A SQL query that should be designed to return one row and one column.

String database - The name of the database connection to execute against. If omitted or "", the project's default database connection will be used. [optional]

String tx - A transaction identifier. If omitted, the query will be executed in its own transaction. [optional]

- Returns

Any - The value from the first row and first column of the results. Returns None if no rows were returned.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code counts the number of active alarms and acknowledges them all if there is at least one.  
numAlarms = system.db.runScalarQuery("SELECT COUNT(*) FROM alarmstatus " + "WHERE unacknowledged = 1")  
if numAlarms > 0:  
    # There are alarms - acknowledge all of them  
    system.db.runUpdateQuery( "UPDATE alarmstatus SET unacknowledged = 0" )
```

Code Snippet

```
# This code reads a single value from a table and shows it to the user an a popup.  
level = system.db.runScalarQuery("SELECT Level FROM LakeInfo WHERE LakeId='Tahoe'")  
system.gui.messageBox("The lake level is: %d feet" % level)
```

Keywords

system db runScalarQuery, db.runScalarQuery

system.db.runSFNamedQuery

This function is used in [Python Scripting](#).

Description

Runs a named query that goes through the [Store and Forward](#) system. Note that the number of parameters in the function is determined by scope. Both versions of the function are listed on this page.

Note: Only Update Named Queries are allowed in Store and Forward.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Project Scope Syntax

system.db.runSFNamedQuery(path, params)

- Parameters

[String](#) path - The path to the Named Query to run. Note that this is the full path to the query, including any folders.

[Dictionary\[String, Any\]](#) params - A Python dictionary of parameters for the Named Query to use.

- Returns

[Boolean](#) - Returns true if successfully sent to the Store and Forward system.

- Scope

Vision Client

Gateway Scope Syntax

system.db.runSFNamedQuery([project], path, params)

- Parameters

[String](#) project - The project name the query exists in. [optional]

[String](#) path - The path to the Named Query to run. Note that this is the full path to the query, including any folders.

[Dictionary\[String, Any\]](#) params - A Python dictionary of parameters for the Named Query to use.

- Returns

[Boolean](#) - Returns true if successfully sent to the Store and Forward system.

- Scope

Gateway, Perspective Session

Code Examples

Simple Example - Without Parameters

```
# This example runs a Named Query without any parameters in the Project scope.  
# The second argument in the function is NOT optional, so named queries that do not require a parameter  
# must still pass an empty dictionary as an argument.
```

```
# Request the Named Query with an empty dictionary as the second parameter.  
system.db.runSFNamedQuery("folderName/myNamedQuery", {})
```

Gateway Scope Example

```
# This example runs a Named Query without any parameters in the Gateway scope.  
# The last argument in the function is NOT optional, so named queries that do not require a parameter  
# must still pass an empty dictionary as an argument.  
  
# Request the Named Query to execute.  
system.db.runSFNamedQuery("ProjectName", "folderName/myNamedQuery", {})
```

Simple Example - With Parameters

```
# This example runs a Named Query while passing some parameters in the Project scope.  
# The Named Query is assumed to have two parameters already defined on the Named Query:  
# param1 : A string  
# param2 : An integer  
  
# Create a Python dictionary of parameters to pass  
params = {"param1": "my string", "param2": 10}  
  
# Run the Named Query  
system.db.runSFNamedQuery("myUpdateQuery", params)
```

Keywords

```
system db runSFNamedQuery, db.runSFNamedQuery
```

system.db.runSFPrepUpdate

This function is used in [Python Scripting](#).

Description

Runs a prepared statement query through the [Store and Forward](#) system and to multiple datasources at the same time. Prepared statements differ from regular queries in that they can use a special placeholder, the question-mark character (?) in the query where any dynamic arguments would go, and then use an array of values to provide real information for those arguments. Make sure that the length of your argument array matches the number of question-mark placeholders in your query. This call should be used for UPDATE, INSERT, and DELETE queries.

This is extremely useful for two purposes:

- This method avoids the problematic technique of concatenating user input inside of a query, which can lead to syntax errors, or worse, a nasty security problem called a SQL injection attack. For example, if you have a user-supplied string that is used in a WHERE clause, you use single-quotes to enclose the string to make the query valid. What happens if the user has a single-quote in their text? Your query will fail. Prepared statements are immune to this problem.
- This is the only way to write an INSERT or UPDATE query that has binary or BLOB data. Using BLOBS can be very handy for storing images or reports in the database, where all clients have access to them.

Client Permission Restrictions

[Permission Type](#): Legacy Database Access

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.db.runSFPrepUpdate(query, args, datasources)

- Parameters

[String](#) query - A query (typically an UPDATE, INSERT, or DELETE) to run as a prepared statement, with placeholders (?) denoting where the arguments go.

[List\[Any\]](#) args - A list of arguments. Will be used in order to match each placeholder (?) found in the query.

[List\[String\]](#) datasources - List of datasources to run the query through.

- Returns

[Boolean](#) - Returns true if successfully sent to Store and Forward system.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Example 1: Run through single datasource
print system.db.runSFPrepUpdate("INSERT INTO recipes (name, sp1, sp2, sp3) VALUES (?,?,?,?,?)", [ 'A Name',
1032, 234, 1], datasources=[ "MySQLDatasource" ])
```

Code Snippet

```
# Example 2: Run through two datasources
print system.db.runSFPrepUpdate("INSERT INTO recipes (name, sp1, sp2, sp3) VALUES (?,?,?,?,?)", [ 'A Name',
1032, 234, 1], datasources=[ "MySQLDatasource", "SQLServerDatasource" ])
```

Keywords

system db runSFPrepUpdate, db.runSFPrepUpdate

system.db.runSFUpdateQuery

This function is used in **Python Scripting**.

Description

Runs a query through the [Store and Forward](#) system and to multiple datasources at the same time.

Client Permission Restrictions

Permission Type: Legacy Database Access

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.db.runSFUpdateQuery(query, datasources)

- Parameters

String query - A query (typically an UPDATE, INSERT, or DELETE) to run.

List[String] datasources - List of datasources to run the query through.

- Returns

Boolean - Returns True if successful, False if not.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Example 1: Run through single datasource
print system.db.runSFUpdateQuery("INSERT INTO recipes (name, sp1, sp2, sp3) VALUES ('A Name', 1032, 234, 1)", ["MySQLDatasource"])
```

Code Snippet

```
# Example 2: Run through two datasources
print system.db.runSFUpdateQuery("INSERT INTO recipes (name, sp1, sp2, sp3) VALUES ('A Name', 1032, 234, 1)", ["MySQLDatasource", "SQLServerDatasource"])
```

Keywords

system db runSFUpdateQuery, db.runSFUpdateQuery

system.db.runUpdateQuery

This function is used in **Python Scripting**.

Description

Runs a query against a database connection, returning the number of rows affected. Typically this is an UPDATE, INSERT, or DELETE query. If no database is specified, or the database is the empty-string "", then the current project's default database connection will be used.

Note that you may want to use the runPrepUpdate query if your query is constructed with user input (to avoid the user's input from breaking your syntax) or if you need to insert binary or BLOB data.

Client Permission Restrictions

Permission Type: Legacy Database Access

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.db.runUpdateQuery(query, [database], [tx], [getKey], [skipAudit])

- Parameters

String query - A SQL query, usually an INSERT, UPDATE, or DELETE query, to run.

String database - The name of the database connection to execute against. If omitted or "", the project's default database connection will be used. [optional]

String tx - A transaction identifier. If omitted, the update will be executed in its own transaction. [optional]

Boolean getKey - A flag indicating whether or not the result should be the number of rows affected (getKey=0) or the newly generated key value that was created as a result of the update (getKey=1). Not all databases support automatic retrieval of generated keys. [optional]

Boolean skipAudit - A flag which, if set to true, will cause the update query to skip the audit system. Useful for some queries that have fields which won't fit into the audit log. [optional]

- Returns

Integer - The number of rows affected by the query, or the key value that was generated, depending on the value of the getKey flag.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# This code acknowledges all unacknowledged alarms # and shows the user how many alarms were acknowledged.  
rowsChanged = system.db.runUpdateQuery("UPDATE alarmstatus SET unacknowledged = 0")  
system.gui.messageBox("Acknowledged %d alarms" % rowsChanged)
```

Code Snippet

```
# This example inserts a new user and gives it the 'admin' role. Demonstrates the ability to retrieve a  
newly created key value.  
# get the username/password  
name = event.source.parent.getComponent('Name').text  
desc = event.source.parent.getComponent('Description').text  
building = event.source.parent.getComponent('Building').selectedValue
```

```
# insert the value
id = system.db.runUpdateQuery("INSERT INTO machines (machine_name, description) " + "VALUES ('%s', '%s')"
%(name, desc), getKey=1)

# add a row to the user role mapping table
system.db.runUpdateQuery("INSERT INTO machine_building_mapping " + "(machine_id, building) VALUES (%d, %d)"
%(id, building))
```

Keywords

system db runUpdateQuery, db.runUpdateQuery

system.db.setDatasourceConnectURL

This function is used in **Python Scripting**.

Description

Changes the connect URL for a given database connection.

Client Permission Restrictions

Permission Type: Datasource Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.db.setDatasourceConnectURL(name, connectUrl)

- Parameters

 String name - The name of the database connection in Ignition.

 String connectUrl - The new connect URL.

- Returns

 Nothing

- Scope

 Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Example:  
  
system.db.setDatasourceConnectURL("MySQL", "jdbc:mysql://localhost:3306/test")
```

Keywords

system db setDatasourceConnectURL, db.setDatasourceConnectURL

system.db.setDatasourceEnabled

This function is used in **Python Scripting**.

Description

Enables/disables a given database connection.

Client Permission Restrictions

Permission Type: Datasource Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.db.setDatasourceEnabled(name, enabled)

- Parameters

String name - The name of the database connection in Ignition.

Boolean enabled - Specifies whether the database connection will be set to enabled or disabled state.

- Returns

 Nothing

- Scope

 Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Enabling a Database Connection

```
# Enable the database named "MySQL"  
  
system.db.setDatasourceEnabled("MySQL", 1)
```

Code Snippet - Disabling a Database Connection

```
# Disable the database named "MySQL"  
  
system.db.setDatasourceEnabled("MySQL", 0)
```

Keywords

system db setDatasourceEnabled, db.setDatasourceEnabled

system.db.setDatasourceMaxConnections

This function is used in **Python Scripting**.

Description

Sets the Max Active and Max Idle parameters of a given database connection.

Client Permission Restrictions

Permission Type: Datasource Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.db.setDatasourceMaxConnections(name, maxConnections)

- Parameters

String name - The name of the database connection in Ignition.

Integer maxConnections - The new value for Max Active and Max Idle.

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Setting the Max Connections of a Data Source

```
# Set the max connection count for the "MySQL" database to 20.  
system.db.setDatasourceMaxConnections("MySQL", 20)
```

Keywords

system db setDatasourceMaxConnections, db.setDatasourceMaxConnections

system.device

Device Functions

The following functions give you access to view and edit device connections in the Gateway.

[In This Section ...](#)

system.device.addDevice

This function is used in **Python Scripting**.

Description

Adds a new device connection in Ignition. Accepts a dictionary of parameters to configure the connection. Acceptable parameters differ by device type: i.e., a Modbus/TCP connection requires a hostname and port, but a simulator doesn't require any parameters. When using this function, the arguments *must* be passed in as **keyword arguments**.

Client Permission Restrictions

Permission Type: Device Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.device.addDevice(deviceType, deviceName, deviceProps)

- Parameters

String `deviceType` - The device driver type. Possible values are listed in the Device Types table below.

String `deviceName` - The name that will be given to the the new device connection.

Dictionary[String, Any] `deviceProps` - A dictionary of device connection properties and values. Each deviceType has different properties, but most require at least a hostname. Keys in the dictionary are case-insensitive, spaces are omitted, and the names of the properties that appear when manually creating a device connection.

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Device Types

The tables below represent Inductive Automation device types that can be created with this function. Some device types require manual configurations to become fully functional, such as loading configuration files or adding mapped entries. In these cases you won't be able to completely configure the device with this function alone. Those device types are marked with "**(requires manual configuration)**" in the table below.

In addition, this function can also add devices from 3rd party modules; you will need to supply the driver type, which the module developer will be able to provide.

Driver Name	Device Type
Allen-Bradley Logix Driver	LogixDriver
Allen-Bradley MicroLogix	MicroLogix
Allen-Bradley PLC5	PLC5
Allen-Bradley SLC	SLC
DNP3 Driver	Dnp3Driver
Legacy Allen-Bradley CompactLogix	CompactLogix
Legacy Allen-Bradley	ControlLogix

ControlLogix	
Modbus RTU	ModbusRtuOverTcp
Modbus TCP	ModbusTcp
Omron FINS TCP (requires manual configuration)	com.inductiveautomation.FinsTcpDeviceType
Omron FINS UDP (requires manual configuration)	com.inductiveautomation.FinsUdpDeviceType
Omron NJ Driver (requires manual configuration)	com.inductiveautomation.omron.NjDriver

Driver Name	Device Type
Siemens S7-300	S7300
Siemens S7-400	S7400
Siemens S7-1200	S71200
Siemens S7-1500	S71500
Simulators Dairy Demo Simulator	DairyDemoSimulator
Simulators Generic Simulator	Simulator
Simulators SLC Simulator	SLCSimulator
TCP Driver	TCPDriver
UDP Driver	UDPDriver

Device Properties

The `deviceProps` parameter is where you supply configuration values to the new connection. Value properties depend on which `deviceType` was specified. A listing of `deviceProps` keys can be found on the [system.device.addDevice - deviceProps Listing](#) page.

The keys in the `deviceProps` parameter are **case-insensitive**. Device properties not specified in the `deviceProps` parameter will fallback to default values if not specified (where applicable: i.e., "hostname" typically does not have a default value).

Code Examples

Code Snippet - Creating a New Simulator Device

```
# Below is an example of creating a new Generic Simulator device connection.
# Note that we MUST pass a dictionary as the 3rd parameter, even if it's empty.

# Call the function
system.device.addDevice(deviceType = "Simulator", deviceName = "New_Generic_Simulator", deviceProps = {} )
```

Code Snippet - Creating a New Allen Bradley Logix Device

```
# Add a device using the Allen-Bradley Logix Driver for firmware v21+ devices
deviceProps = {}
deviceProps["Hostname"] = "192.168.1.2"
system.device.addDevice(deviceName="Test1", deviceType="LogixDriver", deviceProps=deviceProps)
```

Code Snippet - Creating a New Siemens Device

```
# Below is an example of creating a new S7-1500 device connection.

# Build a Dictionary of parameters
newProps = {
    "HostName" : "10.0.0.1",
    "Port" : 102 # <---If adding additional parameters, make sure to add a comma.
}

# Call the function
system.device.addDevice(deviceType = "S71500", \
                        deviceName = "My_S7_1500_Device", \
                        deviceProps = newProps )
```

Keywords

system device addDevice, device.addDevice

system.device.addDevice - deviceProps Listing

Description

Below is a table of properties callable by system.device.addDevice.

Note that the Description and Enabled properties may not be configured with this function, although a device connection could be disabled with a call to [system.device.setDeviceEnabled\(\)](#) after creating the connection.

LogixDriver Keys

Device Property	Key
Hostname	hostname
Port	port
Timeout	timeout
Max Concurrent Requests	concurrency
Slot Number	slotnumber
Connection Path	path
Automatic Rebrowse	automaticrebrowseenabled
CIP Connection Size	cipconnectionsize

On this page ...

- [LogixDriver Keys](#)
- [CompactLogix Keys](#)
- [com.inductiveautomation.BacnetIpDeviceType](#)
- [com.inductiveautomation.omron.NjDriver Keys](#)
- [com.inductiveautomation.FinsTcpDeviceType](#)
- [com.inductiveautomation.FinsUdpDeviceType](#)
- [ControlLogix Keys](#)
- [Dnp3Driver Keys](#)
- [MicroLogix Keys](#)
- [ModbusRtuOverTcp and ModbusTcp Keys](#)
- [PLC5 Keys](#)
- [S7300, S7400, S71200, and S71500 Keys](#)
- [SLC Keys](#)
- [TCPDriver Keys](#)
- [UDPDriver Keys](#)

CompactLogix Keys

Device Property	Key
Hostname	hostname
Timeout	timeout
Connection Path	path
Concurrent Requests	concurrentRequests
Disable Automatic Browse	disableAutomaticBrowse
Show String Arrays	showStringArrays
Status Request Poll Rate	pollRate

com.inductiveautomation.omron.NjDriver Keys

Device Property	Key
Hostname	hostname
Timeout	timeout
Concurrency	concurrency
Connection Size	connectionSize
Slot Number	slotNumber

com.inductiveautomation.FinsTcpDeviceType

Device Property	Key

Hostname	hostname
Port	port
Timeout	timeout
Local Address	localAddress
Source Network	sourceNetwork
Source Node	sourceNode
Source Unit	sourceUnit
Destination Network	destinationNetwork
Destination Nodes	destinationNode
Destination Unit	destinationUnit
Concurrent Requests	concurrentRequest
Max Request Size	maxRequestSize
Max Gap Size	maxGapSize
Write Priority Ratio	writePriorityRatio

com.inductiveautomation.FinsUdpDeviceType

Device Property	Key
Bind Address	bindAddress
Bind Port	bindPort
Remote Address	remoteAddress
Remote Port	remotePort
Timeout	timeout
Source Network	sourceNetwork
Source Node	sourceNode
Source Unit	sourceUnit
Destination Network	destinationNetwork
Destination Nodes	destinationNode
Destination Unit	destinationUnit
Concurrent Requests	concurrentRequest
Max Request Size	maxRequestSize
Max Gap Size	maxGapSize
Write Priority Ratio	writePriorityRatio

ControlLogix Keys

Device Property	Key
Hostname	hostname
Timeout	timeout
Connection Path	path
Concurrent Requests	concurrentRequests
Disable Automatic Browse	disableAutomaticBrowse

Show String Arrays	showStringArrays
Status Request Poll Rate	pollRate
Slot Number	slotNumber

Dnp3Driver Keys

Device Property	Key	Acceptable Values
Hostname	hostname	
Port	port	
Source Address	sourceAddress	
Destination Address	destinationAddress	
Integrity Poll Interval	integrityPollInterval	
Direct Operate Enabled	directOperateEnabled	
Unsolicited Messages Enabled	unsolicitedMessagesEnabled	
Message Fragment Size	maxMessageFragmentSize	
Message Timeout	timeout	
Retries	retries	
Link Layer Confirmation	linkLayerConfirmationEnabled	
Default Outstation Conformance Level	outstationConformanceDefault	<ul style="list-style-type: none"> • "UNKNOWN" • "ONE" • "TWO" • "THREE" • "FOUR"
Analog Input Points	analogInputDefaultValueType	<ul style="list-style-type: none"> • "INTEGER" • "SHORT" • "FLOAT" • "DOUBLE" • "VARIATION_0"
Analog Input Frozen Points	analogInputFrozenDefaultValueType	<ul style="list-style-type: none"> • "INTEGER" • "SHORT" • "FLOAT" • "DOUBLE" • "VARIATION_0"
Analog Output Points	analogOutputDefaultValueType	<ul style="list-style-type: none"> • "INTEGER" • "SHORT" • "FLOAT" • "DOUBLE" • "VARIATION_0"

Counter Points	counterDefaultValueType	<ul style="list-style-type: none"> • "INTEGER" • "SHORT" • "VARIATION_0"
Counter Frozen Points	counterFrozenDefaultValueType	<ul style="list-style-type: none"> • "INTEGER" • "SHORT" • "VARIATION_0"
Binary Input Points	binaryInputDefaultValueType	<ul style="list-style-type: none"> • "PACKED" • "WITH_FLAGS" • "VARIATION_0"
Double-Bit Binary Input Points	doubleBitBinaryInputDefaultValueType	<ul style="list-style-type: none"> • "PACKED" • "WITH_FLAGS" • "VARIATION_0"
Binary Output Points	binaryOutputDefaultValueType	<ul style="list-style-type: none"> • "PACKED" • "WITH_FLAGS" • "VARIATION_0"

MicroLogix Keys

Device Property	Key
Hostname	hostname
Timeout	timeout
Browse Cache Timeout	browseCacheTimeout
Connection Path	path
Disable Processor Browse	disableProcessorBrowse
Zero TNS Connection	useZeroTnsConnections

ModbusRtuOverTcp and ModbusTcp Keys

Device Property	Key
Hostname	hostname
Port	port
Communication Timeout	communicationTimeout
Max Holding Registers Per Request	maxHoldingRegistersPerRequest
Max Input Registers Per Request	maxInputRegistersPerRequest
Max Coils Per Request	maxCoilsPerRequest
Max Discrete Inputs Per Request	maxDiscreteInputsPerRequest
Reverse Word Order	reverseWordOrder
One-based Addressing	zeroBasedAddressing

Span Gaps	spanGaps
Allow Write Multiple Registers Request	writeMultipleRegistersRequestAllowed
Force Multiple Register Writes	forceMultipleRegisterWritesEnabled
Allow Write Multiple Coils Request	writeMultipleCoilsRequestAllowed
Allow Read Multiple Registers Request	readMultipleRegistersRequestAllowed
Allow Read Multiple Coils	readMultipleCoilsAllowed
Allow Read Multiple Discrete Inputs	readMultipleDiscreteInputsAllowed
Reconnect After Consecutive Timeouts	reconnectAfterConsecutiveTimeouts
Max Retry Count	maxRetryCount
Reverse String Byte Order	reverseStringByteOrder
Right Justify String	rightJustifyStrings
Read Raw Strings	readRawStrings

PLC5 Keys

Device Property	Key
Hostname	hostname
Timeout	timeout
Browse Cache Timeout	browseCacheTimeout
Connection Path	path
Disable Processor Browse	disableProcessorBrowse
Zero TNS Connection	useZeroTnsConnections

S7300, S7400, S71200, and S71500 Keys

Device Property	Key
Hostname	hostname
Timeout	timeout
Port	port
PDU Size	pduSize
Rack Number	rackNumber
CPU Slot Number	cpuSlotNumber
Reconnect After Consecutive Timeouts	reconnectAfterConsecutiveTimeouts

SLC Keys

Device Property	Key
Hostname	hostname
Timeout	timeout
Browse Cache Timeout	browseCacheTimeout
Connection Path	path
Disable Processor Browse	disableProcessorBrowse

Zero TNS Connection	useZeroTnsConnections
---------------------	-----------------------

TCPDriver Keys

Device Property	Key	Acceptable Values
Port(s)	ports	
Address	address	
Inactivity Timeout	inactivityTimeout	
Message Delimiter Type	messageDelimiterType	<ul style="list-style-type: none"> • "PacketBased" • "CharacterBased" • "FixedSize"
Message Delimiter	messageDelimiter	
Field Count	fieldCount	
Field Delimiter	fieldDelimiter	
Writeback Enabled	writebackEnabled	
Writeback Message Delimiter	writebackDelimiter	

UDPDriver Keys

Device Property	Key	Acceptable Values
Port(s)	ports	
Address	address	
Message Delimiter Type	messageDelimiterType	<ul style="list-style-type: none"> • "PacketBased" • "CharacterBased" • "FixedSize"
Message Delimiter	messageDelimiter	
Field Count	fieldCount	
Field Delimiter	fieldDelimiter	
Message Buffer Size	messageBufferSize	
Multicast	multicast	

system.device.listDevices

This function is used in [Python Scripting](#).

Description

Returns a dataset of information about each configured device. Each row represents a single device.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.device.listDevices()

- Parameters

Nothing

- Returns

[Dataset](#) - A dataset, where each row represents a device. Contains 4 columns *Name*, *Enabled*, *State*, and *Driver*.

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Listing Devices Configured on Your Gateway

```
deviceDataset = system.device.listDevices()

# Assign the deviceDataset to a Power Table. This example assumes
# the Power Table is in the same container as the component that called this script
event.source.parent.getComponent('Power Table').data = deviceDataset
```

Keywords

system device listDevices, device.listDevices

system.device.refreshBrowse

This function is used in [Python Scripting](#).

Description

Forces Ignition to browse the controller. Only works for Allen-Bradley controllers.

Client Permission Restrictions

This scripting function has no [Client Permission](#) restrictions.

Syntax

system.device.refreshBrowse(deviceName)

- Parameters

String deviceName - The name of the device in Ignition.

- Returns

Nothing

- Scope

Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Example:  
system.device.refreshBrowse( "CLX" )
```

Keywords

system device refreshBrowse, device.refreshBrowse

system.device.removeDevice

This function is used in **Python Scripting**.

Description

Removes a given device from Ignition.

Client Permission Restrictions

Permission Type: Device Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.device.removeDevice(deviceName)

- Parameters

 String deviceName - The name of the device in Ignition.

- Returns

 Nothing

- Scope

 Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Removing a Device from the Gateway

```
# Example:  
system.device.removeDevice("CLX")
```

Keywords

system device removeDevice, device.removeDevice

system.device.setDeviceEnabled

This function is used in **Python Scripting**.

Description

Enables/disables a device in Ignition.

Client Permission Restrictions

Permission Type: Device Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.device.setDeviceEnabled(deviceName, enabled)

- Parameters

String deviceName - The name of the device in Ignition.

Boolean enabled - Specifies whether the device connection will be set to enabled or disabled state.

- Returns

Nothing

- Scope

 Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet

```
# Example 1: Enable a device.  
  
system.device.setDeviceEnabled("CLX", 1)
```

Code Snippet

```
# Example 2: Disable a device.  
  
system.device.setDeviceEnabled("CLX", 0)
```

Keywords

system device setDeviceEnabled, device.setDeviceEnabled

system.device.setDeviceHostname

This function is used in **Python Scripting**.

Description

Changes the hostname of a device. Used for all Ethernet based drivers.

Client Permission Restrictions

Permission Type: Device Management

Client access to this scripting function is blocked to users that do not meet the role/zone requirements for the above permission type. This function is unaffected when run in the Gateway scope.

Syntax

system.device.setDeviceHostname(deviceName, hostname)

- Parameters

 String deviceName - The name of the device in Ignition.

 String hostname - The new IP address or hostname.

- Returns

 Nothing

- Scope

 Gateway, Vision Client, Perspective Session

Code Examples

Code Snippet - Changing Device Hostname

```
# Example:  
  
system.device.setDeviceHostname("CLX", "10.10.1.20")
```

Keywords

system device setDeviceHostname, device.setDeviceHostname

system.dnp3

DNP3 Functions

The following functions give you access to interact with the DNP3 devices.

Constants

```
system.dnp3.NUL = 0
system.dnp3.PULSE_ON = 1
system.dnp3.PULSE_OFF = 2
system.dnp3.LATCH_ON = 3
system.dnp3.LATCH_OFF = 4
system.dnp3.CLOSE = 1
system.dnp3.TRIP = 2
```

Status Codes

Many of the dnp3 functions return a status code. Those codes and their meaning are listed below.

Code Number	Identifier Name	Description
0	SUCCESS	Request accepted, initiated, or queued.
1	TIMEOUT	Request no accepted because the operate message was received after the arm timer timed out. The arm timer was started when the select operation for the same point was received.
2	NO_SELECT	Request no accepted because no previous matching select request exists. (An operate message was sent to activate an output that was not previously armed with a matching select message).
3	FORMAT_ER ROR	Request not accepted because there were formatting errors in the control request (either select, operate, or direct operate).
4	NOT_SUPPO RTED	Request not accepted because a control operation is not supported for this point.
5	ALREADY_A CTIVE	Request not accepted, because the control queue is full or the point is already active.
6	HARDWARE_ ERROR	Request not accepted because of control hardware problems.
7	LOCAL	Request not accepted because Local/Remote switch is in Local position.
8	TOO_MANY_ OBJS	Request not accepted because too many objects appeared in the same request.
9	NOT_AUTHO RIZED	Request not accepted because of insufficient authorization.
10	AUTOMATIO N_INHIBIT	Request not accepted because it was prevented or inhibited by a local automation process.
11	PROCESSIN G_LIMITED	Request not accepted because the device cannot process any more activities than are presently in progress.
12	OUT_OF_RA NGE	Request not accepted because the value is outside the acceptable range permitted for this point.
13 to 125	RESERVED	Reserved for future use.
126	NON_PARTIC IPATING	Sent in request messages indicating that the outstation will not issue or perform the control operation.
127	UNDEFINED	Request not accepted because of some other undefined reason.

In This Section ...

