Ignition 8.3 User Manual — Vision Component Reference

Generated 2026-10-12 from https://www.docs.inductiveautomation.com/docs/8.3/

Vision Components

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

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

- Admin Palette
- Alarming Palette
- Buttons Palette
- Calendar Palette
- Charts Palette
- Containers Palette
- Display Palette
- Input Palette

- Misc Palette
- Reporting Palette
- Tables Palette
- The Window Object

Vision - Admin Palette

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

Roster Management

User Management

Schedule Management

SFC Monitor

A component to monitor SFC performance.

Vision - Roster Management



Component Palette Icon:



Description

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

> **NOTE**
>
> The Alarm Notification module is required in order to use the Roster Management component.

Properties

Name	Description	Property Type	Scripting	Ca
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Co
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Co
Name	The name of this component.	String	.name	Co
Quality	The data quality code	QualityCode	.quality	Da

Name	Description	Property Type	Scripting	Ca
	for any Tag bindings on this component.			
Styles	Contains the component's styles.	Dataset	.styles	Ap
Table Color	Changes the background color of the table rows. When a row is selected, its color will revert to highlighted.	Color	.tableBackground	Ap
Table Header Color	Changes the background color of the table headers.	Color	.tableHeaderBackground	Ap
Table Header Text Color	Changes the text color of the table headers.	Color	.tableHeaderForeground	Ap
Table Text Color	Changes the text color of the table rows. When a row	Color	.tableForeground	Ap

Name	Description	Property Type	Scripting	Ca
	is selected, its text will revert to black.			
User Source	The user source to manage users in. If blank, uses the project's default user source.	String	.addFromUserSource	Be
Visible	If disabled, the component will be hidden.	boolean	.visible	Co
Window Color	Changes the background color of the window.	Color	.windowBackground	Ap
Window Header Color	Changes the background color of the window header.	Color	.windowHeaderBackground	Ap
Window Header Save Button Background	Changes the background color of the window	Color	.windowHeaderSaveButtonBackground	Ap

Name	Description	Property Type	Scripting	Ca
	header save button.			
Window Header Save Button Text Color	Changes the text color of the window header save button.	Color	.windowHeaderSaveButtonForeground	Ap
Window Header Text Color	Changes the window header text color.	Color	.windowHeaderForeground	Ap
Window Text Color	Changes the window header text color.	Color	.windowForeground	Ap

Scripting

See the **Vision - Roster Management Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

- **Vision Component Customizers**

Vision - Roster Management Scripting Functions

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

Component Functions

This component does not have component functions associated with it.

Extension Functions

filterRoster

- Description
  - Called for each roster loaded into the management table. Return false to hide this roster from the management table. This code is executed in a background thread.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - User Object **roster** - The name of the roster.
- Return
  - Boolean

filterAvailableUser

- Description
  - Called for each user in a user source to be shown as an available user for the roster currently being edited. Return false to hide this user so that it cannot be added to the roster. This code is executed in a background thread.
- Parameters

    - Component **self** - A reference to the component that is invoking this function.
    - String **roster** - The name of the roster being edited.
    - String **userSource** - The name of the user source being used to populate the list of available users.
    - User Object **user** - The user object itself. Call user.get('propertyName') to inspect. Common properties: 'username','schedule','language'. Call user.getRoles() for a list of rolenames.
- Return
    - Boolean

onSaveRoster

- Description
    - Called when the save button is pressed when editing a roster. This code is executed in a background thread.
- Parameters
    - Component **self** - A reference to the component that is invoking this function.
    - Object **saveContext** - An object that can be used to reject the edit by calling saveContext.rejectSave('reason')
    - String **rosterName** - The name of the roster being edited.
- Return
    - None

onCreateRoster

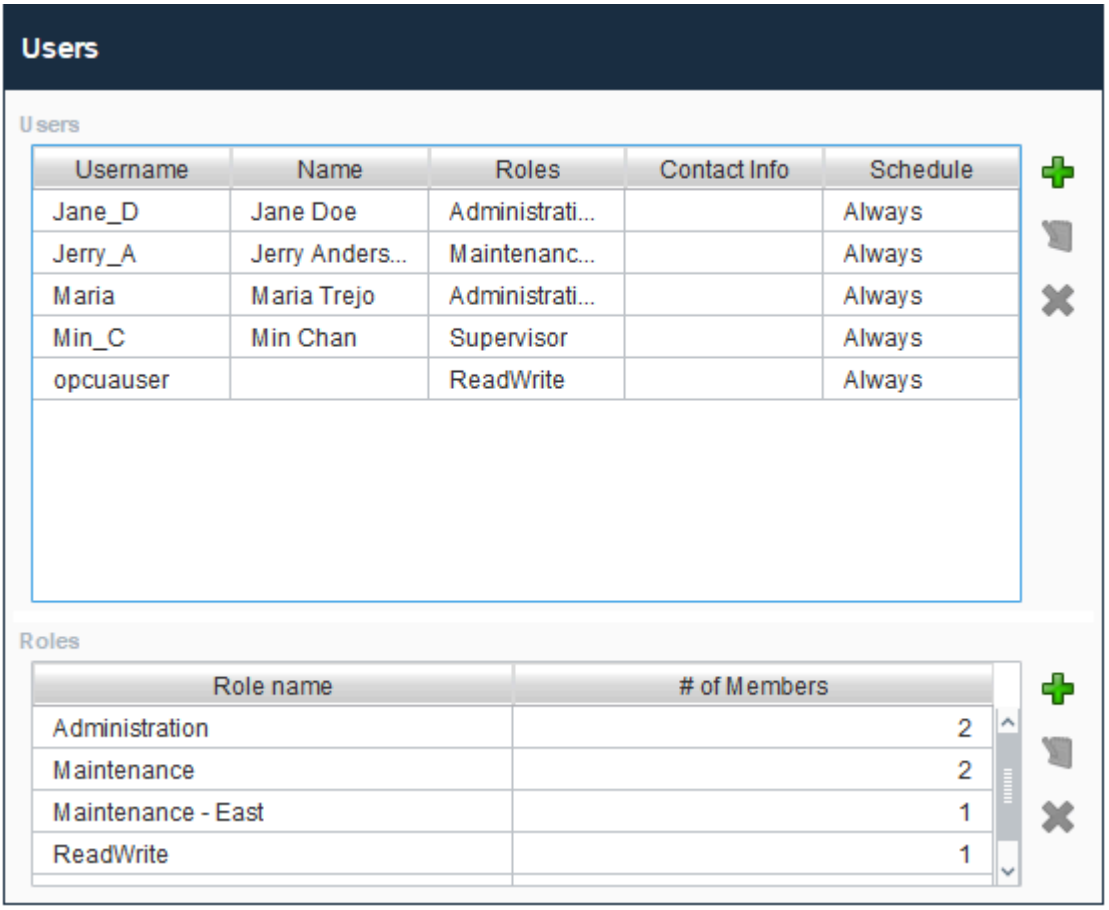
- Description
    - Called when the add button is pressed. This code is executed in a background thread.
- Parameters
    - Component **self** - A reference to the component that is invoking this function.
    - Object **createContext** - An object that can be used to reject the edit by calling createContext.rejectCreate('reason').
    - String **rosterName** - The name of the roster being created.
- Return

- None

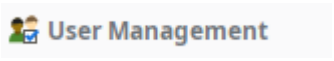
onDeleteRoster

- Description
  - Called when the delete button is pressed. This code is executed in a background thread.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - Object **deleteContext** - An object that can be used to reject the edit by calling deleteContext.rejectDelete('reason')
  - String **rosterNames** - A list of the roster names being deleted.
- Return
  - None

Vision - User Management



Component Palette Icon:



Description

The User Management component provides a built-in way to edit User Source users and roles from a Vision Client. To make changes to the Gateway's system user source from the Designer or Client, the **Allow User Admin setting** must be enabled. This allows for the administration of the Gateway's system user source from the Designer and Vision Client. Unless this is enabled, the Vision Module's User Management component is prevented from modifying the Gateway system's selected user source and you will see an error at the bottom of the component if it is attempted.

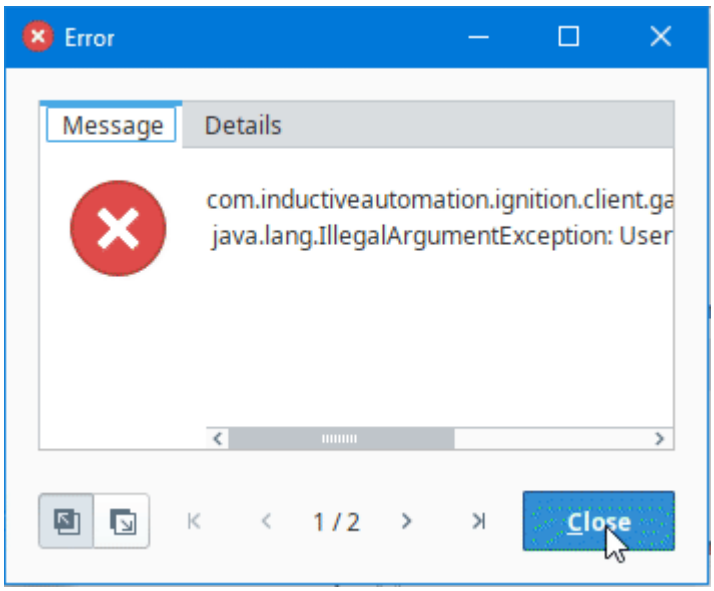
Schedule modifications are applied on top of the user's defined schedule to determine their effective schedule when evaluating for Schedule Restricted login.

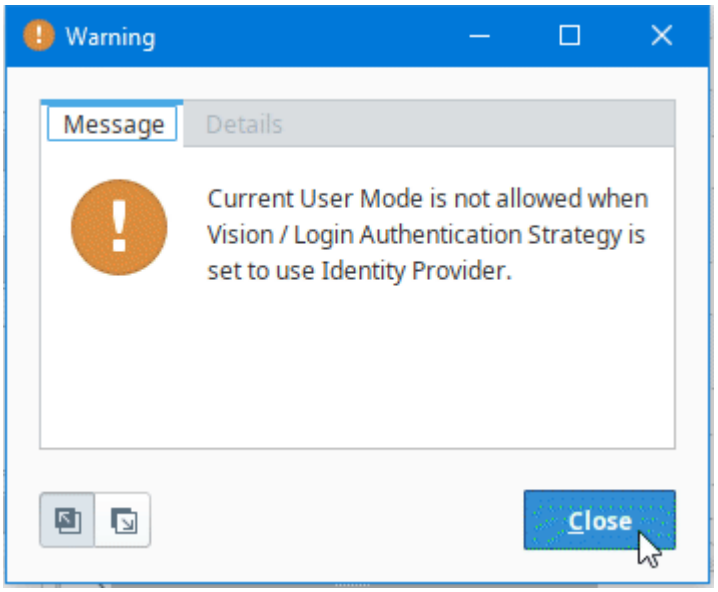
This component can be run in one of three modes:

- **Manage Users Mode**: In this mode, the component manages all of the users contained in the user source. Users and roles may be added, removed, and edited.
- **Edit Single Mode**: In this mode, the component only edits a single user. Which user is being edited is controlled via the "User Source" and "Username" properties.
- **Edit Current Mode**: In this mode, the user who is currently logged into the project can edit themselves. Obviously, the ability to assign roles is not available in this mode. This can be useful to allow users to alter their own password, adjust their contact information, and update their schedules.

> **CAUTION**
>
> Be careful to only expose this component to users who should have the privileges to alter other users. Access to this component in **Manage Users** mode will allow users to edit other users' passwords and roles.

The User Management component cannot be used in **Edit Current** mode when the Vision Client is using an Identity Provider to log in. Attempting to use the User Management component in this situation will result in the Vision client throwing an error and a warning:




Warning
Message
Details
Current User Mode is not allowed when Vision / Login Authentication Strategy is set to use Identity Provider.
Close

Properties

Name	Description	Property Type	Scripting
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border
Contact Info Editing Enabled	If true, a user's contact info will be editable.	boolean	.allowContactInfoEditing
Editing Schedule Available Color	Changes the color of the available times in the schedule. See **Color Selector**.	Color	.schedulePreviewAvailableColor
Editing Schedule Available Text Color	Changes the text color of events on the schedule preview. See **Color Selector**.	Color	.eventForeground
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled

Name	Description	Property Type	Scripting
Font	Font of the text on this component.	Font	.font
Mode	Affects what mode the user management component runs in. **Manage Users**: Allows edits to all Users and Roles in a single source determined by the User Source property. Default. intValue = 0 **Edit Current**: - Allows edits to the currently logged in user details intValue = 1 **Edit Single**: Allows edits to a specific user determined by the User Source and Username properties.intValue = 2	int	.mode
Name	The name of this component.	String	.name
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality

Name	Description	Property Type	Scripting
Role Assigning Enabled	If true, a user's roles will be editable.	boolean	.allowRoleAssigning
Role Management Enabled	If true, role management is available.	boolean	.allowRoleManagement
Row Height	Alter the size of the rows in the component's tables.	int	.rowHeight
Schedule Adjustments Enabled	If true, a user's schedule adjustments will be editable.	boolean	.allowScheduleModifications
Show Contact Info Column	Controls whether the user table shows the contact info column or not.	boolean	.columnContactInfo
Show Name Column	Controls whether the user table shows the name column or not.	boolean	.columnName
Show Roles Column	Controls whether the user table shows the roles column or not.	boolean	.columnRoles
Show Schedule Column	Controls whether the user table shows the	boolean	.columnSchedule

Name	Description	Property Type	Scripting
	schedule column or not.		
Show Username Column	Controls whether the user table shows the username column or not.	boolean	.columnUsername
Styles	Contains the component's styles.	Dataset	.styles
Table Color	Changes the background color of the tables, User Roles and Role Member lists. Note: When a row is selected it will revert to highlighted.	Color	.tableBackground
Table Header Color	Changes the background color of the table headers. See **Color Selector**.	Color	.tableHeaderBackground
Table Header Text Color	Changes the text color of the table headers. See **Color Selector**.	Color	.tableHeaderTextColor
Table Text Color	Changes the text color of the tables. Note: When a row is selected, it will	Color	.tableForeground

Name	Description	Property Type	Scripting
	revert to black. See **Color Selector**.		
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode
User Source	The user source to manage users in. If blank, uses the project's default user source.	String	.userProfile
Username	The name of the user being edited. Read-only except when mode is Edit Single, in which case it defines the user to be edited.	String	.username
Username Editing Enabled	If true, usernames will be editable.	boolean	.allowUsernameEditing
Visible	If disabled, the component will be hidden.	boolean	.visible
Window Color	Changes the window background color. See **Color Selector**.	Color	.windowBackground

Name	Description	Property Type	Scripting
Window Header Color	Changes the window header background color. See **Color Selector**.	Color	.windowHeaderBackground
Window Header Save Button Background Color	Changes the window header save button background color. See **Color Selector**.	Color	.windowHeaderSaveButtonBackground
Window Header Save Button Text Color	Changes the window header save button text color. See **Color Selector**.	Color	.windowHeaderSaveButtonForeground
Window Header Text Color	Changes the window header text color. See **Color Selector**.	Color	.windowHeaderForeground
Window Text Color	Changes the text color of the window. See **Color Selector**.	Color	.windowForeground

Scripting

See the **Vision - User Management Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

- **Vision Component Customizers**

Vision - User Management Scripting Functions

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

Component Functions

This component does not have component functions associated with it.

Extension Functions

filterUser

- Description
  - Called for each user loaded into the management table. Return false to hide this user from the management table. This code is executed in a background thread.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - User Object **user** - The user object itself. Call user.get('propertyName') to inpsect. Common properties: 'username',' schedule', 'language', user.getRoles() for a list of rolenames.
- Return
  - Boolean

filterRole

- Description
  - Called for each role loaded into the management table. Return false to hide this role from the management table. This code is executed in a background thread.
- Parameters

    - Component **self** - A reference to the component that is invoking this function.
    - String **role** - The role name.
- Return
    - Boolean

filterSchedule

- Description
    - Called for each schedule loaded into the schedule dropdown in the edit user panel. Return false to hide this schedule from the dropdown. This code is executed in a background thread.
- Parameters
    - Component **self** - A reference to the component that is invoking this function.
    - String **schedule** - The schedule name.
- Return
    - Boolean

onCreateUser

- Description
    - Called when the add button is pressed in the users table
- Parameters
    - Component **self** - A reference to the component that is invoking this function.
    - Object **saveContext** - An object that can be used to reject the add by calling saveContext.rejectSave('reason')
- Return
    - None

onDeleteUser

- Description

- Called when the delete button is pressed in the users table. This code is executed in the background thread and is called once for each user selected.

- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - Object **saveContext** - An object that can be used to reject the edit by calling saveContext.rejectSave('reason'). If more than one user is rejected, reasons will be concatenated.
  - Object **user** - The user that is trying to be deleted. Call user.get('propertyName') to inspect. Common properties: 'username', 'schedule', 'language'. Call user.getRoles() for a list of rolenames.

- Return
  - None

onSaveUser

- Description
  - Called when the save button is pressed when adding or editing a user. This code is executed in a background thread.

- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - Object **saveContext** - An object that can be used to reject the edit by calling saveContext.rejectSave('reason').
  - User Object **user** - The user that is trying to be saved. Call user.get('propertyName') to inspect. Common properties: 'username', 'schedule','language'. Call user.getRoles() for a list of rolenames.

- Return
  - None

onCreateRole

- Description
  - Called when the add button is pressed in the roles table.

- Parameters
    - Component **self** - A reference to the component that is invoking this function.
    - Object **saveContext** - An object that can be used to reject the add by calling saveContext.rejectSave('reason')
- Return
    - None

onDeleteRole

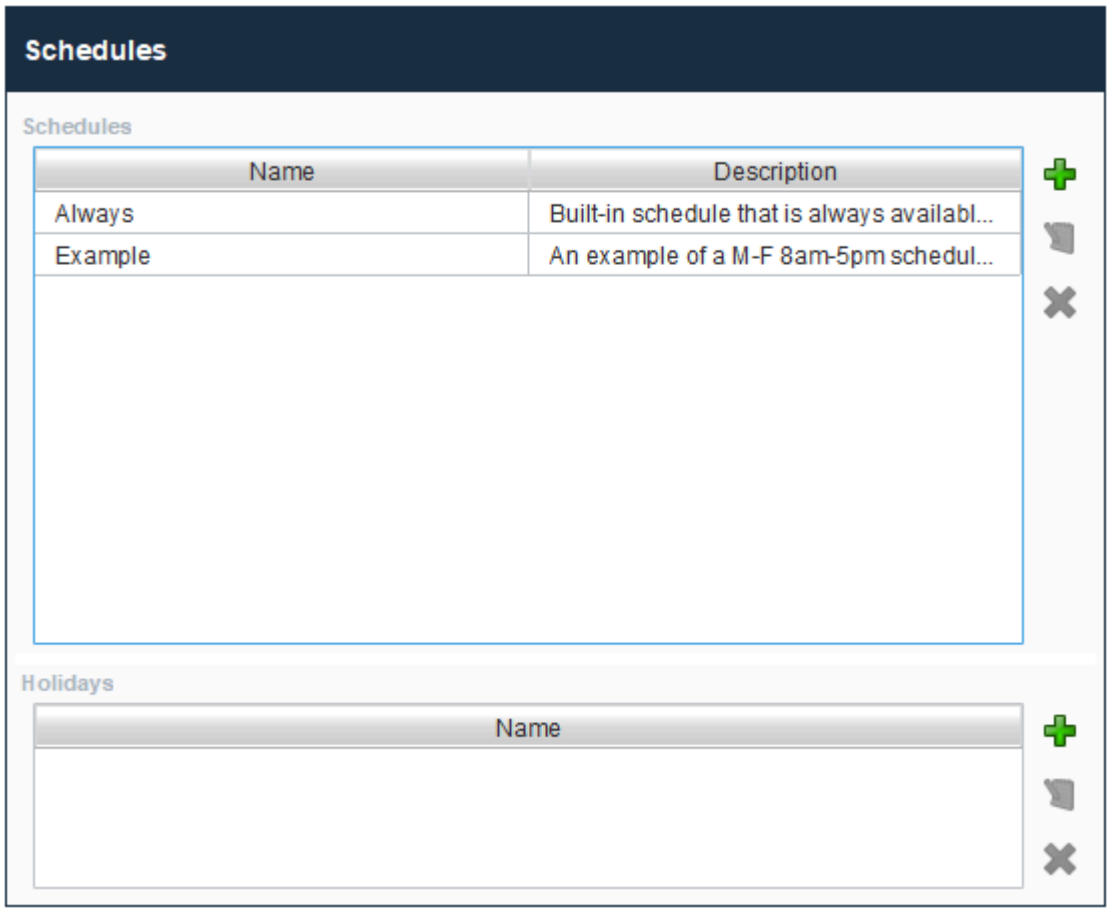
- Description
    - Called when the save button is pressed when adding or editing a role. This code is executed in a background thread.
- Parameters
    - Component **self** - A reference to the component that is invoking this function.
    - Object **saveContext** - An object that can be used to reject the edit by calling saveContext.rejectSave('reason'). If more than one role is rejected, reasons will be concatenated.
    - String **name** - The role name that is being deleted.
- Return
    - None

onSaveRole

- Description
    - Called when the save button is pressed when adding or editing a role. This code is executed in a background thread.
- Parameters
    - Component **self** - A reference to the component that is invoking this function.
    - Object **saveContext** - An object that can be used to reject the edit by calling saveContect.rejectSave('reason').
    - String **oldName** - The role name before editing. Will be None for a role being added.
    - String **newName** - The new name of the edited role.

- Return
    - None

Vision - Schedule Management



Component Palette Icon:

Schedule Management

Description

> **NOTE**
>
> Making changes to users from a client with this component requires that the User Management permissions has been enabled for the project.

This component allows for management of schedules. Schedules can be defined by specifying which days of the week and which times of day they are active on. The times of day are defined using a string of time ranges, where the times are specified in 24-hr format with dashes between the beginning and the end. Multiple ranges can be specified by separating them with commas. Examples:

8:00-17:00	Valid from 8am to 5pm
6:00-12:00, 12:45-14:00	Valid from 6am to noon, and then again from 12:45pm to 2pm
0:00-24:00	Always valid.

Schedules that alternate weekly or daily can be specified by using the repetition settings. All repeating schedules need a starting day. For example, you could have a schedule that repeats on a weekly basis, with 1-week on and 1-week off. This schedule would be active for seven days starting on the starting day, and then inactive for the next seven days, then active for seven days, and so on. Note that the days of the week and time settings are evaluated in addition to the repetition settings. This means that both settings must be true for the schedule to be active. Also note that if you set "Repeat / Alternate" to a setting other than "Off" and you do not specify a starting day, the schedule will never be active. See **Color Selector**.

Properties

Name	Description	Property Type	Scripting	Ca
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Co
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Co
Name	The name of this component.	String	.name	Co
Quality	The data quality code	QualityCode	.quality	Da

Name	Description	Property Type	Scripting	Ca
	for any Tag bindings on this component.			
Schedule Available Color	Changes the color of the available times in the schedule. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSLvalue. See **Color Selector**.	Color	.schedulePreviewAvailableColor	Ap
Schedule Available Text Color	Changes the text color of events on the schedule preview. See **Color Selector**.	Color	.eventForeground	Ap
Styles	Contains the component's styles.	Dataset	.styles	Ap
Table Color	Changes the	Color	.tableBackground	Ap

Name	Description	Property Type	Scripting	Ca
	background color of the tables, User Roles and Role Member lists. See **Color Selector**. Note: When a row is selected it will revert to highlighted.			
Table Header Color	Changes the background color of the table headers. See **Color Selector**.	Color	.tableHeaderBackground	Ap
Table Header Text Color	Changes the text color of the table headers. See **Color Selector**.	Color	.tableHeaderTextColor	Ap
Table Text Color	Changes the text color of the tables. Note: When a row is selected,	Color	.tableForeground	Ap

Name	Description	Property Type	Scripting	Ca
	it will revert to black. See **Color Selector**.			
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Be
Visible	If disabled, the component will be hidden.	boolean	.visible	Co
Window Color	Changes the window background color. See **Color Selector**.	Color	.windowBackground	Ap
Window Header Color	Changes the window header background color. See **Color Selector**.	Color	.windowHeaderBackground	Ap

Name	Description	Property Type	Scripting	Ca
Window Header Save Button Background Color	Changes the window header save button background color. See **Color Selector**.	Color	.windowHeaderSaveButtonBackground	Ap
Window Header Save Button Text Color	Changes the window header save button text color. See **Color Selector**.	Color	.windowHeaderSaveButtonForeground	Ap
Window Header Text Color	Changes the window header text color. See **Color Selector**.	Color	.windowHeaderForeground	Ap
Window Text Color	Changes the text color of the window. See **Color Selector**.	Color	.windowForeground	Ap

Scripting

See the **Vision - Schedule Management Scripting Functions** page for the full list of scripting functions available for this component.

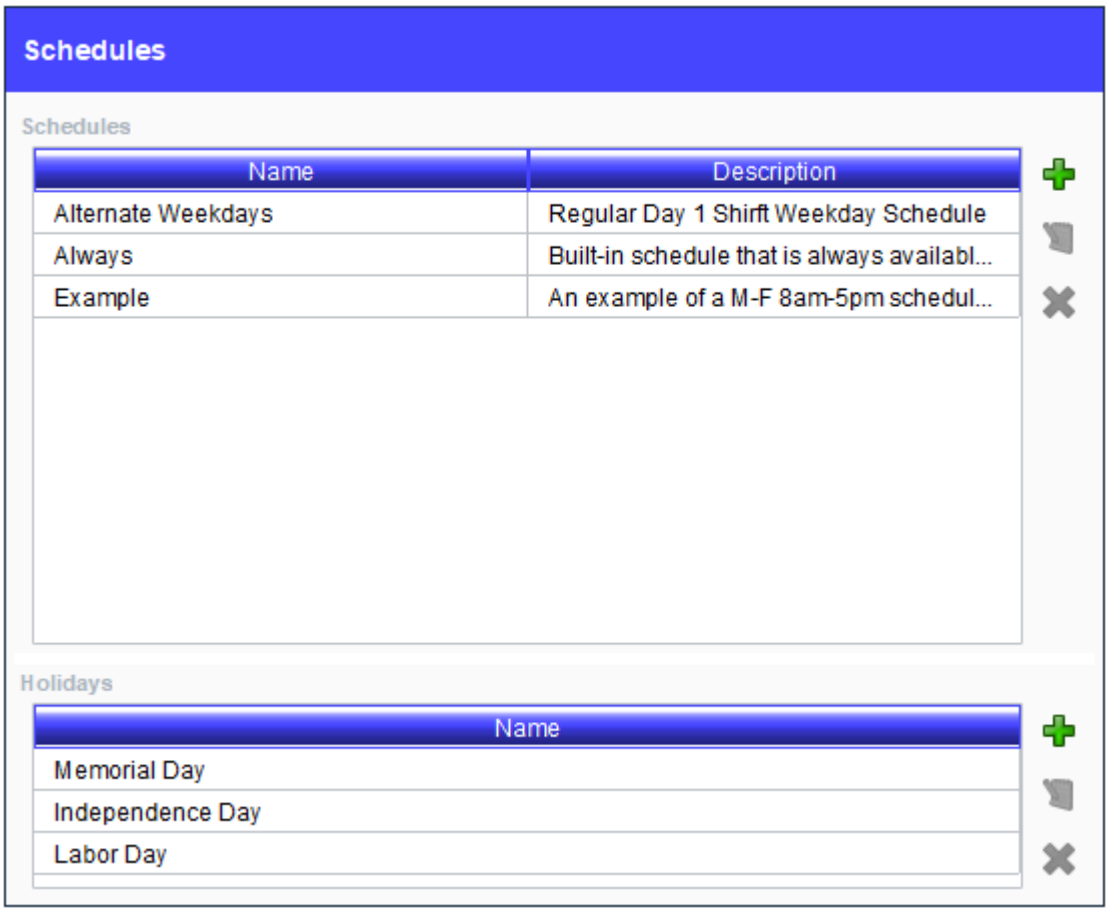
Event Handlers

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

Customizers

- **Vision Component Customizers**

Example



Property Name	Value
Name	Schedules
Enabled	True
Visible	True
Touchscreen Mode	Single-Click
Table Header Color	71,71,255
Table Header Text Color	255,255,255
Window Header Color	71,71,255

Vision - Schedule Management Scripting Functions

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

Component Functions

This component does not have component functions associated with it.

Extension Functions

filterSchedule

- Description
  - Called for each schedule loaded into the management table. Return false to hide this schedule from the management table. This code is executed in a background thread.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - String **schedule** - The schedule name
- Return
  - Boolean

filterHoliday

- Description
  - Called for each holiday loaded into the management table. Return false to hide this holiday from the management table. This code is executed in a background thread.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.

    - String **holiday** - The holiday name.
- Return
    - Boolean

onCreateSchedule

- Description
    - Called when the add button is pressed when adding a schedule. This code is executed in a background thread.
- Parameters
    - Component **self** - A reference to the component that is invoking this function.
    - Object **saveContext** - An object that can be used to reject the add by calling saveContect.rejectSave('reason').
- Return
    - None

onDeleteSchedule

- Description
    - Called when the delete button is pressed for one or more schedules. This code is executed in a background thread, once for each schedule to be deleted.
- Parameters
    - Component **self** - A reference to the component that is invoking this function.
    - Object **saveContext** - An object that can be used to reject the add by calling saveContext.rejectSave('reason')
    - String **name** - The name of the schedule to be deleted.
- Return
    - None

onSaveSchedule

- Description

    - Called when the save button is pressed when adding or editing a schedule. This code is executed in a background thread.

- Parameters

    - Component **self** - A reference to the component that is invoking this function.
    - Object **saveContext** - An object that can be used to reject the edit by calling saveContect.rejectSave('reason').
    - String **oldName** - The schedule name before editing. Will be None for a schedule being added.
    - String **newName** - The new name of the edited schedule.

- Return

    - None

onCreateHoliday

- Description

    - Called when the add button is pressed when to add a holiday. This code is executed in a background thread.

- Parameters

    - Component **self** - A reference to the component that is invoking this function.
    - Object **saveContext** - An object that can be used to reject the edit by calling saveContext.rejectSave('reason').

- Return

    - None

onDeleteHoliday

- Description

    - Called when the delete button is pressed for one or more holidays. This code is executed in a background thread, once for each holiday to be deleted.

- Parameters

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

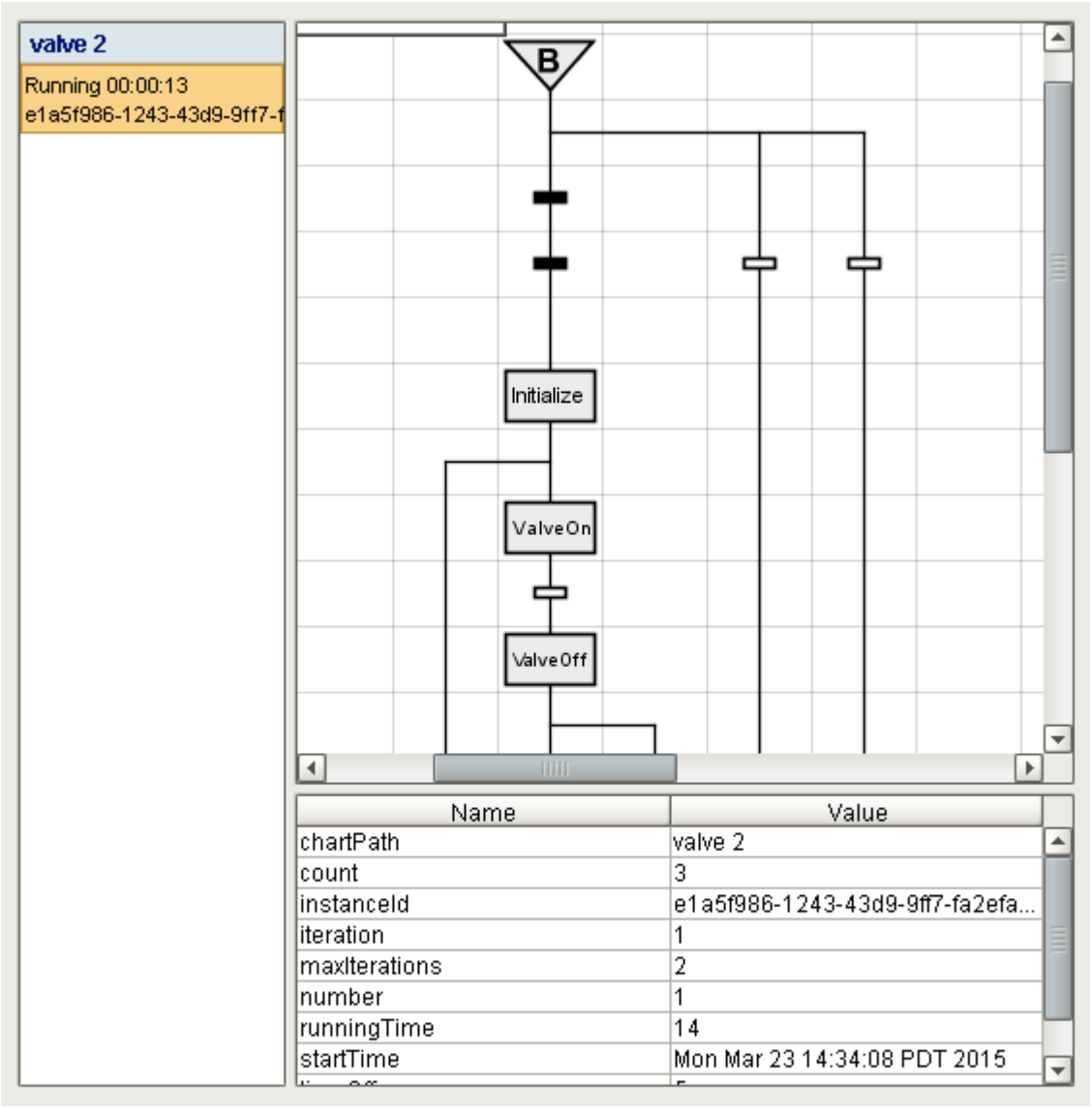
- Object **saveContext** - An object that can be used to reject the add by calling saveContext.rejectSave('reason')
- String **name** - The name of the holiday to be deleted.

- Return
  - None

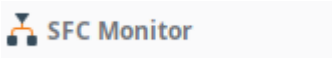
onSaveHoliday

- Description
  - Called when the save button is pressed when adding or editing a holiday. This code is executed in a background thread.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - Object **saveContext** - An object that can be used to reject the edit by calling saveContext.rejectSave('reason'). If more than one role is rejected, reasons will be concatenated.
  - String **oldName** - The holiday name before editing. Will be None for a holiday being added.
  - String **newName** - The new name of the edited holiday.
- Return
  - None

Vision - SFC Monitor



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), and Field 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

Name	Description	Property Type	Scripting	Category
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

Component Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

Customizers

- **Vision Component Customizers**

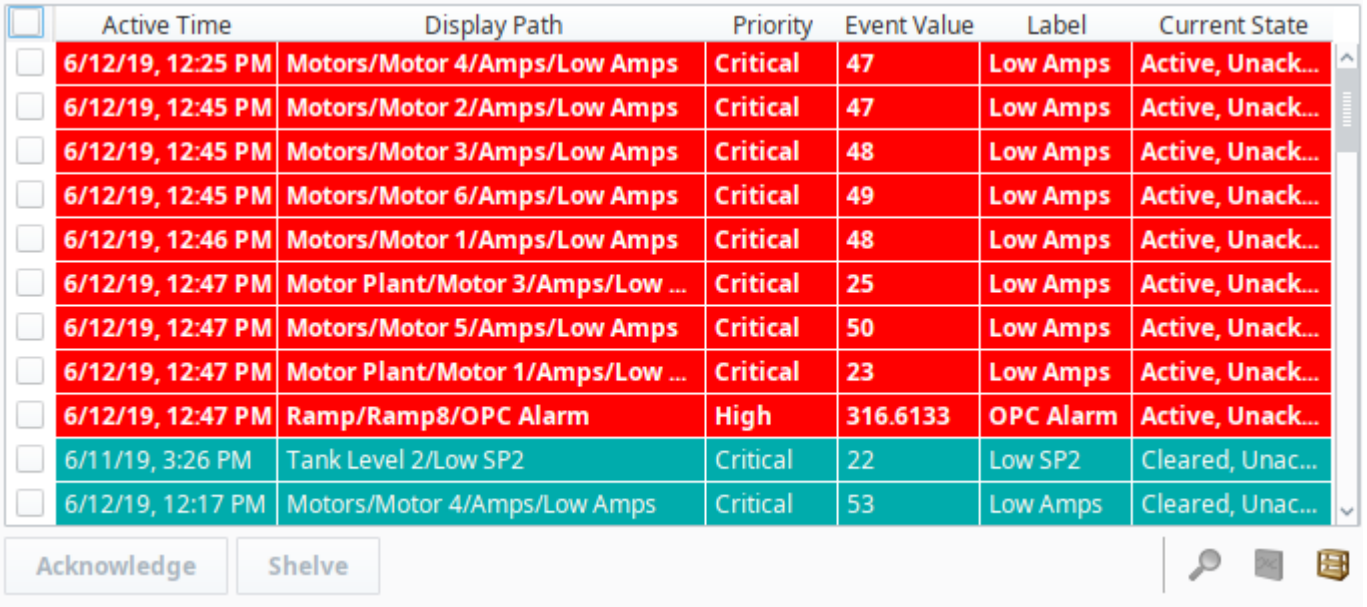
Vision - Alarming Palette

The following components give you options for displaying Alarm information.

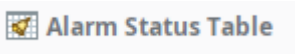
Alarm Status Table

Alarm Journal Table

Vision - Alarm Status Table



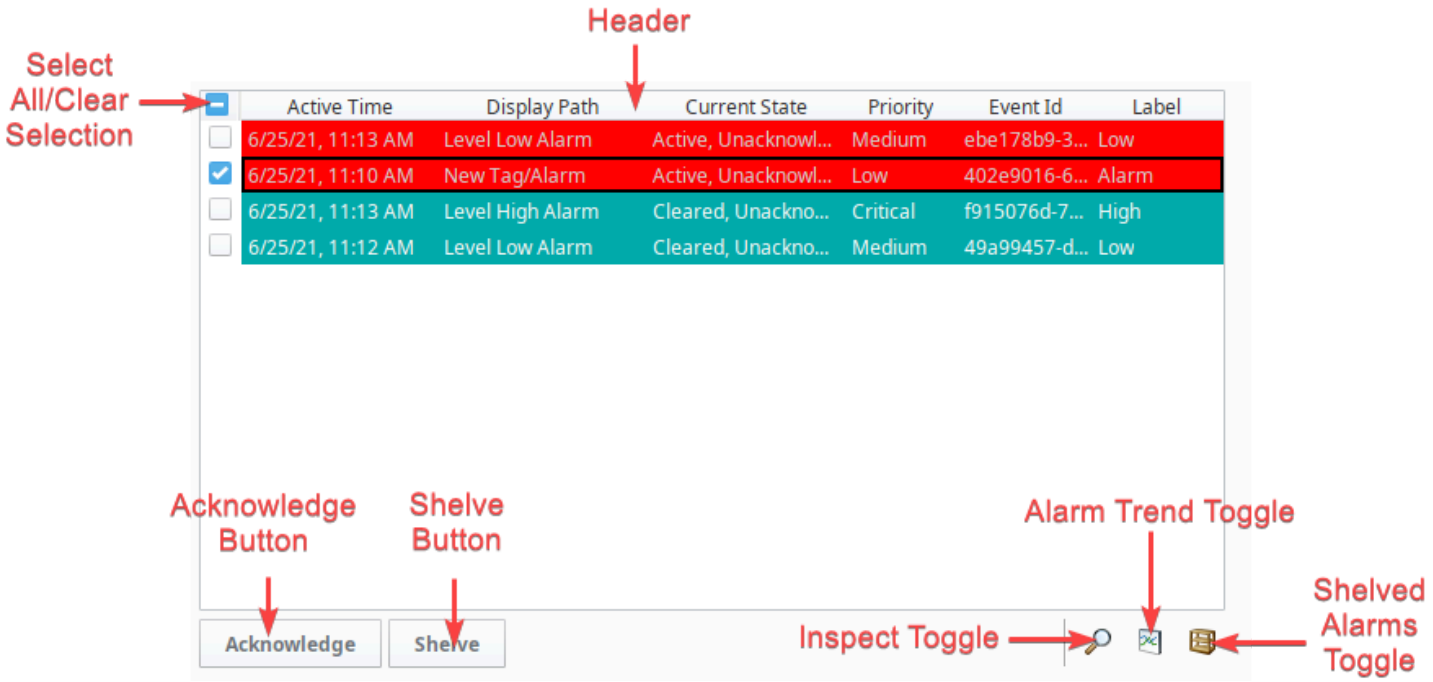
Component Palette Icon



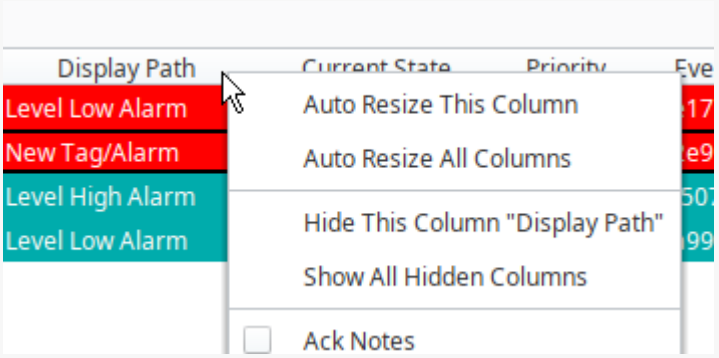
The alarm status table displays the current state of the alarms available to the gateway (including those provided by Remote Tag Providers). It can be configured to show active, unacknowledged, cleared, and acknowledged alarms. By default it shows all non-cleared/non-ack'ed alarms. Acknowledgement is handled by selecting (checking) alarms and pressing the "Acknowledge" button. If any of the selected alarms require acknowledge notes, then a small text area will be presented in which the operator must add notes to the acknowledgement.

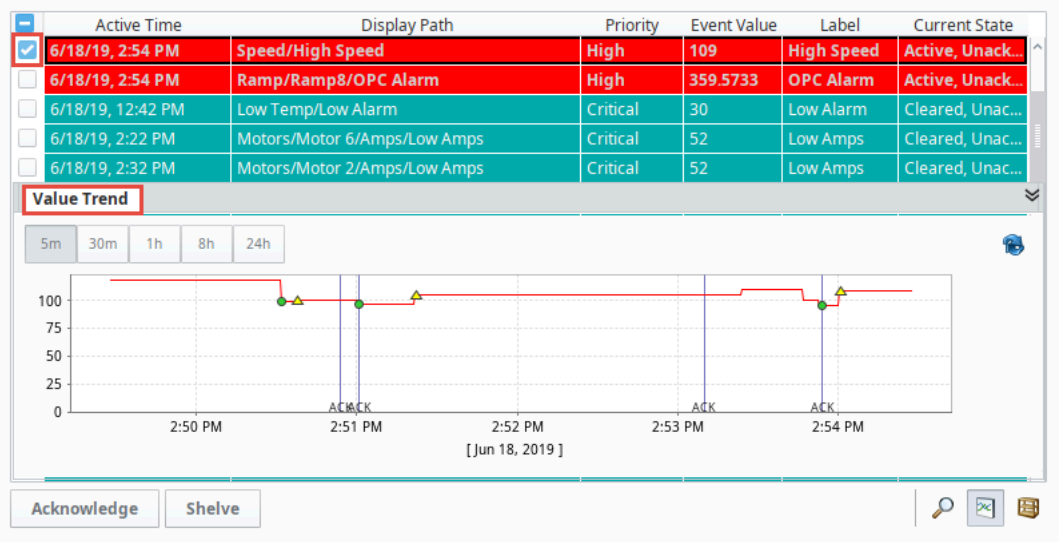
> **NOTE**
>
> The Alarm Status Table component allows you to select an individual alarm, multiple alarms, or the Select All checkbox in the header bar. You can also use the Shift+Click multi select feature to select a range of alarms for acknowledging and shelving. Check one alarm and Shift+Click another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

Interface Elements



Below is a listing of interface elements on the Alarm Status Table component. Note that these interactions are available from a Vision Client, as well as the Designer while Preview Mode is enabled.

Name	Description
Selecting an Entry	Click on an entry in the to select it. The Checkboxes on the left of an entry can also be used to select the entry. Holding Shift while clicking allows for selecting a range of entries.
Header	Entries in the Alarm Status Table can be sorted by each column. Simply click on the desired column header to sort by that column. Holding the **Ctrl** key while clicking on column headers will allow for sorting across multiple columns. Columns can be reordered in the Vision Client by simply dragging and dropping them. In addition, right-clicking on the header will bring up a list of available columns to show or hide. 
Select All/Clear Selection	The checkbox in the upper left corner of the component can be used to select all entries in the table, as well as clear selection from all entries.
Acknowledge Button	Pressing this button will **acknowledge** the selected alarm(s).
Shelve Button	Pressing this button will **shelve** the selected alarm(s).
Inspect Toggle	Pressing this toggle will bring up the Inspection panel, allowing you to view more details on the selected alarm.
Alarm Trend Toggle	Pressing this toggle will bring up a chart, showing the recent historical values of the selected entries. This feature requires that the tag the alarm is configured on is being recorded by the Historian system. In addition, creating an **Alarm Journal Profile** and setting the Alarm Status Table's **Journal Name** property will create annotations on the chart, highlighting

Name	Description
	past alarm events. 
Shelved Alarms Toggle	Pressing this toggle will display a panel that shows all currently shelved alarms in the system. From here shelved alarms can be unshelved.

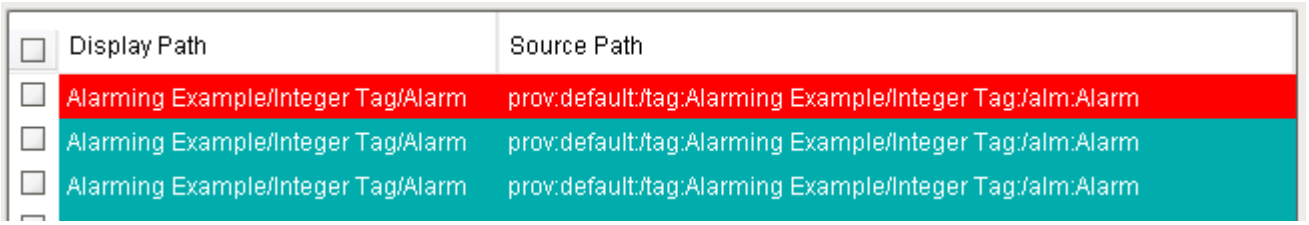
Filtering

The Property Editor has a dedicated Filtering category of properties where you can configure determine which alarms appear on the component. By default, all alarms within the system appear on the table, but in many cases it can be desirable to only show a subset of alarms on the component at a time. Strings can be entered in these fields including wildcards to show specific subsets of alarms. These filtering properties can also be bound so the paths are dynamically created. Finally, you can enter multiple comma-delimited values to filter by.

Display Path Filtering

The **Display Path** can be customized on each alarm. The default value for an alarm's Display Path is a tag path that leads to the name of the alarm. The image below, the top row has an alarm named **Alarm**, and is located on a tag path of **Alarming Example/Integer Tag**, thus the Display path will resolve to '**Alarming Example/Integer Tag/Alarm**'. However, the Display Path can be customized when configuring the alarm. This is generally utilized to display readable messages as to what the issue is.

To filter entries on the table by the Display Path, simply set a value on the **Display Path Filter** property. The * wildcard can be used in the Display Path Filter. See the Source Path Filter Examples table below for more information.



Source Path Filtering

The **Source Path** is also a path to the alarm, but also shows the Tag Provider the alarm is located in. Again using the image above, if the name of the Tag provider is 'default', then the source path would resolve to 'prov:**default**:/tag:**Alarming example/Integer Tag**:/alm:**Alarm**'. Unlike the Display Path, the Source Path on an alarm can never be overridden.

To filter entries on the table by Source Path, simply set a value on the **Source Filter** property. The * wildcard can be used in the Source Filter. See the Source Path Filter Examples table below for more information.

Source Path Filter Examples

The source path is made up of the alarm name, the tag path to the host tag, and the Tag Provider the tag resides in. This means the Source Filter property can be used to filter on tag folders and Tag Providers. For example, you may want the table to only show alarms that contain a certain term or folder path. The table below demonstrates some filters as well as some hypothetical results.

Example Filter	Result
`prov:tagProvider:/tag:Inputs/PS_1:/alm:MyAlarm`	Retrieve alarm information from the alarm `prov:tagProvider:/tag:Inputs/PS_1:/`
`*PS_1:/alm:MyAlarm`	Retrieves alarm information from any path Thus the following paths would be returne `prov:tagProvider:/tag:Inputs/PS_1:/` `prov:tagProvider:/tag:anotherFolder`
`prov:tagProvider:/tag:PS_*`	Retrieves alarm information from any sour "`prov:tagProvider:/tag:PS_`", such as: `prov:tagProvider:/tag:PS_1:/alm:MyA` `prov:tagProvider:/tag:PS_2/Tag2:/al`
`*MyAlarm*`	Retrieves any alarm information that has

State Filtering

The component can also filter entries based on the state of an alarm event. For example, the component can be configured to show only active alarms by enabling the **Show Active and Unacked** and **Show Active and Acked** properties.

Priority Filtering

The component can also filter alarm events based on a minimum priority level, allow the component to ignore lower priority alarm events. This is handled by the **Min Priority** property.

Alarm Property Filtering

Entries in the table can be filtered base on values of alarm properties by using the **filterAlarm** extension function. This includes Alarm Associated Data properties. See **How to Filter by Associated Data on the Vision Alarm Status Table**.

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), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Chart Resolution	The resolution for the ad-hoc 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
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Deprecated
Display Path Filter	Filter alarms by alarm display path, falling back	String	.displayPathFilter	Filters

Name	Description	Property Type	Scripting	Category
	to the source path if a custom display path isn't set. Specify multiple paths by separating them with commas. Supports the wildcard "*", which represents any number of characters. In the example below, only alarms that contained "High Temperature Alarm" would appear in the table. `*High Temperature Alarm*`			
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

Name	Description	Property Type	Scripting	Category
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

Name	Description	Property Type	Scripting	Category
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

Name	Description	Property Type	Scripting	Category
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 milliseconds 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.	Color	.selectionColor	Appearance

Name	Description	Property Type	Scripting	Category
	See **Color Selector**.			
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. New entries added to this dataset will be selectable by users when they attempt to shelve an alarm on the component. Allowable units are second, minute, hour, or day.	Dataset	.shelvingTimes	Data
Show Ack Button	Show the acknowledge button on the footer panel.	boolean	.showAck	Appearance
Show Active and Acked	Show alarms that are active and acknowledged.	boolean	.activeAndAcked	Filters
Show Active and Unacked	Show alarms that are active and unacknowledged.	boolean	.activeAndUnacked	Filters

Name	Description	Property Type	Scripting	Category
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

Name	Description	Property Type	Scripting	Category
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, causing the table to only show alarms that match the filter. Specify multiple paths by separating them with commas. Supports the wildcard "*". See Source Path Filter Examples.	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

Name	Description	Property Type	Scripting	Category
Table Font	The font for the table rows.	Font	.font	Appearance
Table Header Alignment	The alignment for each column in the table header.	int	.headerAlignment	Appearance
Table Header Font	The font for the table header.	Font	.tableHeaderFont	Appearance
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

See the **Vision - Alarm Status Table Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

- **Vision Component Customizers**

Examples

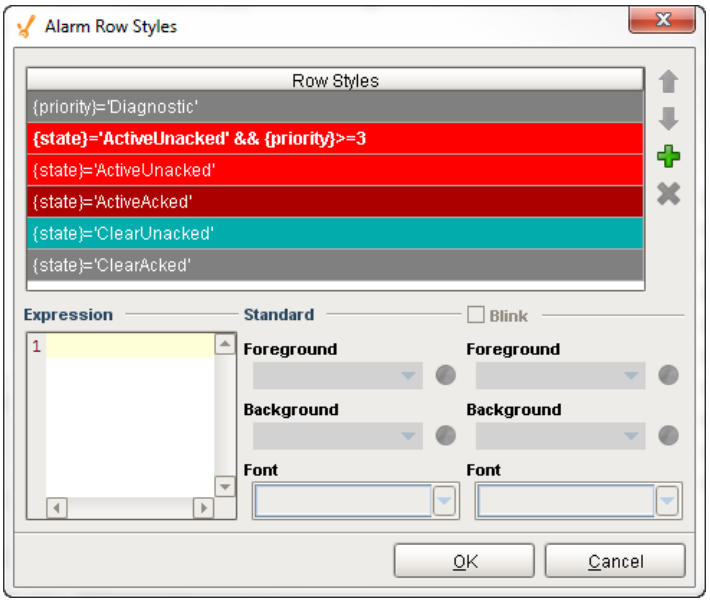
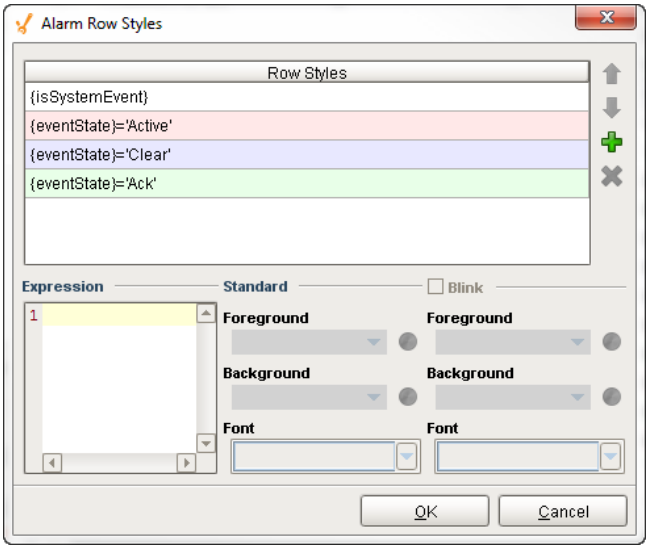
Example 1 - Filter by Associated Data

Click here to see the User Manual page on filtering the Alarm Status Table by Alarm Associated Data,

Example 2 - Restrict Acknowledgement

Click here to see the User Manual page on Restricting Acknowledgement on the Alarm Status Table.

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.

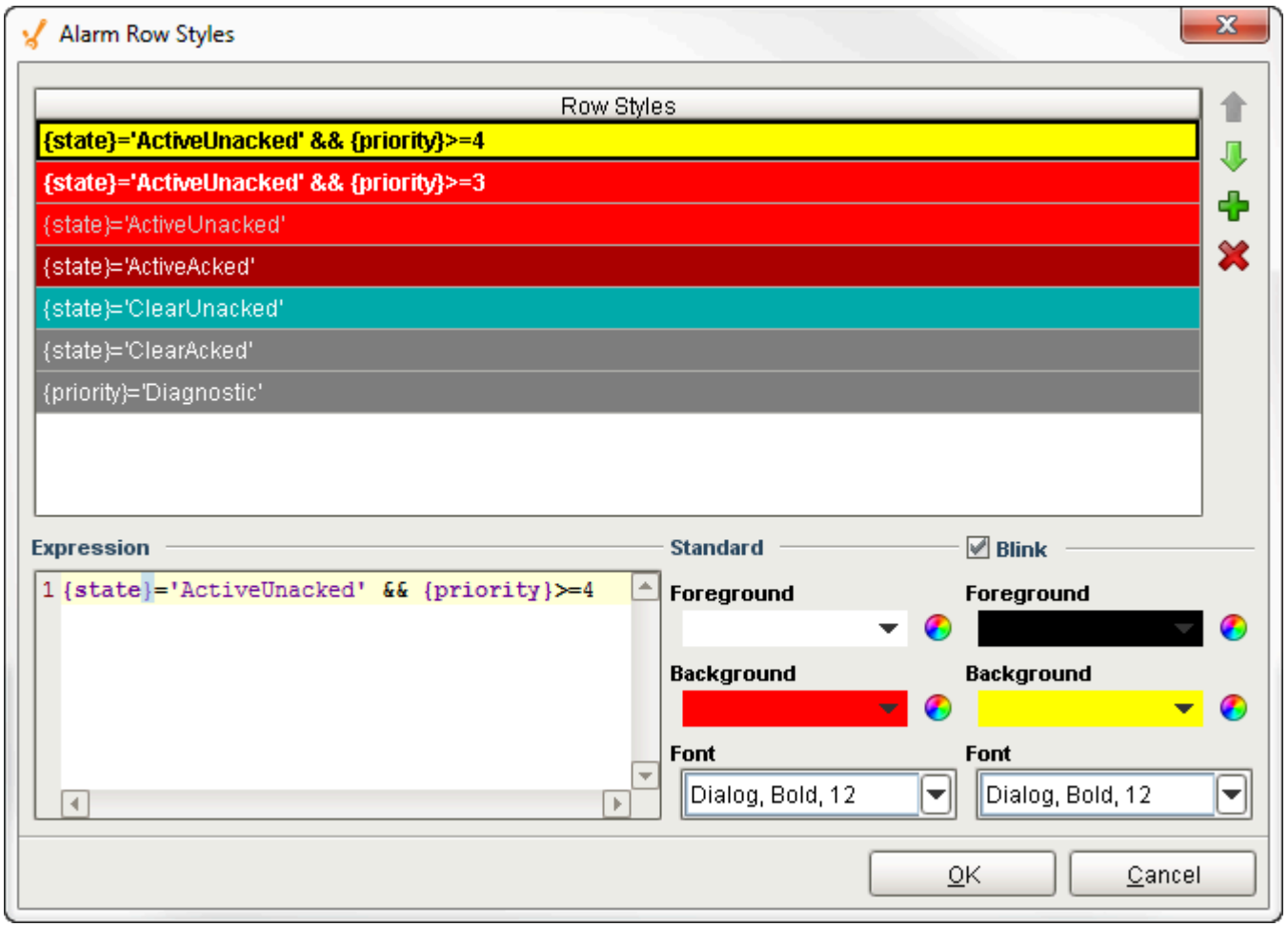
Property Descriptions

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.

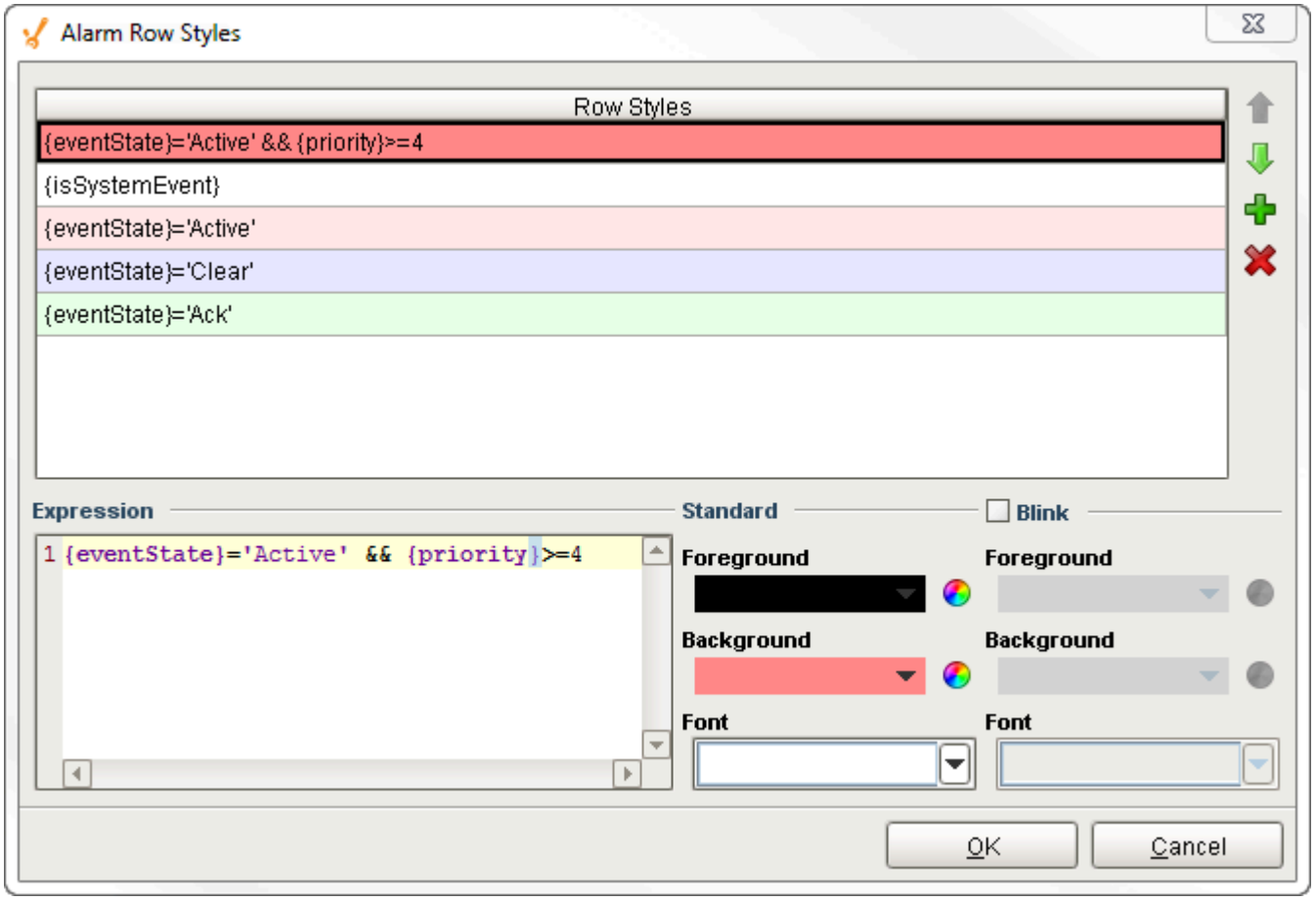
Examples

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 Status Table Scripting Functions

This page details the various component and extension functions available for **Vision's Alarm Status Table component**.

Component Functions

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

- Description
  - This specialized print function will paginate the table onto multiple pages. This function accepts keyword-style invocation.
- Keyword Args
  - boolean **fitWidth** - If true, the table's width will be stretched to fit across one page's width. Rows will still paginate normally. If false, 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.

.getAlarms()

- Description
  - Returns a dataset of the alarms currently displayed in the Alarm Status Table component. The columns will be: EventId, Source, DisplayPath, EventTime, State, and Priority.
- Keyword Args
  - None
- Return
  - Dataset - A dataset of alarms.

Extension Functions

createPopupMenu

- Description
  - Returns a popup menu that will be displayed when the user triggers a popup menu (right click) in the table. Use **system.vision.createPopupMenu()** to build and return the 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, created by **system.vision.createPopupMenu**, to display.

createPopupMenu Example

The createPopupMenu extension function can be used to add custom right-click menus to the Alarm Status Table. For example, the script below creates a menu with "Acknowledge" and "Shelve" options:

```
def ack_selected(event=None):
    # Example acknowledge logic here
    system.vision.showMessage("Acknowledged")

def shelve_selected(event=None):
    # Example shelve logic here
    system.vision.showMessage("Shelved")

menuItemNames = ["Acknowledge", "Shelve"]
menuItemFunctions = [ack_selected, shelve_selected]

# Create and return the popup menu.
# The Alarm Status Table will display this menu on right-click.
return system.vision.createPopupMenu(menuItemNames, menuItemFunctions)
```

filterAlarm

- Description
    - Called for each event loaded into the alarm status table. Return false to hide this event from the table. This code is executed in a background thread so it has a minimal impact on client performance.
- Parameters
    - Component **self**- A reference to the component that is invoking this function.
    - PyAlarmEvent **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.

filterAlarm Example

The filterAlarm Extension Function can be used to filter results on the table based on Alarm Associated Data, allowing you to devise custom filtering criteria. For example, if you wanted the table to only show alarms that had an associated data property named "Production", you could add the following:

Filtering Example

```
group = alarmEvent.get("Group")
if group == "Production":
    return True
return False        # It is important to always include logic where False can be returned for alarm events that don't match your criteria
```

If the script fails to compile, then the table will show alarms as if a filterAlarm script was not configured.

isAcknowledgeEnabled

- Description
  - Returns a boolean that represents whether the selected alarm can be acknowledged
- Parameters
  - Component **self**- A reference to the component that is invoking this function.
  - List **selectedAlarmEvents** - The alarm events selected on the Alarm Status Table. For an individual alarmEvent, call alarmEvent.get('propertyName') to inspect. Common properties: 'name','source','priority'.
- Return
  - Boolean- Returns true or false for every alarm event in the table.

isShelvedEnabled

- Description
  - Returns a boolean that represents whether the selected alarm can be shelved.
- Parameters
  - Component **self**- A reference to the component that is invoking this function.
  - List **selectedAlarmEvents** - The alarm events selected on the Alarm Status Table. For an individual alarmEvent, call alarmEvent.get('propertyName') to inspect. Common properties: 'name', 'source', 'priority'.
- Return
  - Boolean- Returns true or false for every alarm event in the table.

onDoubleClicked

- Description
  - Called when an alarm is double-clicked on to provide custom functionality.
- Parameters
  - Component **self**- A reference to the component that is invoking this function.
  - Alarm Event **alarmEvent** - The alarm event that was double clicked. For an individual alarmEvent, call alarmEvent.get('propertyName') to inspect. Common properties: 'name', 'source', 'priority'.
- Return
  - None

onAcknowledge

- Description
  - Called when the Acknowledge button is pressed; the script runs before the ack happens. Return False to abort the acknowledgement, return True to continue as normal.
- Parameters
  - Component **self**- A reference to the component that is invoking this function.
  - List **alarms** - A list of the alarms to be acknowledged.
- Return
  - Boolean- Returns true or false for every alarm event that is selected.

onShelve

- Description
  - Called when the Apply button is pressed on the Shelving panel; the script runs before the shelving happens. Return False to abort shelving, 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 shelved.

- Return
    - Boolean - Returns true or false for every alarm event that is selected.

How to Filter by Associated Data on the Vision Alarm Status Table



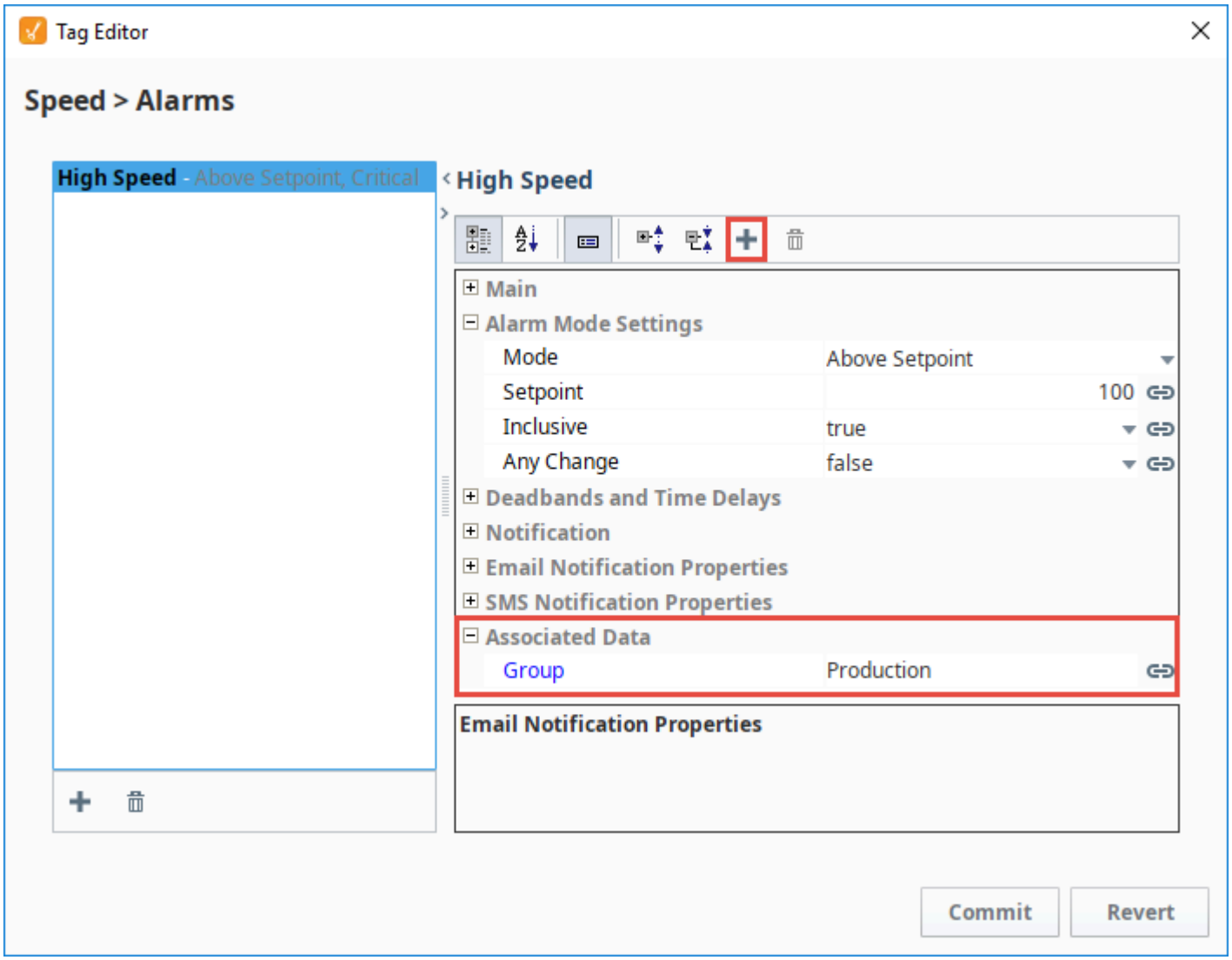
Filtering on Associated Data

Another way to filter alarms in the **Alarm Status Table** is using associated data that was added to an alarm. You can easily and quickly search, filter, and display on specific alarms based on associated data configured in an alarm.

Using a Script to Filter on Associated Data

It's a common design practice to associate alarm groupings on associated data of an alarm. Scripting gives you the ability to filter on associated data, but also gives you the freedom to filter on anything that you want. In the following example, we used a script to filter on associated data, but first we need to add associated data and set up an alarm group.

Here we have a Memory tag called '**Speed**' with a configured alarm called '**High Speed**.' To add associated data, click the **Add** ✚ icon at the top of the Tag Editor, scroll down the list of alarm properties to Associated Data, rename '**New Data**' to '**Group**' and add a static value called '**Production**.' Click **Commit** and save your tag.



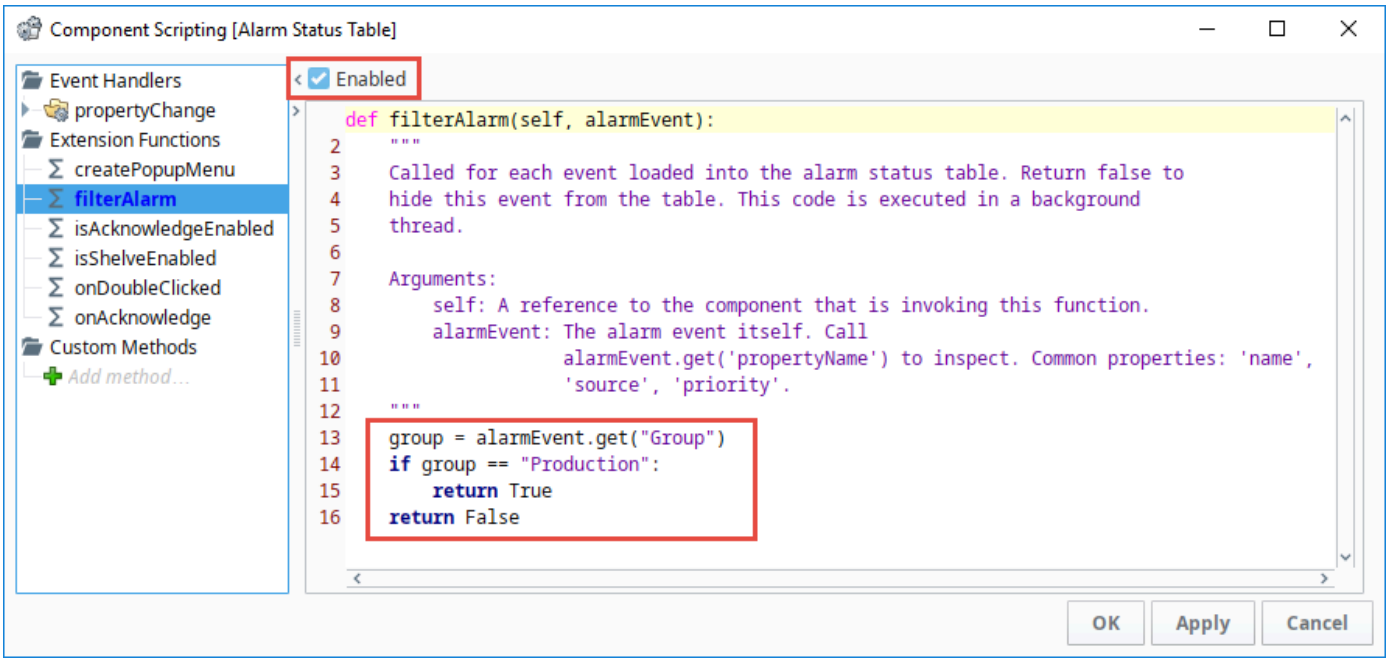
Next, create a script to filter for all alarms in the Alarm Status Table that have the associated data called **'Production**.' Scripting allows you to use the '**filterAlarm**' extension function specifically for filtering on associated data.

1. Right click on your Alarm Status Table component, and scroll down and select **Scripting**.
2. In the Component Scripting window, under **Extension Functions**, select '**filterAlarm**.'
3. Click the **Enabled** checkbox.
4. Enter the code below into the '**filterAlarm**' script. You can filter on anything you want here, but in this example, we are going to filter on '**Production**.'

Extension Function - filterAlarm for 'Production'

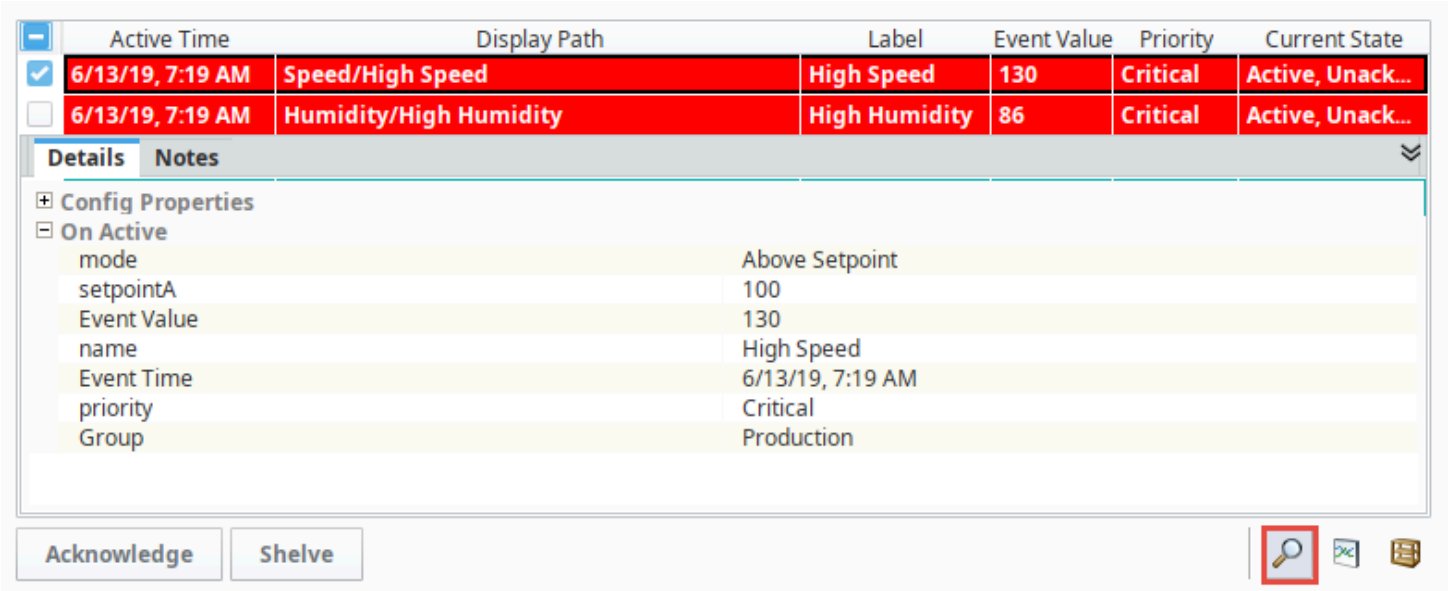
```
group = alarmEvent.get("Group")
if group == "Production":
    return True
return False
```

Your script will look like the image below. This script will only display alarms matching the associated data for '**Production**.' For every alarm matching '**Production**,' it will return **'True'** and show alarm results in the Alarm Status Table. If the associated data does not match '**Production**,' it will return **'False**,' and the Alarm Status Table will be empty. Note, when you're finished filtering on associated data, don't forget to disable your script.



5. Click **OK**.

6. Now the Alarm Status Table below shows all the alarms in the '**Production**' group. Select an alarm and click the **Search** icon to see all the Details about the alarm.



NOTE

- If you see alarms that do not match your associated data, check your filter settings in the Property Editor of the Alarm Status Table. You may need to uncheck the 'Show Clear and

Unacked' and 'Show Clear and Acked' settings depending on what you want your operators to see.

- If you have an error in this filtering script, it will return 'true' for every alarm instance (and show all alarms) instead of displaying many errors to your users. You can find more information about the error from the **Output Console** under the **Tools Menu** in the Designer, and from the Vision Client under **Help > Diagnostics** and selecting the **Console** tab.

How To Restrict Acknowledgement on the Vision Alarm Status Table Component

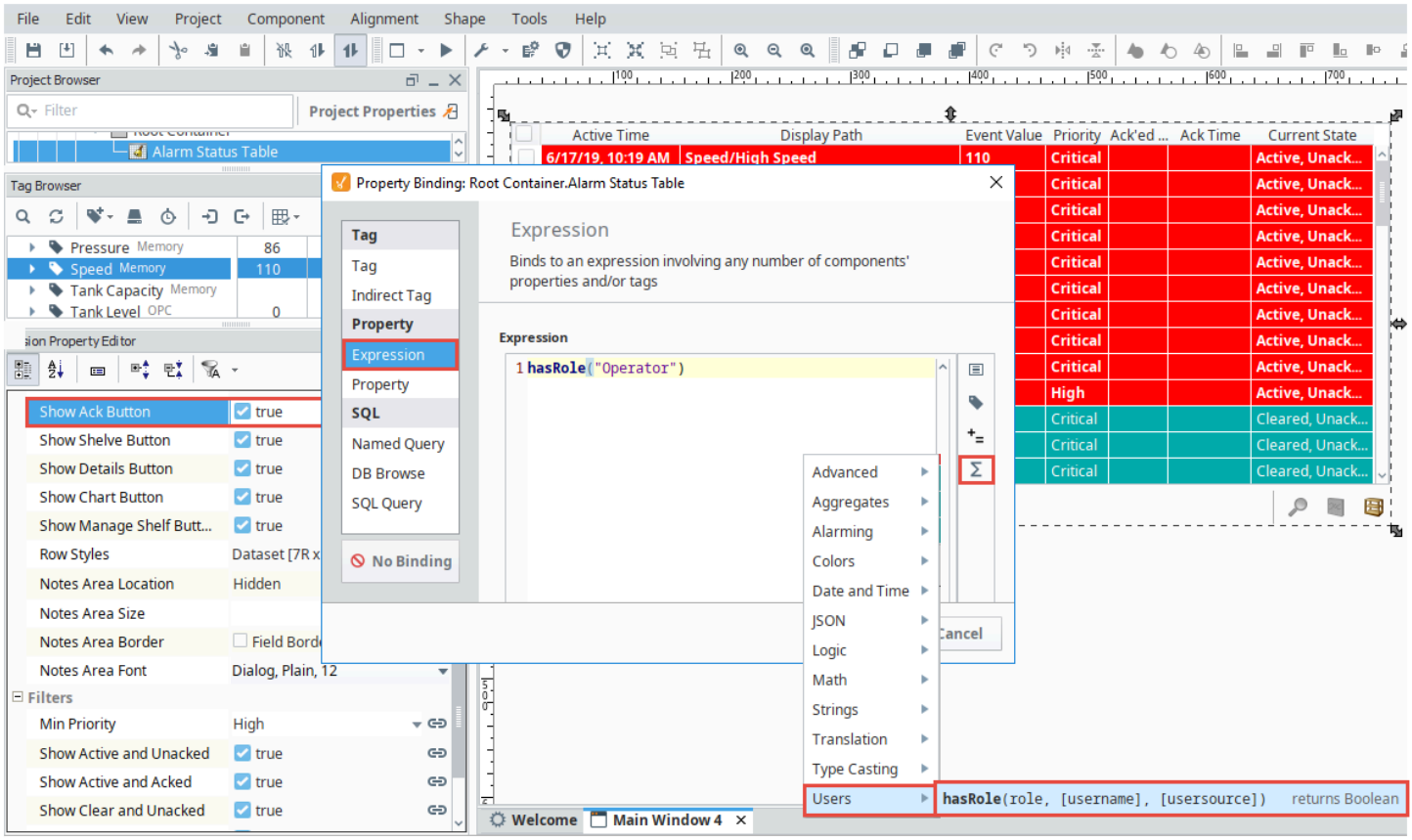
Security for Alarm Acknowledgement

If you want to restrict who can use the Acknowledge button, there is the **Show Ack Button** in the Property Editor that can be set to '**False**.' By setting the **Show Ack Button** to '**false**,' this hides the Acknowledge button on the Alarm Status Table.

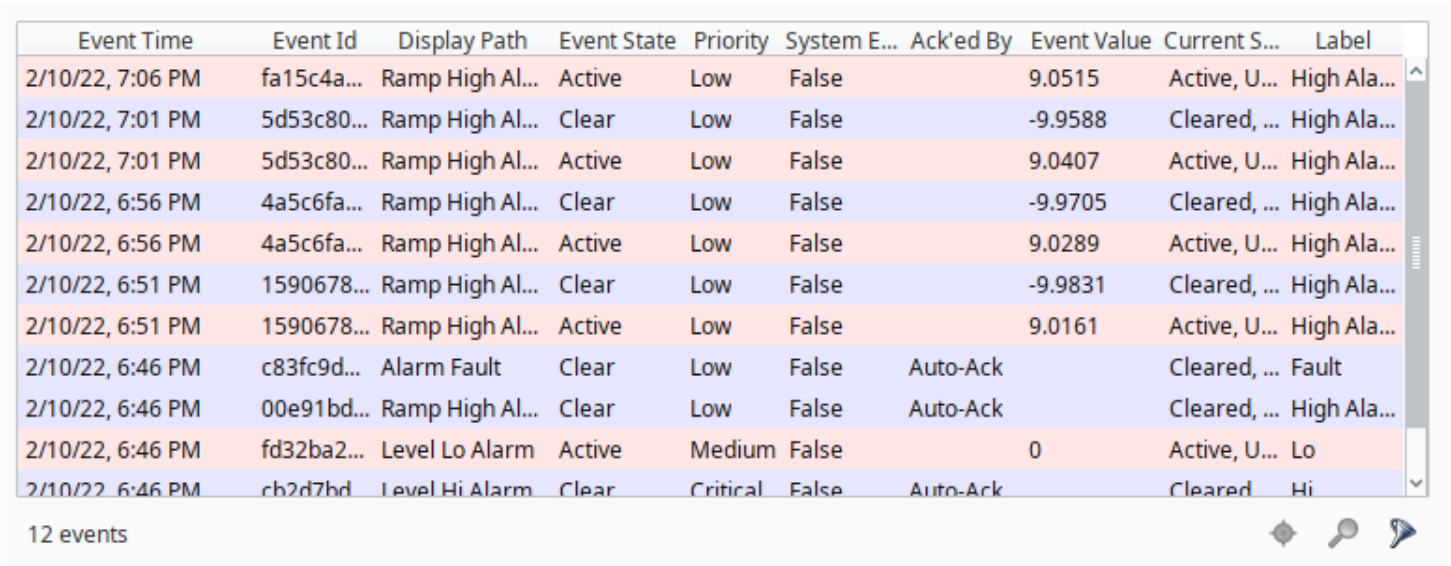
In order for operators to acknowledge alarms, the correct permission must be assigned. This example shows how to set permissions to acknowledge alarms for users in the **Operator** role. You can setup permissions for any role, user and user source in your system.

1. Select the Alarm Status Table component, and click the **Show Ack Button** binding icon to open the Property Binding window.
2. Under **Property Binding Type**, select **Expression**.
3. Click the **Function** Σ icon and scroll down to **Users**, and select '**hasRole**.' This enters the function name.
4. Edit the expression to read: **hasRole("Operator")**
5. Click **OK**.

If you currently have the "Operator" role you will notice that the **Show Ack Button** property is now '**true**,' otherwise, it will be '**false**.'


File Edit View Project Component Alignment Shape Tools Help
Project Browser
Filter
Project Properties
Alarm Status Table
Tag Browser
Pressure Memory 86
Speed Memory 110
Tank Capacity Memory
Tank Level OPC 0
Property Editor
Show Ack Button true
Show Shelve Button true
Show Details Button true
Show Chart Button true
Show Manage Shelf Butt... true
Row Styles Dataset [7R x
Notes Area Location Hidden
Notes Area Size
Notes Area Border Field Bord
Notes Area Font Dialog, Plain, 12
Filters
Min Priority High
Show Active and Unacked true
Show Active and Acked true
Show Clear and Unacked true
Active Time
Display Path
Event Value
Priority
Ack'ed ...
Ack Time
Current State
6/17/19, 10:19 AM
Speed/High Speed
110
Critical
Active, Unack...
High
Cleared, Unack...
Property Binding: Root Container.Alarm Status Table
Tag
Tag
Indirect Tag
Property
Expression
Property
SQL
Named Query
DB Browse
SQL Query
No Binding
Expression
Binds to an expression involving any number of components' properties and/or tags
Expression
1 hasRole("Operator")
Advanced
Aggregates
Alarming
Colors
Date and Time
JSON
Logic
Math
Strings
Translation
Type Casting
Users
hasRole(role, [username], [usersource]) returns Boolean
Cancel
Welcome
Main Window 4

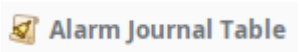
Vision - Alarm Journal Table



Event Time	Event Id	Display Path	Event State	Priority	System E...	Ack'ed By	Event Value	Current S...	Label
2/10/22, 7:06 PM	fa15c4a...	Ramp High Al...	Active	Low	False		9.0515	Active, U...	High Ala...
2/10/22, 7:01 PM	5d53c80...	Ramp High Al...	Clear	Low	False		-9.9588	Cleared, ...	High Ala...
2/10/22, 7:01 PM	5d53c80...	Ramp High Al...	Active	Low	False		9.0407	Active, U...	High Ala...
2/10/22, 6:56 PM	4a5c6fa...	Ramp High Al...	Clear	Low	False		-9.9705	Cleared, ...	High Ala...
2/10/22, 6:56 PM	4a5c6fa...	Ramp High Al...	Active	Low	False		9.0289	Active, U...	High Ala...
2/10/22, 6:51 PM	1590678...	Ramp High Al...	Clear	Low	False		-9.9831	Cleared, ...	High Ala...
2/10/22, 6:51 PM	1590678...	Ramp High Al...	Active	Low	False		9.0161	Active, U...	High Ala...
2/10/22, 6:46 PM	c83fc9d...	Alarm Fault	Clear	Low	False	Auto-Ack		Cleared, ...	Fault
2/10/22, 6:46 PM	00e91bd...	Ramp High Al...	Clear	Low	False	Auto-Ack		Cleared, ...	High Ala...
2/10/22, 6:46 PM	fd32ba2...	Level Lo Alarm	Active	Medium	False		0	Active, U...	Lo
2/10/22, 6:46 PM	cb2d7bd...	Level Hi Alarm	Clear	Critical	False	Auto-Ack		Cleared ...	Hi

12 events

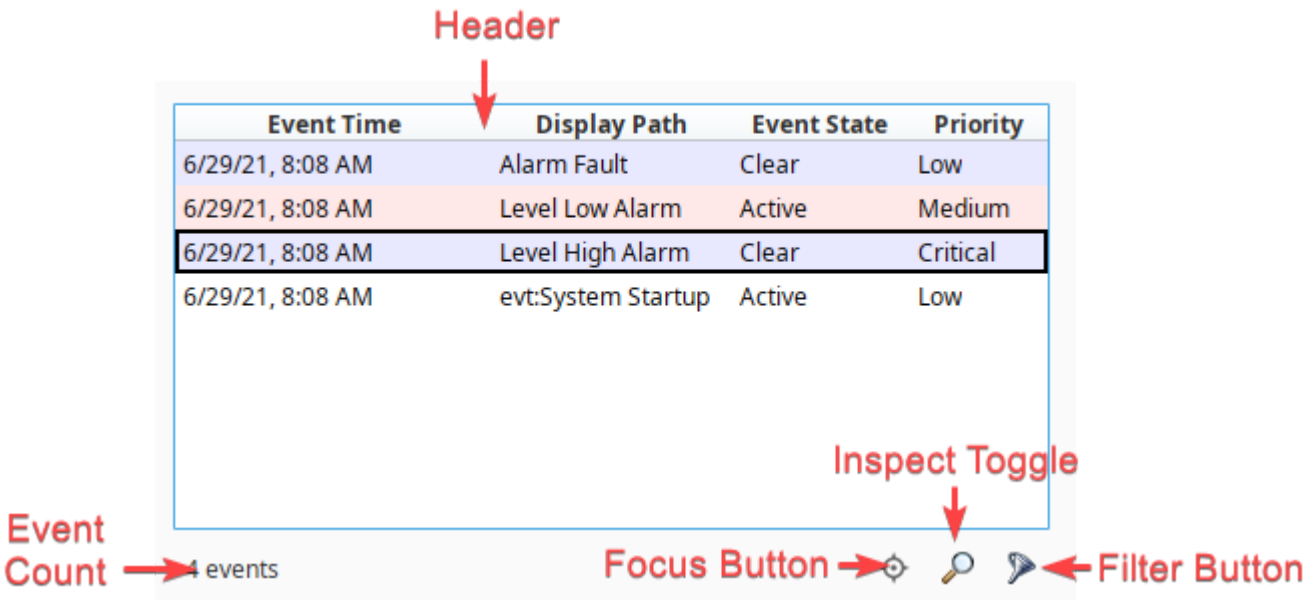
Component Palette Icon

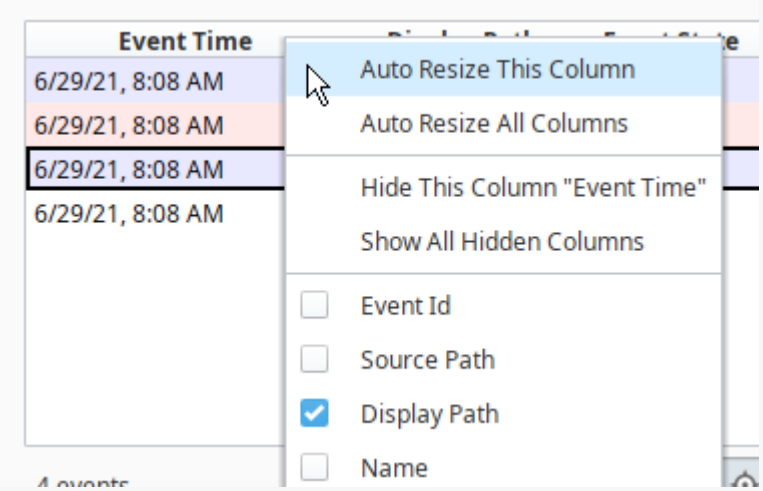


The alarm journal table provides a built-in view to explore alarm history that has been stored in an alarm journal. If you only have one alarm journal specified on your Gateway, then you do not need to specify the journal name. If you have more than one specified, then you need to provide the name of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

The journal table shows the alarm history that is found between the **Start Date** and **End Date** properties. When you first put an alarm journal table on a window, these properties will be set to show the most recent few hours of journal history. Note that without further configuration, the journal table will always show the few hours before it was created. To properly configure an alarm journal table, bind its start and end date properties to something what will update, such as the Date Range component or expressions involving the now expression function. 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.

Interface Elements



Element	Description
Selecting an Event	Click on any event (row) in the table to select it. Some of the other interactions, such as the Inspect Toggle, require that an event is first selected.
Header	Events in the Alarm Journal Table can be sorted by each column. Simply click on the desired column header to sort by that column. Columns can be reordered in the Vision Client by simply dragging and dropping them. In addition, right-clicking on the header will bring up a list of available columns to show or hide. 
Event Count	A count representing the number of events available in the table, accounting for filters applied to the table.
Focus Button	Clicking this button while an event is selected will open a popup with the following to two selections: **Target Alarm Source**: Displays only alarms that match the selected alarm's Source Path. **Target Event Id**: Clears all alarms from the table, except those that have a matching value for the Event Id column. This is commonly used to show only the active, acknowledgement, and clear events for a single alarm, effectively allowing you to see the lifecycle of that alarm instance. Clicking on the Focus Button a second time will end focus filtering.
Inspect Toggle	Clicking this button while an event is selected will open a popup that shows all alarm properties for the selected event.

Element	Description
Filter Button	Clicking this button will call the Filter panel, allowing you to filter the results in the table based on event type, priority, or search string.

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), and Field 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

Name	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated
Disabled Events	If enabled, will show events created by alarms being disabled.	boolean	.includeDisabledEvents	Filters
Display Path Filter	Filter alarms by alarm display path, falling back to the source path if display path isn't set. Specify multiple paths by separating them with commas. Supports the wildcard `*`. Refer to the **Filtering Alarms** section on the Alarming page for more information on Display Path Filter syntax.	String	.displayPathFilter	Filters
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common

Name	Description	Property Type	Scripting	Category
Enabled Events	If enabled, will show events created by alarms being enabled.	boolean	.includeEnabledEvents	Filters
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

Name	Description	Property Type	Scripting	Category
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
Order by Ascending	Sorts alarms by the event time in ascending order when true and descending order when false. Default is true.	boolean	.orderByEventTimeAsc	Filters
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Read Timeout	The timeout, in milliseconds, for running the	int	.readTimeout	Behavior

Name	Description	Property Type	Scripting	Category
	alarm history query.			
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

Name	Description	Property Type	Scripting	Category
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 `*`. Refer to the **Filtering Alarms** section on the Alarming page for more information on Source Filter syntax.	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	boolean	.includeSystemEvents	Filters

Name	Description	Property Type	Scripting	Category
	startup and shutdown.			
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
Table Header Font	The font for the table header rows.	Font	.tableHeaderFont	Appearance
Table Header Alignment	The alignment for each column in the table header.	int	.headerAlignment	Appearance
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

See the **Vision - Alarm Journal Table Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

- **Vision Component Customizers**

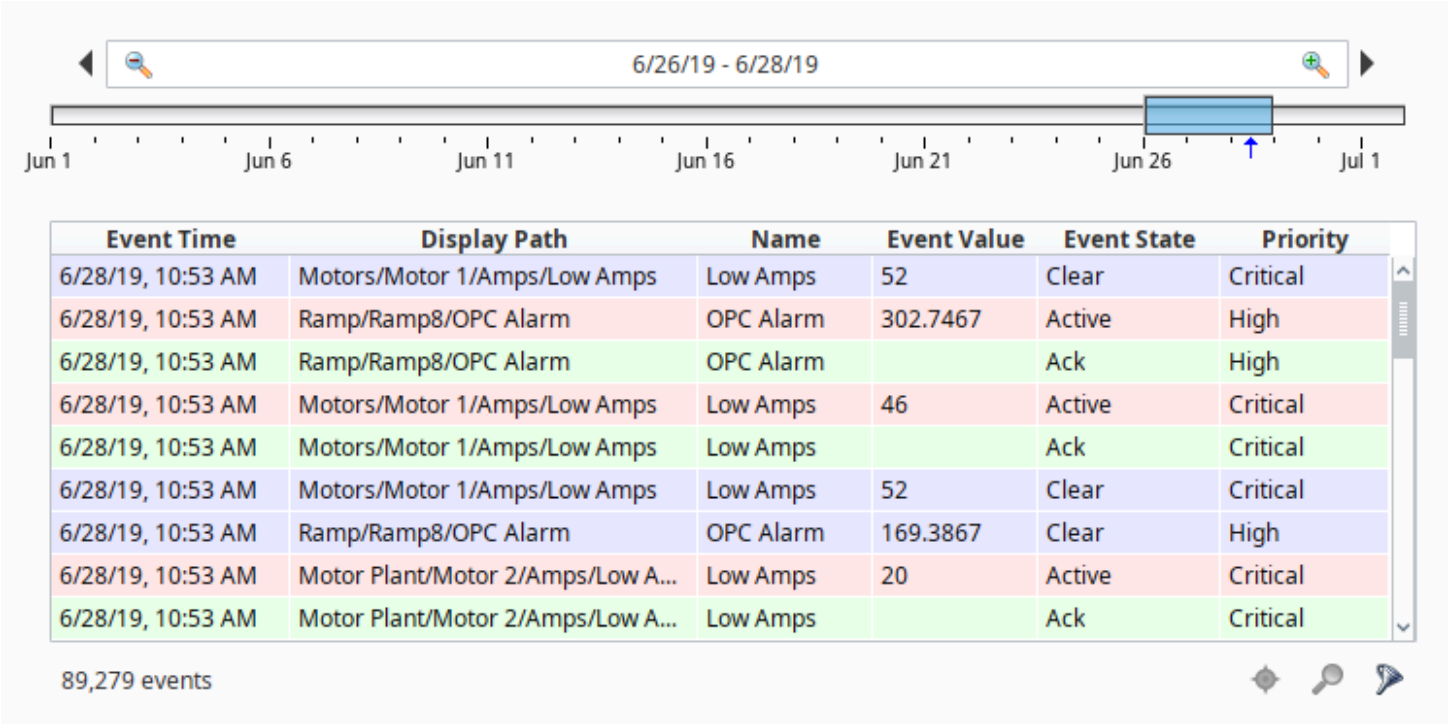
Vision Alarm Journal - Row Styles

INDUCTIVE UNIVERSITY

Alarm Journal - Row Styles

Watch the video

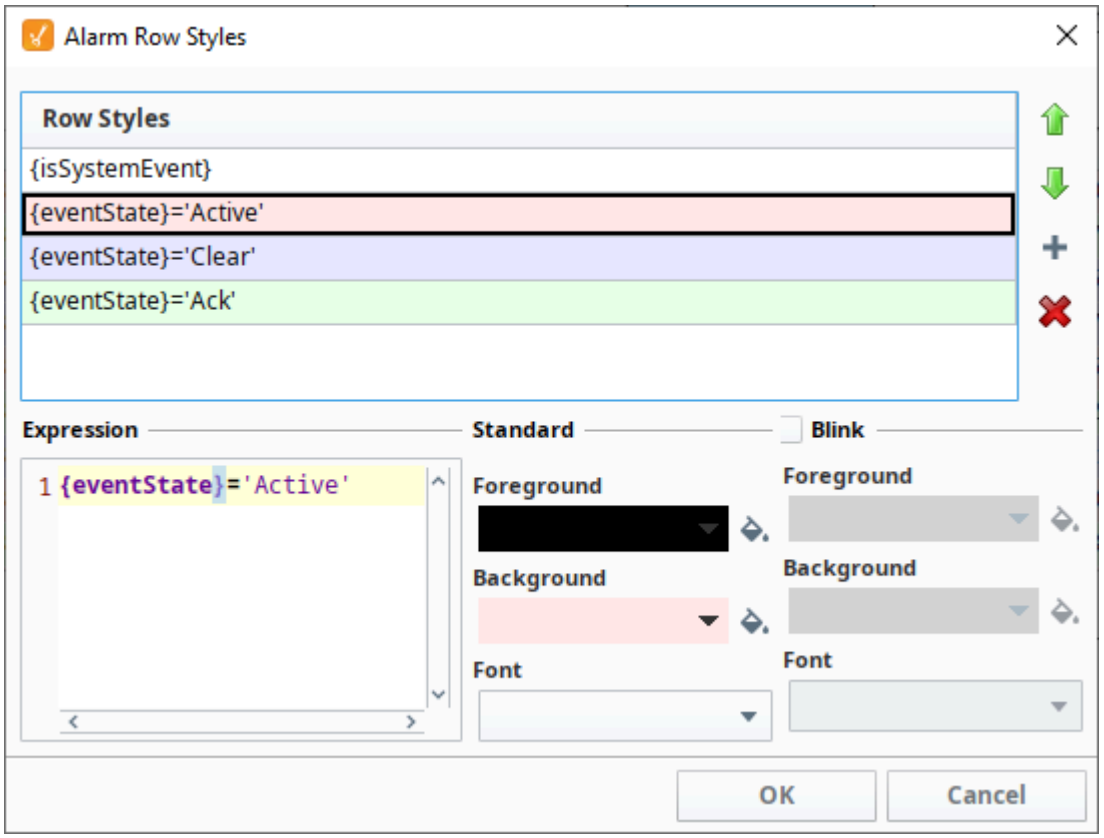
The Alarm Journal Table allows you customize row styles for different states of alarm history. Just like the Alarm Status Table, the **Alarm Journal Table** comes with a particular set of colors associated with each of the alarm states as shown in the image below. You can change these colors for each of the states by going to the Alarms Styles Customizer that the Alarm Journal Table component provides.



Customizing Alarm Row Styles

Alarm Row Styles is where you can modify an existing row style, add more styles, or delete a style. The Alarm Row Styles Customizer gives you a head start for building a new expression. The expression allows you to do any evaluation you want using any filter properties of the alarm: Priority, State, Display Path, Active Time, and Clear Time.

In the Designer, right click on the **Alarm Journal Table** component, go to **Customizers > Alarm Row Styles** to see the default row styles. Alarm Row Styles is an ordered list and each style has an expression. How it works is, the first style that returns **'True'** for a given alarm is the one that is going to be used. So you want to make sure the order is the correct order that you want. If you want to change the order, select a row and click on the up or down arrow icons. Click on each of the row styles to view the expression associated with the row style.



Creating Row Styles for Different Alarm States

The main reason to create a new row style is for overlapping conditions for styling. Let's create another state alarm with a new row style and state.

1. With the Alarm Row Styles customizer open, click on the **Add** icon.

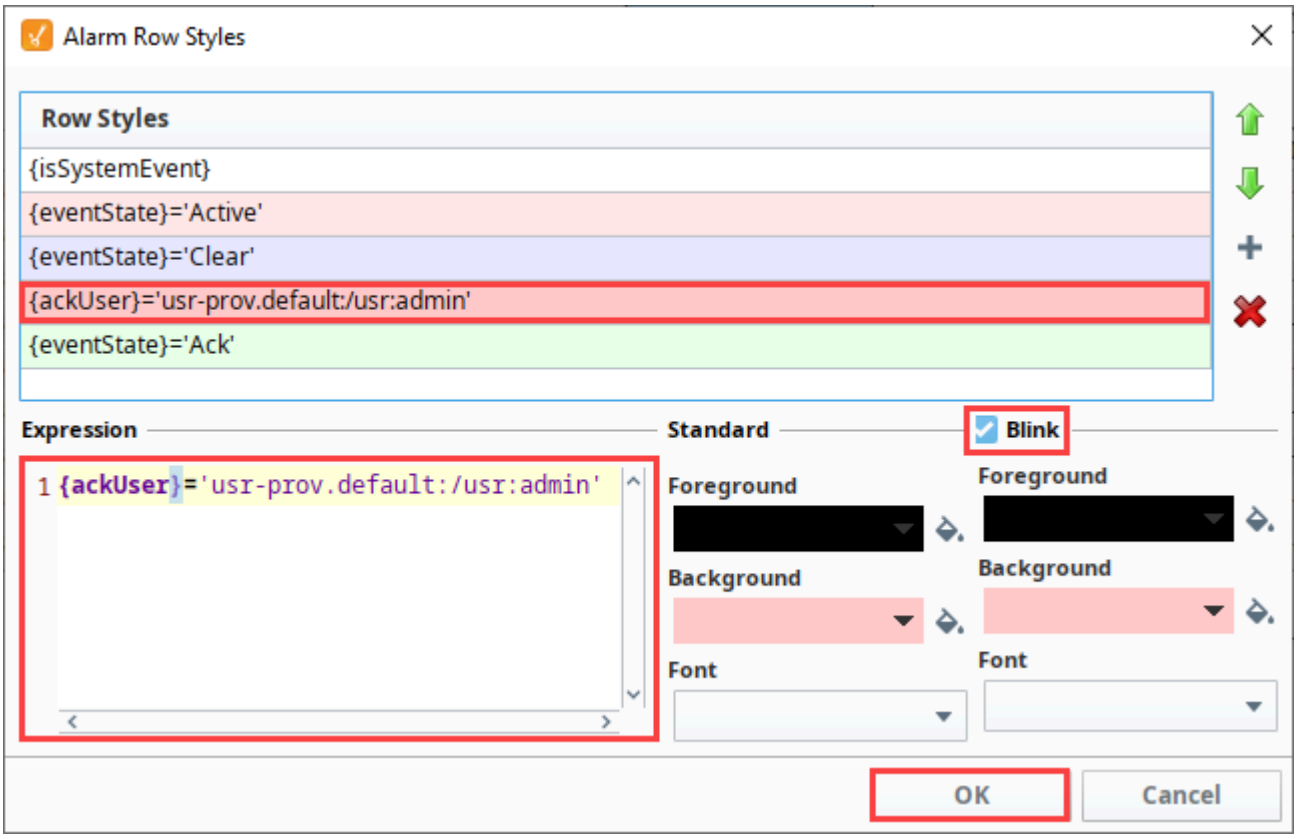
2. By default, the new alarm state is added to the bottom of the list. The following expression creates a new state for alarms '**Ack by the user admin**.' Copy and paste the expression into the Expression area of the Row Styles window.

   ```
   {ackUser}='usr-prov.default:/usr:admin'
   ```

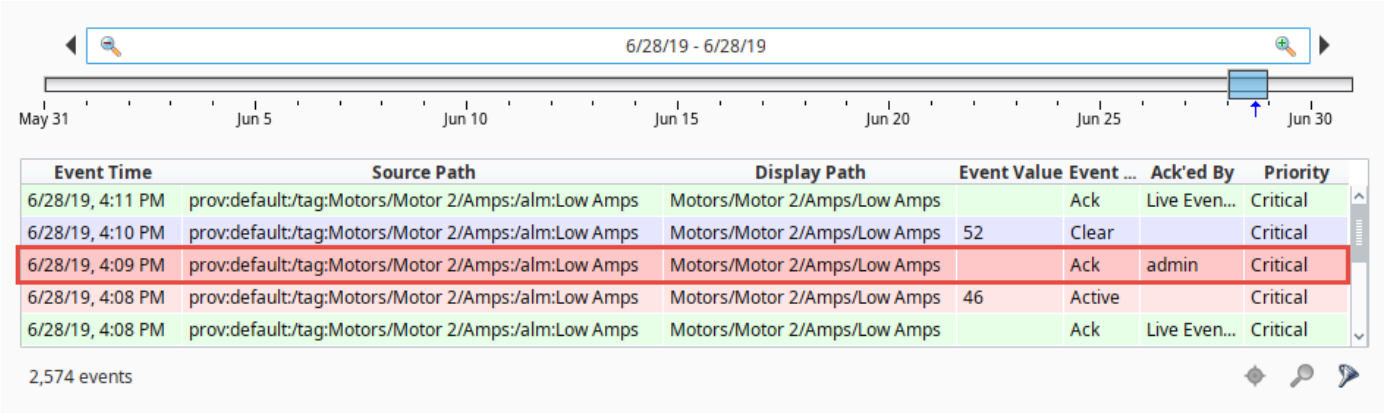
3. To make this alarm state catch the attention of the operator, let's make the row style standout by making it blink. Make the foreground color '**Black**' and the background color '**Yellow**.' Check the **Blink** box and make the foreground color '**Black**' and background color '**Red**.'

4. Move the new state for '**Ack by the user admin**' above the '**Ack**' state so it gets evaluated first, otherwise if '**Ack**' state is evaluated first, it will become '**True**' first and the new state '**Ack by the user admin**' will never be evaluated.

5. Press **OK** to save your updates.



6. Now, when the user '**admin**' acknowledges an alarm, it will blink yellow and red in the Alarm Journal Table.



Refer to the **Tag Alarm Properties** page to learn more about alarm properties.

Vision - Alarm Journal Table Scripting Functions

This page details the various component and extension functions available for **Vision's Alarm Journal Table component**.

Component Functions

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

- Description
  - This specialized print function will paginate the table onto multiple pages. This function accepts keyword-style invocation.
- Keyword Args
  - boolean **fitWidth** - If true, the table's width will be stretched to fit across one page's width. Rows will still paginate normally. If false, 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.

.getAlarms()

- Description
  - Returns a dataset of the alarms currently displayed in the Alarm Journal Table component. The columns will be: EventId, Source, DisplayPath, EventTime, State, Priority and IsSystemEvent
- Keyword Args
  - None
- Return
  - Dataset - A dataset of alarms.

Extension Functions

createPopupMenu

- Description
  - Returns a popup menu that will be displayed when the user triggers a popup menu (right-click) in the table. Use **system.vision.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.vision.createPopupMenu()**

filterAlarm

- Description
  - Called for each event loaded into the alarm status table. Return false to hide this event from the table. This code is executed in a background thread.

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

onDoubleClicked

- Description
    - Called when an alarm is double-clicked on to provide custom functionality. Does not return a value.
- Parameters
    - Component **self** - A reference to the component that is invoking this function.
- Alarm Event **alarmEvent** - The alarm event itself. Call alarmEvent.get('propertyName') to inspect. Common properties: 'name', 'source', 'priority'.
- Return
    - None

Vision - Buttons Palette

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

Button

2 State Toggle

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

Multi-State Button

This component consists of two or more buttons arranged in a column, row, or grid.

One-Shot Button

The One-Shot button is great for telling a PLC to do something. It simply writes a value, and then waits for it to be...

Momentary Button

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

Toggle Button

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

Check Box

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

Radio Button

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

Tab Strip

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

Vision - Button



Component Palette Icon:



The Button component can be configured to open and/or close windows, write to tags, and run scripts when triggered by an event handler.

Properties

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

Property	Description	Property Type	Scripting	Category
	Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.			
Default Button	If true, this button will be activated when the user presses Enter on the window.	boolean	.defaultBtn	Behavior
Disabled Image Path	The relative path of the image to be displayed when this component is not enabled.	String	.disabledPath	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Fill Area?	Controls whether or not this button's internal area is filled.	boolean	.contentAreaFilled	Appearance
Focusable	If a button is not focusable, you will not be able to interact with it with the keyboard. This means you can't "tab" over to it.	boolean	.focusable	Behavior

Property	Description	Property Type	Scripting	Category
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See **Color Selector**.	Color	.foreground	Appearance
Horizontal Alignment	The horizontal alignment of the button's contents (text and/or image).	int	.horizontalAlignment	Layout
Horizontal Text Position	The horizontal position of the button's text relative to its image.	int	.horizontalTextPosition	Layout
Icon-Text Spacing	The space (in pixels) between the icon (if any) and the text (if any).	int	.iconTextGap	Appearance
Image Path	The relative path of the image.	String	.path	Appearance
Margin	The space between a button's text and its borders.	Insets	.margin	Layout
Mnemonic	A single letter that will activate	String	.mnemonicChar	Behavior

Property	Description	Property Type	Scripting	Category
	the button using 'ALT-mnemonic'.			
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 bindings on this component.	QualityCode	.quality	Data
Rollover	If true, the button may indicate that the mouse is hovering over it.	boolean	.rolloverEnabled	Behavior
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Text	Text of this component.	String	.text	Appearance
Vertical Alignment	The vertical alignment of the button's contents (text and/or image).	int	.verticalAlignment	Layout
Vertical Text Position	The vertical position of the	int	.verticalTextPosition	Layout

Property	Description	Property Type	Scripting	Category
	button's text relative to its image.			
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any tag bindings on this component.	int	.dataQuality	Data
Opaque	If true, button will be opaque. Default is false.	boolean	.opaque	Common

Scripting

See the **Vision - Button Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Examples

Styled Button



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

Styled Button



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

Vision - Button Scripting Functions

This page details the various component and extension functions available for **Vision's Button component**.

Component Functions

.doClick()

- Description
  Virtually "clicks" the button, meaning that its actionPerformed event handler will run.
- Parameters
  Nothing
- Return
  Nothing
- Scope
  Client

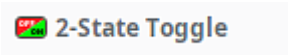
Extension Functions

This component does not have any extension functions associated with it.

Vision - 2 State Toggle



Component Palette Icon:



This button component will toggle a value between two states, such as On/Off, Stop/Run, etc. To toggle between more than two states, use the **Multi-State Button**.

Properties

Property	Description	Property Type	Scripting	Category
Background Color	The background color of the button. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.buttonBG	Appearance
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Confirm Text	The message displayed in the confirmation box if Confirm? is true.	String	.confirmText	Behavior
Confirm?	If true, a confirmation box	boolean	.confirm	Behavior

Property	Description	Property Type	Scripting	Category
	will be shown.			
Control Value	Bind this to the tag that controls the state. (Typically, this is bound to the same location as Indicator Value).	int	.controlValue	Data
Current State	Read-only property that shows the button's current state (0 or 1)	int	.state	Data
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Disabled Image Path	The relative path of the image to be displayed when this component is not enabled.	String	.disabledPath	Appearance
Enabled	If disabled, a component	boolean	.componentEnabled	Common

Property	Description	Property Type	Scripting	Category
	cannot be used.			
Fill Area?	Controls whether or not this button's internal area is filled.	boolean	.contentAreaFilled	Appearance
Focusable	If a button is not focusable, you will not be able to interact with it with the keyboard.	boolean	.focusable	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See **Color Selector**.	Color	.foreground	Appearance
Horizontal Alignment	The horizontal alignment of the button's contents (text and/or image)	int	.horizontalAlignment	Layout
Horizontal Text Position	The horizontal position of the button's text relative to its image.	int	.horizontalTextPosition	Layout
Icon-Text Spacing	The space (in pixels) between the icon (if any)	int	.iconTextGap	Appearance

Property	Description	Property Type	Scripting	Category
	and the text (if any).			
Image Path	The relative path of the image.	String	.path	Appearance
Indicator Value	Bind this to the tag that indicates the current state. (If you don't have separate tags for status and control, this is bound to the same location as Control Value)	int	.indicatorValue	Data
Margin	The space between a button's text and its borders.	Insets	.margin	Layout
Mnemonic	A single letter that will activate the button using 'ALT-mnemonic'.	String	.mnemonicChar	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common

Property	Description	Property Type	Scripting	Category
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Rollover	If true, the button may indicate that the mouse is hovering over it.	boolean	.rolloverEnabled	Behavior
State 1 Value	The value that will be written to controlValue when the button is pushed in state 2.	int	.state1Value	Data
State 2 Value	The value that will be written to controlValue when the button is pushed in state 1.	int	.state2Value	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Text	Text of this component.	String	.text	Appearance
Vertical Alignment	The vertical alignment of the button's contents (text and/or image).	int	.verticalAlignment	Layout

Property	Description	Property Type	Scripting	Category
Vertical Text Position	The vertical position of the button's text relative to its image.	int	.verticalTextPosition	Layout
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated
Opaque	If true, button will be opaque. Default is false.	boolean	.opaque	Common

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Example

2-State Toggle with Styles Configured



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

Vision - Multi-State Button



Component Palette Icon:



This component consists of two or more buttons arranged in a column, row, or grid. Each button corresponds to an integer-valued state and is displayed with the correct style based on the component's Indicator Value. When a button is pressed then released, its value is written to the Control Value.

Properties

Property	Description	Property Type	Scripting	Category
Confirm Text	The message to display in a confirmation box if Confirm? is true. Default is "Are you sure?"	string	.confirmText	Behavior
Confirm?	If true, a confirmation box will be shown.	boolean	.confirm	Behavior
Control Value	Value that controls the state. Typically, this is bound to the same location as the Indicator Value.	int	.controlValue	Data
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Display Style	The display style (rows or columns) for this N-state button.	int	.displayStyle	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common

Property	Description	Property Type	Scripting	Category
Focusable	If false, users cannot interact with the component using the keyboard.	boolean	.focusableEnabled	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Grid Cols	The number of columns if the Display Style is set to "Grid" mode.	int	.gridCols	Appearance
Grid Rows	The number of rows if the Display Style is set to "Grid" mode.	int	.gridRows	Appearance
Horizontal Gap	The horizontal spacing between buttons.	int	.hGap	Appearance
Indicator Value	Value that indicates the current state. Typically, this is bound to the same location as the Control Value.	int	.indicatorValue	Data
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common

Property	Description	Property Type	Scripting	Category
Quality	The data quality code for any tag bindings on this component.	QualityCode	.quality	Data
Rollover	If true, the button may indicate that the mouse is hovering over it.	boolean	.rolloverEnabled	Behavior
States	A Dataset that stores the information for the different states.	Dataset	.states	Behavior
Vertical Gap	The vertical spacing between buttons.	int	.vGap	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

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

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

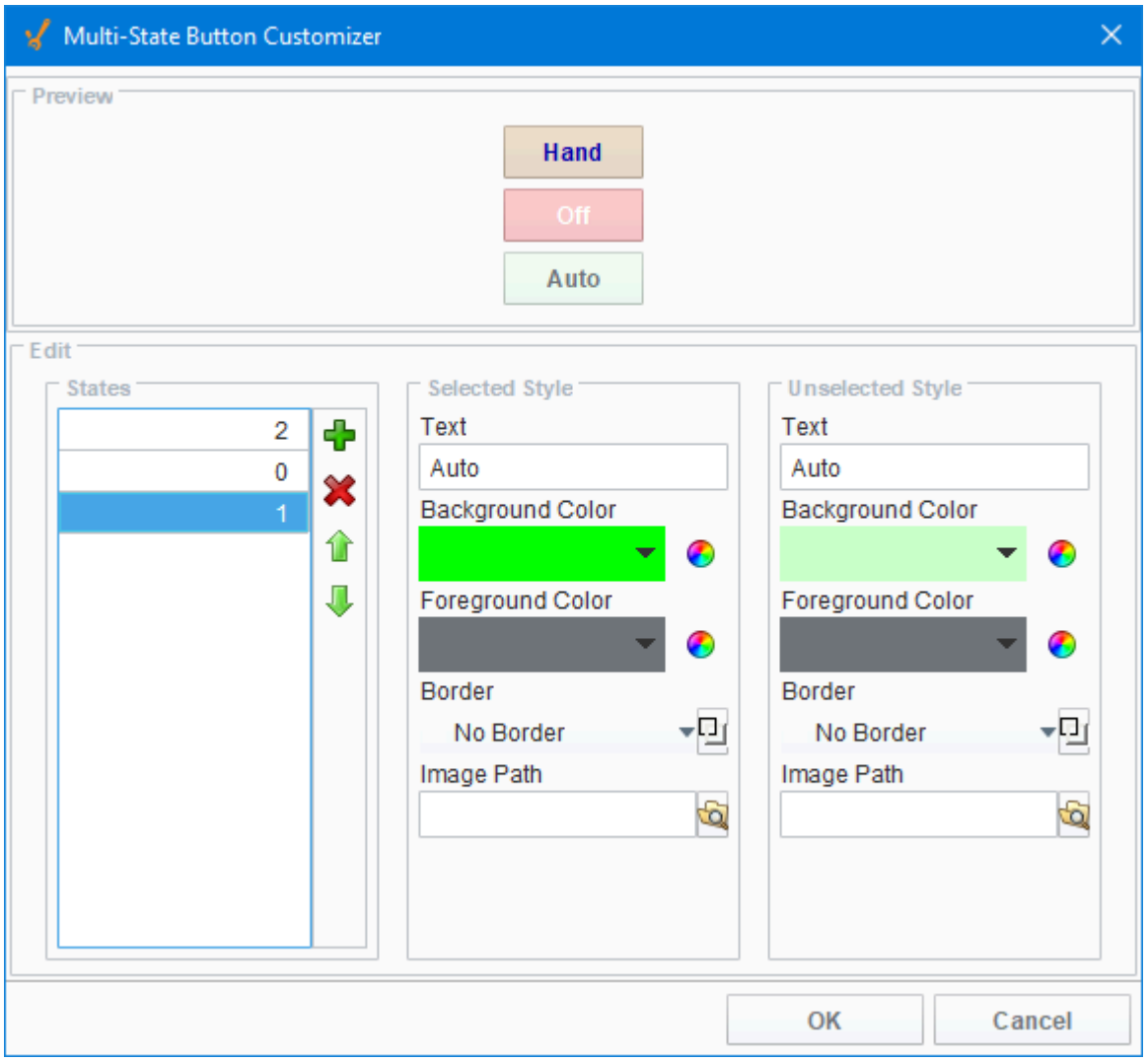
Event Handlers

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

Customizers

The Multi-State Button Customizer consists of four panels:

- **Preview**: Displays a preview image of the component
- **States**: Displays a list of possible states. Each state corresponds to one button. You may add, remove, and re-order states.
- **Selected Style**: Displays configurable style properties for the selected button. The button will use this style when its state is active.
- **Unselected Style**: Displays configurable style properties for the selected button. The button will use this style when its state is inactive.

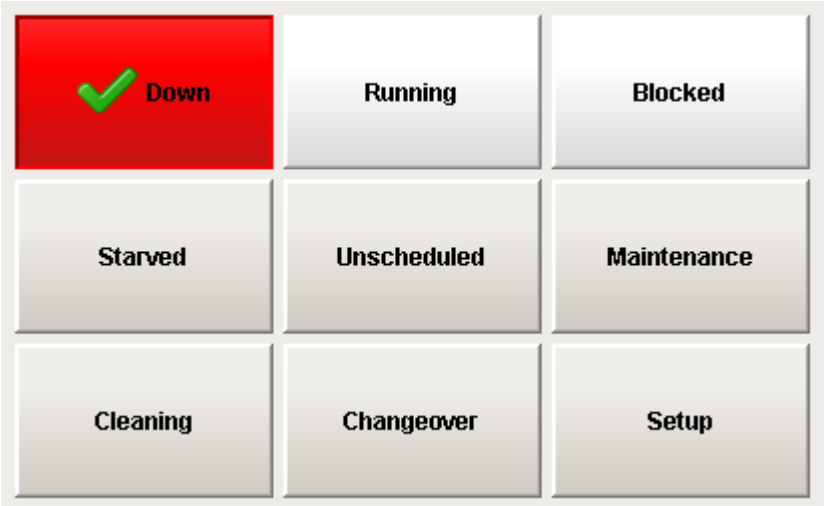


Multi-State Button Customizer Property Descriptions

Property	Description
Text	Text displayed on the button.
Background Color	Color of the button
Foreground Color	Color of the text
Border	Type of border around the button
Image Path	Relative path name for an image on the button

- **Vision Component Customizers**
- **Style Customizer**

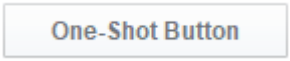
Example



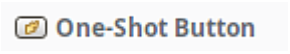
Property Name	Value
Display Style	Grid
Styles	

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

Vision - One-Shot Button



Component Palette Icon:



The One-Shot button is great for telling a PLC to do something. It simply writes a value, and then waits for it to be reset by the PLC before it is available again. This is only applicable when the PLC is programmed to reset the value after reading it. If your PLC expects the HMI to reset the bit, use the Momentary Button.

> **NOTE**
>
> This component is considered safer than the momentary button, because it receives positive feedback from the PLC that the signal was received, avoiding the timing dangers associated with a Momentary Button.

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

Properties

Property	Description	Property Type	Scripting	Category
Background Color	The background color of the button. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.buttonBG	Appearance
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Confirm Text	The message to ask the user if confirmation is turned on.	String	.confirmText	Behavior

Property	Description	Property Type	Scripting	Category
Confirm?	If true, a confirmation box will be shown.	boolean	.confirm	Behavior
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Disable While Writing	If true, the button will be disabled while it is writing.	boolean	.disableWhileWriting	Behavior
Disabled Image Path	The relative path of the image to be displayed when this component is not enabled.	String	.disabledPath	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Fill Area?	Controls whether or not this button's internal area is filled	boolean	.contentAreaFilled	Appearance

Property	Description	Property Type	Scripting	Category
Focusable	If a button is not focusable, you will not be able to interact with it with the keyboard. This means you can't "tab" over to it.	boolean	.focusable	Behavior
Font	Font of text on this component	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.foreground	Appearance
Horizontal Alignment	The horizontal alignment of the button's contents (text and/or image).	int	.horizontalAlignment	Layout
Horizontal Text Position	The horizontal position of the button's text relative to its image.	int	.horizontalTextPosition	Layout

Property	Description	Property Type	Scripting	Category
Icon-Text Spacing	The space (in pixels) between the icon (if any) and the text (if any).	int	.iconTextGap	Appearance
Idle Text	The text of the button while its value is not being written.	String	.normalText	Behavior
Image Path	The relative path of the image.	String	.path	Appearance
Margin	The space between a button's text and its borders.	Insets	.margin	Layout
Mnemonic	A single letter that will activate the button using 'ALT-*mnemonic*'.	String	.mnemonicChar	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag	QualityCode	.quality	Data

Property	Description	Property Type	Scripting	Category
	bindings on this component.			
Rollover	If true, the button may indicate that the mouse is hovering over it.	boolean	.rolloverEnabled	Behavior
Set Value	The value to set the control value to when the button is pushed.	int	.setValue	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Value	The current value. Should be bound bi-directionally to a tag.	int	.value	Data
Vertical Alignment	The vertical alignment of the button's contents (text and/or image).	int	.verticalAlignment	Layout
Vertical Text Position	The vertical position of the button's text relative to its image.	int	.verticalTextPosition	Layout

Property	Description	Property Type	Scripting	Category
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Writing Text	The text of the button while its value is being written.	String	.writePendingText	Behavior

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated
Opaque	If true, button will be opaque. Default is false.	boolean	.opaque	Common

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

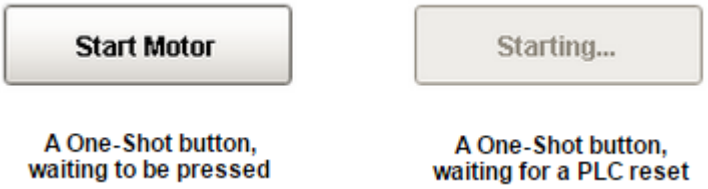
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the **Component Events** page

Customizers

- Vision Component Customizers
- Style Customizer

Example

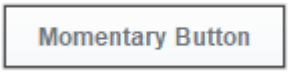
One Shot Button



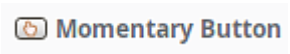
A One-Shot button, waiting to be pressed

A One-Shot button, waiting for a PLC reset

Vision - Momentary Button



Component Palette Icon:



Momentary buttons are used to set a value for either a fixed amount of time, or however long the button remains held down, whichever is longer. Once the button is released, or the minimum time expires, the value is reset.

The momentary button uses its Control Value property to affect the underlying data. Typically, this property uses a bidirectional tag binding to an OPC tag. When pressed, it will write its On Value to the Control Value property. When released, it will either write Off Value to the Control Value immediately, or wait until On Time has elapsed (since the pressed event).

The button's Indicator Value, which is typically bound to the same OPC tag as Control Value, is used to draw an "active" indication border around the button. This gives the operator positive feedback that the value has written successfully. It also lets an operator at one terminal know if an operator at a different terminal is using the button currently.

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

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

Properties

Property	Description	Property Type	Scripting	Category
Background Color	The background color of the button. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.buttonBG	Appearance
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Control Value	Bind this to the tag that you want to control. (Typically, this is bound to the same location as **Indicator Value**).	int	.controlValue	Data

Property	Description	Property Type	Scripting	Category
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Disabled Image Path	The relative path of the image to be displayed when this component is not enabled.	String	.disabledPath	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Fill Area?	Controls whether or not this button's internal area is filled.	boolean	.contentAreaFilled	Appearance
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel,	Color	.foreground	Appearance

Property	Description	Property Type	Scripting	Category
	chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.			
Horizontal Alignment	The horizontal alignment of the button's contents (text and/or image).	int	.horizontalAlignment	Layout
Horizontal Text Position	The horizontal position of the button's text relative to its image.	int	.horizontalTextPosition	Layout
Icon-Text Spacing	The space (in pixels) between the icon (if any) and the text (if any).	int	.iconTextGap	Appearance
Image Path	The relative path of the image.	String	.path	Appearance
Indicator Value	Bind this to the tag that indicates the current state of the control value. (Typically, this is bound to the same location as **Control Value**).	int	.indicatorValue	Data

Property	Description	Property Type	Scripting	Category
Indicator Width	The width of the indication border that shows whether or not the indicator value is currently set.	int	.indicatorWidth	Appearance
Max Hold Time	The maximum amount of time to keep the control value at the "On Value". When set to 0, this property is ignored.	int	.maxOnTime	Behavior
Min Hold Time	The minimum amount of time to keep the control value at the "On Value".	int	.onTime	Behavior
Mnemonic	A single letter that will activate the button using `ALT-mnemonic`.	String	.mnemonicChar	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common

Property	Description	Property Type	Scripting	Category
Off Color	The color of the indicator border when the indicator value is off. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.offColor	Appearance
Off Value	The value that will be written to the Control Value on mouse-up.	int	.offValue	Behavior
On Color	The color of the indicator border when the indicator value is on. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.onColor	Appearance
On Value	The value that will be written to the Control Value on mouse-down.	int	.onValue	Behavior

Property	Description	Property Type	Scripting	Category
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Rollover?	If true, the button may indicate that the mouse is hovering over it.	boolean	.rolloverEnabled	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Text	Text of this component.	String	.text	Appearance
Vertical Alignment	The vertical alignment of the button's contents (text and/or image).	int	.verticalAlignment	Layout
Vertical Text Position	The vertical position of the button's text relative to its image.	int	.verticalTextPosition	Layout
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Examples

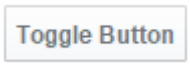
Momentary Button Activation States



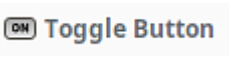
Momentary Button waiting to be pressed

Activated Momentary Button

Vision - Toggle Button



Component Palette Icon:



The Toggle button represents a bit: on (selected) or off (not selected). Visually the button looks down or depressed when it is selected, and up when it is not selected. Logically, this component is very similar to the Check Box component.

> **NOTE**
>
> For implementing a controls screen, the **2 State Toggle** is usually more appropriate than this component.

Properties

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

Property	Description	Property Type	Scripting	Category
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Fill Area?	Controls whether or not this button's internal area is filled.	boolean	.contentAreaFilled	Appearance
Focusable	If a button is not focusable, you will not be able to interact with it with the keyboard. This means you can't "tab" over to it.	boolean	.focusable	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.foreground	Appearance
Image Path	The relative path of the image.	String	.path	Appearance
Label	Text displayed on this button.	String	.text	Appearance

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

Property	Description	Property Type	Scripting	Category
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

Customizers

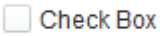
- **Vision Component Customizers**

- Style Customizer

Examples



Vision - Check Box



Component Palette Icon:



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

Properties

Property	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), and Field 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

Property	Description	Property Type	Scripting	Category
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Fill Background	If true, the label's background color will be drawn. If false, it will have a transparent background.	boolean	.fillBackground	Appearance
Focusable	If a button is not focusable, you will not be able to interact with it with the keyboard. This means you can't "tab" over to it.	boolean	.focusable	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.foreground	Appearance
Horizontal Alignment	The horizontal alignment of the button's contents (text and/or image).	int	.horizontalAlignment	Layout

Property	Description	Property Type	Scripting	Category
Margin	The internal margin that provides padding for the contents of this button.	Insets	.margin	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Rollover	If true, the button may indicate that the mouse is hovering over it.	boolean	.rolloverEnabled	Behavior
Selected	The current state of the checkbox.	boolean	.selected	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Text	The text displayed on the checkbox.	String	.text	Appearance

Property	Description	Property Type	Scripting	Category
Vertical Alignment	The vertical alignment of the button's contents (text and/or image).	int	.verticalAlignment	Layout
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

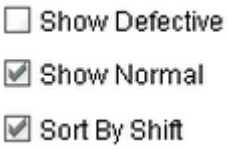
Event Handlers

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

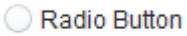
Customizers

- Vision Component Customizers
- Style Customizer

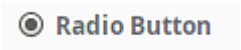
Examples



Vision - Radio Button



Component Palette Icon:



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

Properties

Property	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), and Field 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

Property	Description	Property Type	Scripting	Category
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Fill Background	If true, the label's background color will be drawn. If false, it will have a transparent background.	boolean	.fillBackground	Appearance
Focusable	If a button is not focusable, you will not be able to interact with it with the keyboard. This means you can't "tab" over to it.	boolean	.focusable	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.foreground	Appearance
Horizontal Alignment	The horizontal alignment of the button's contents (text and/or image).	int	.horizontalAlignment	Layout

Property	Description	Property Type	Scripting	Category
Margin	The internal margin that provides padding for the contents of this button.	Insets	.margin	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Rollover	If true, the button may indicate that the mouse is hovering over it.	boolean	.rolloverEnabled	Behavior
Selected	The current state of the Radio Button.	boolean	.selected	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Text	Text of this component.	String	.text	Appearance

Property	Description	Property Type	Scripting	Category
Vertical Alignment	The vertical alignment of the button's contents (text and/or image).	int	.verticalAlignment	Layout
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

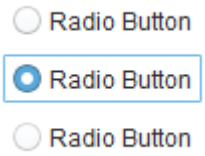
Event Handlers

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

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Example

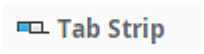


Radio Buttons inside a container will be exclusive therefore selecting one Radio Button will de-select the other Radio Buttons.

Vision - Tab Strip



Component Palette Icon:



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

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

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

Properties

Property	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), and Field 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

Property	Description	Property Type	Scripting	Category
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Intertab Space	The amount of space between each tab.	int	.interTabSpace	Appearance
Name	The name of this component.	String	.name	Common
Navigation Mode	Navigation mode. Disabled does nothing when a tab is pressed. Swap to window swaps to the window whose name corresponds to the name of the selected tab, provided that window exists.	int	.navigationMode	Behavior
Orientation	Orientation of the tab strip.	int	.orientation	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Renderer	The renderer to use when rendering tabs.	int	.renderer	Appearance
Rounding Radius	Rounding radius for the tab corners.	int	.roundingRadius	Appearance

Property	Description	Property Type	Scripting	Category
Selected Tab	Name of the selected tab. This is also the name of the window that, if it exists, will be swapped to when this tab is pressed.	String	.selectedTab	Appearance
Separator Color	Color of the line drawn across the bottom and around each tab. See **Color Selector**.	Color	.separatorColor	Appearance
Separator Thickness	Thickness of the line drawn across the bottom and around each tab.	float	.separatorThickness	Appearance
Size Mode	The sizing mode tabs use when deciding their size. Automatic means every tab is the same fixed size. Individual lets each tab decide its own size based on the size of its text.	int	.sizeMode	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Tab Data	Tab data to be displayed.	Dataset	.tabData	Data

Property	Description	Property Type	Scripting	Category
Text Alignment	The alignment of the tab text.	int	.textAlignment	Appearance
Text Offset	Padding on the left or right side of tab's text, depending on alignment.	int		Appearance
Text Padding	Padding on each side of the text inside a tab.	int	.textPadding	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

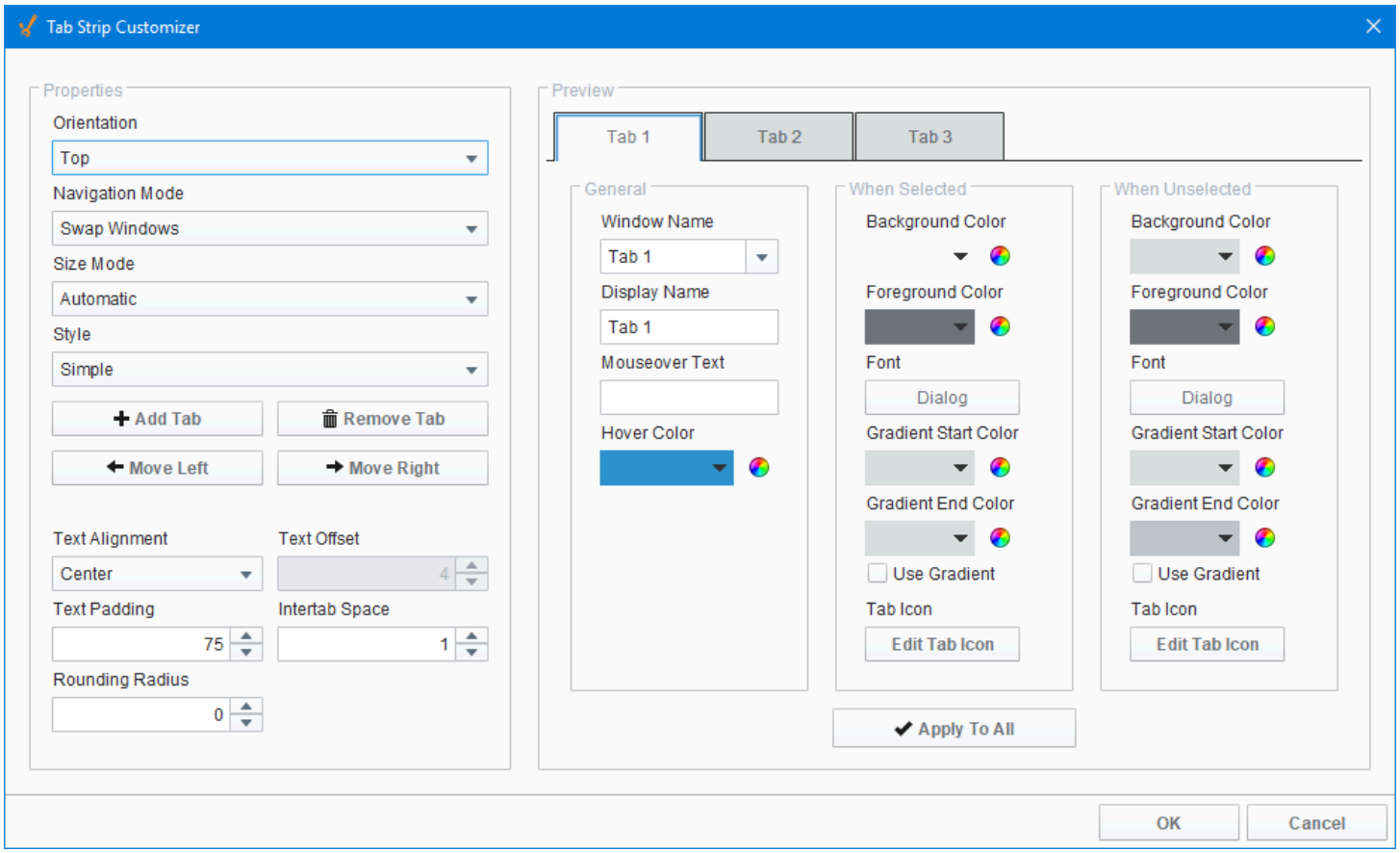
Event Handlers

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

Customizers

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

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



Properties	Description
Orientation	Orientation of the Tab Strip on a window: **Top**, **Left**, **Bottom** and **Right**. For example, use the Top orientation to place the Tab Strip component at the top of your window.
Navigation Mode	Two Navigation modes: • **Swap Windows** - the Tab Strip automatically calls **system.vision.swapTo()** to perform a window swap from the current window to another window when a tab is pressed. Swap Windows is the default mode. • **Disabled** - the Tab Strip only sets the **Selected Tab** property when pressed. You can set the component's behavior using property bindings or by responding to the **propertyChange** scripting event.
Size Mode	Two Size modes: • **Individual** - all the tabs are the same size. • **Automatic** - all the tabs are sized to fit the text.
Style	Three style options to change the appearance of the individual tabs: **Simple**, **Fancy**, and **Folder**.
Add Tab	Adds a new tab next to the selected tab.
Remove Tab	Removes a selected tab.
Move Up / Move Down	Depends on the current Orientation selection. Moves the selected tab **Up** or **Down** in the tab strip when using the **Left** or **Right orientation**.
Move Left / Move Right	Depends on the current Orientation selection. Moves the selected tab either **Left** or **Right** in the tab strip when using the **Top / Bottom orientation**.
Text Alignment	Inserts text in the Center, Left, or Right inside a tab.
Text Offset	Specifies how many pixels to move text to the left or right within a tab.
Text Padding	Specifies the number of pixels around the text in the tab.

Properties	Description
Intertab Space	Specifies the number of pixels between tabs.
Rounding Radius	Specifies the number of pixels to round the corners of the tab depending on the tab orientation.

General Section

Properties	Description
Window Name	Pathname of the window location
Display name	The name to display on the tab.
Mouseover Text	The text to display in the tooltip which pops up when mousing over a tab.
Hover Color	The color to display in the tootip which pops up when mousing over a tab.

When Selected/When Unselected

Properties	Description
Background Color	The background color of the tab.
Foreground Color	The foreground color is the color of the text.
Font	Select the font type, font size, and style.
Gradient Start Color	Select a start color to begin the gradient. Gradients are not valid for the Fancy style, and are shown as being grayed out. Select Simple or Folder style to use the gradient feature.
Gradient End Color	Select an end color to end the gradient. Gradients are not valid for the Fancy style, and are shown as being grayed out. Select Simple or Folder style to use the gradient feature.
Use Gradient	Select Use Gradient checkboxes to use gradient features. Uncheck the Use Gradient checkboxes to disable the gradient feature.
Tab Icon	Select an image from the Image Browser to insert on a tab.
Apply to All	The button applies all of the currently shown settings (except Window Name and Display Name) to all of the tabs. **Note**: This does not save your changes.

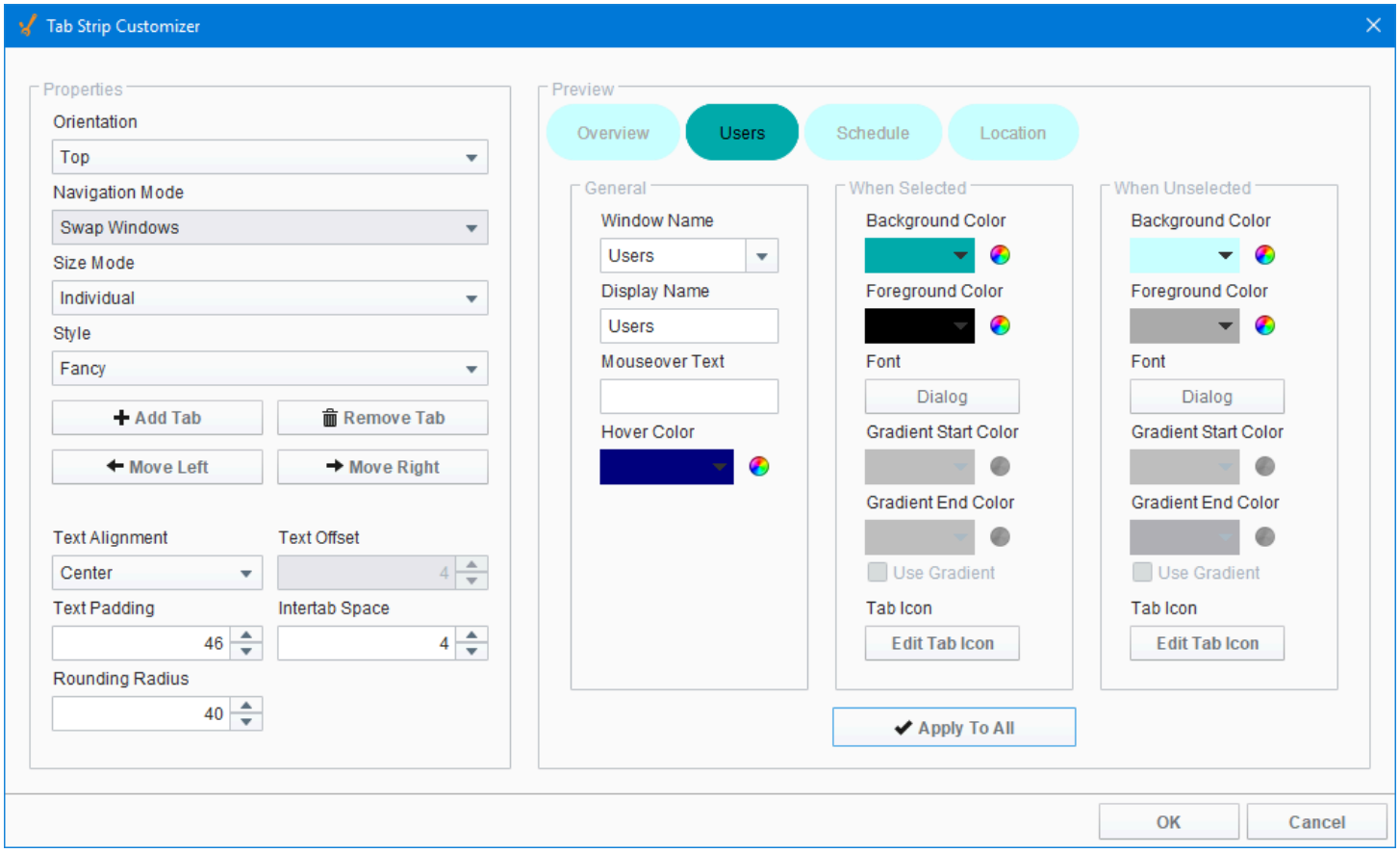
Additional Customizers

- **Vision Component Customizers**
- **Style Customizer**

Example

Horizontal Tabs

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



For a more in-depth example on how to use the Tab Strip for navigation, see the **Navigation - Tab Strip** page.

Vision - Calendar Palette

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

Calendar

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

Popup Calendar

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

Date Range

Day View

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

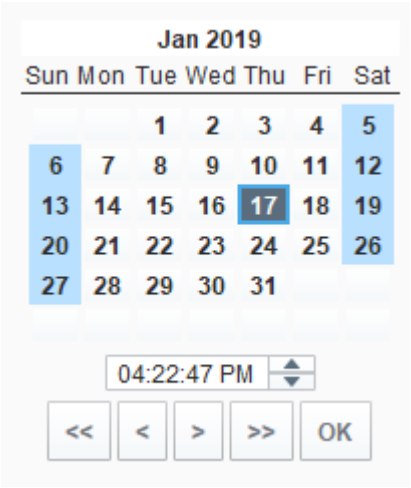
Week View

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

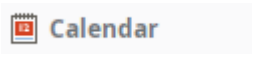
Month View

This component displays events for an entire month.

Vision - Calendar



Component Palette Icon:



Description

Displays a calendar and time input directly embedded in your window. Most commonly used by including one of the two date properties (immediate or latched) from the calendar in a dynamic **SQL Query Binding in Vision**.

Properties

Property	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), and Field 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,	int	.cursorCode	Common

Property	Description	Property Type	Scripting	Category
	Wait, Hand, Move, SW Resize, or SE Resize.			
Date (immediate)	The date as it is selected right now.	Date	.date	Data
Date (latched)	The date the last time "OK" was pressed.	Date	.latchedDate	Data
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See **Color Selector**.	Color	.foreground	Appearance
Format String	The date formatting pattern used to format the string versions of the dates.	String	.format	Behavior
Formatted Date	The date property, as formatted by the format string.	String	.formattedDate	Data

Property	Description	Property Type	Scripting	Category
Formatted Latched Date	The latched date property, as formatted by the format string.	String	.formattedLatchedDate	Data
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Opaque	If false, backgrounds are not drawn. If true, backgrounds are drawn.	boolean	.opaque	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Selected Border	The border for the selected day indicator.	Border	.selectedBorder	Appearance
Show OK Button	Turn this off if you don't want to show the OK button. The latched date and the immediate	boolean	.showOkButton	Behavior

Property	Description	Property Type	Scripting	Category
	date will be equivalent.			
Show Time	Turn this off if you don't want to show the time panel.	boolean	.showTime	Behavior
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Time Display Format	The format for displaying time in the panel.	int	.timeDisplayFormat	Behavior
Time Style	Select how this calendar should treat the time portion of the date.	int	.timeStyle	Behavior
Title Background	The background of the title bar. See **Color Selector**.	Color	.titleBackground	Appearance
Today Background	Background color for the today indicator. See **Color Selector**.	Color	.todayBackground	Appearance
Today Foreground	Foreground color for the today indicator. See **Color Selector**.	Color	.todayForeground	Appearance

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

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

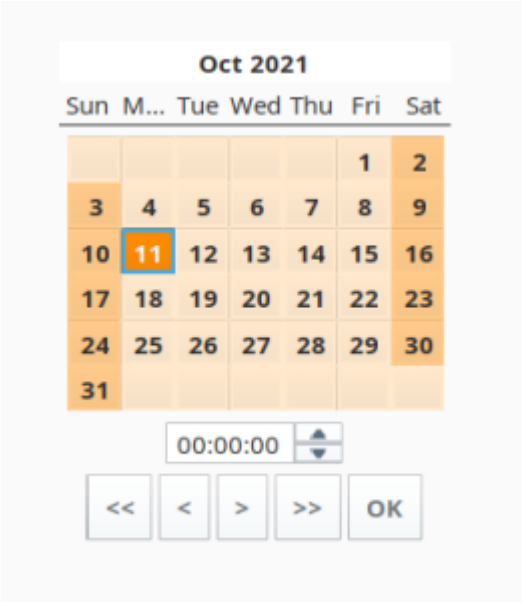
Event Handlers

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

Customizers

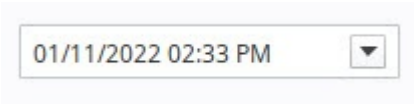
- **Vision Component Customizers**
- **Style Customizer**

Examples

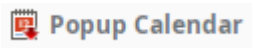


Property Name	Value
Background Color	255,232,204
Today Background	255,140,0
Weekend Background	255,202,138

Vision - Popup Calendar



Component Palette Icon:



Description

The popup calendar displays a drop-down menu on a window. When clicked, a Calendar pops up to allow users to select a date and/or time.

Properties

Property	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), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Calendar Background	The background color for the popup calendar. See **Color Selector**.	Color	.calendarBackground	Appearance
Cursor	The mouse cursor to use when hovering over this	int	.cursorCode	Common

Property	Description	Property Type	Scripting	Category
	component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.			
Date	The date that this component represents.	Date	.date	Data
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. SSee **Color Selector**.	Color	.foreground	Appearance
Format String	The date formatting pattern used to display this date.	String	.format	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common

Property	Description	Property Type	Scripting	Category
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Selected Border	The border for the selected day indicator.	Border	.selectedBorder	Appearance
Show Navigation	Turn this off if you don't want to show the year and month navigation buttons.	boolean	.showNavigation	Appearance
Show OK Button	Turn this off if you don't want to show the OK button. The latched date and the immediate date will be equivalent.	boolean	.showOkButton	Behavior
Show Time	Turn this off if you don't want to show the time panel.	boolean	.showTime	Behavior
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Text	The displayed text of the date (depends on the format string).	String	.text	Data

Property	Description	Property Type	Scripting	Category
Time Display Format	The format for displaying time in the panel.	int	.timeDisplayFormat	Behavior
Time Style	Select how this calendar should treat the time portion of the date.	int	.timeStyle	Behavior
Title Background	The background of the title bar.	Color	.titleBackground	Appearance
Today Background	Background color for the today indicator. See **Color Selector**.	Color	.todayBackground	Appearance
Today Foreground	Foreground color for the today indicator. See **Color Selector**.	Color	.todayForeground	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Weekend Background	Background color for the weekend indicators. See **Color Selector**.	Color	.weekendBackground	Appearance
Weekend Foreground	Foreground color for the weekend indicators. See **Color Selector**.	Color	.weekendForeground	Appearance

Deprecated Properties

Property	Description	Property Type	Scripting	Category
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.

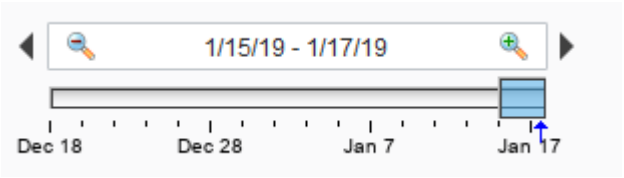
Event Handlers

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

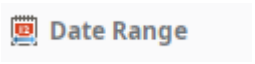
Customizers

- **Vision Component Customizers**
- **Style Customizer**

Vision - Date Range



Component Palette Icon:



Description

The date range component provides an intuitive, drag-and-drop way to select a contiguous range of time. The user is shown a timeline and can drag or stretch the selection box around on the timeline. The selected range is always a whole number of units, where the unit is determined by the current zoom level.

> **NOTE**
>
> The **Start/End** dates and **Outer Start/End** dates will be ignored when the window opens unless the Startup Mode property is set to "None."

Data Density Histogram

As an advanced optional feature, the date range can display a data density histogram inside the timeline. This is useful for historical data with gaps in it, so that the end user isn't hunting for data. (Tip: this is also great for demos, to make it easy to find historical data in a database that isn't being continuously updated).

To use this feature, bind the Data Density dataset to a query that returns just the timestamps of the target table. These timestamps will be used to fill in the histogram behind the timeline. You can use the Outer Range Start Date and Outer Range End Date properties in your query to limit the overall return size for the query.

> **NOTE**
>
> Timestamps must be ordered by date (ascending) to display correctly.

Properties

Property	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), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Box Fill	The fill color for the selection box.	Color	.boxFill	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default,	int	.cursorCode	Common

Property	Description	Property Type	Scripting	Category
	Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.			
Data Density	A dataset that is used to calculate a histogram of data density.	Dataset	.densityData	Data
Date Style	The style to display dates in. For international support.	int	.dateStyle	Appearance
Editor Background	The background color of the textual date range editor portion of this component.	Color	.editorBackground	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
End Date	The ending date of the currently selected range.	Date	.endDate	Data
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance
High Density Color	The color used to indicate high data	Color	.highDensityColor	Appearance

Property	Description	Property Type	Scripting	Category
	density. See **Color Selector**.			
Max Selection	The maximum size of the selected date range.	String	.maxSelectionSize	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Opaque	If false, backgrounds are not drawn. If true, backgrounds are drawn.	boolean	.opaque	Common
Outer Range End	The ending date of the available outer range.	Date	.outerRangeEndDate	Data
Outer Range Start	The starting date of the available outer range.	Date	.outerRangeStartDate	Data
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Selection Highlight	The focus highlight color for	Color	.selectionHighlight	Appearance

Property	Description	Property Type	Scripting	Category
	the selection box. See **Color Selector**.			
Start Date	The starting date of the currently selected range.	Date	.startDate	Data
Startup Mode	Controls whether or not this date range automatically assigns itself a starting range based on the current time	int	.startupMode	Behavior
Startup Range	If startup mode is Automatic, this will be the starting range of time available for selection.	String	.startupRange	Behavior
Startup Selection	If startup mode is Automatic, this will be the starting selected range.	String	.startupSelection	Behavior
Styles	Contains the component's styles.	Dataset	.styles	Appearance

Property	Description	Property Type	Scripting	Category
Tick Density	This is multiplied by the width to determine the current ideal tick unit.	float	.tickDensity	Behavior
Time Style	The style to display times of day. For international support.	int	.timeStyle	Appearance
Today Color	The color of the "Today Arrow" indicator. See **Color Selector**.	Color	.todayIndicatorColor	Appearance
Track Margin	The amount of room on either side of the slider track. May need adjusting of default font is changed.	int	.trackMargin	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Scripting Functions

See the **Date Range Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Example

Code Snippet

```
//A Query binding on another component on the same window might look like this:

SELECT Column1, Column2, Column3
FROM MyTable WHERE
  t_stamp >= "{Root Container.Date Range.startDate}" AND
  t_stamp <= "{Root Container.Date Range.endDate}"
```

Vision - Date Range Scripting Functions

This page details the various component and extension functions available for **Vision's Date Range component**.

Component Functions

.setRange(start, end)

- Description
  - Sets the selected range. The outer range will move if needed. Note: the start and end times are determined based on the zoom level and may not move (or may move farther than intended) if the component is zoomed out too far for the amount of change attempted. IE: When days are showing, moving the start time 5 minutes forward will not effect the start, and moving the end time 5 minutes forward will add one day.
- Parameters
  - Date **start** - The starting date for the new selection.
  - Date **end** - The ending date for the new selection.
- Return
  - Nothing
- Scope
  - Client

Code Snippet

```
```

```
# This example moves the existing Start Date and End Date
# of a Date Range component ahead 8 hours
from java.util import Calendar

# Get the current start and end
dateRangeComponent = event.source.parent.getComponent('Date Range')
startDate = dateRangeComponent.startDate
endDate = dateRangeComponent.endDate

# Calculate the new start and end dates
cal = Calendar.getInstance();
cal.setTime(startDate);
cal.add(Calendar.HOUR, -8);
newStart = cal.getTime();

cal.setTime(endDate);
cal.add(Calendar.HOUR, -8);
newEnd = cal.getTime();

# Set the new range for the component. The outer range will
# automatically expand if needed.
dateRangeComponent.setRange(newStart, newEnd)
```

.setOuterRange(start, end)

- Description
  - Sets the outer range. The selected range will move if needed. Note: the start and end times are determined based on the zoom level and may not move (or may move farther than intended) if the component is zoomed out too far for the amount of change attempted. IE: When days are showing, moving the start time 5 minutes forward will not effect the start, and moving the end time 5 minutes forward will add one day.
- Parameters
  - Date **start** - The starting date for the new outer range.
  - Date **end** - The ending date for the new outer range.
- Return
  - Nothing
- Scope
  - Client

Code Snippet

```
# This example moves the existing Outer Date Range
# of a Date Range component back two days
from java.util import Calendar

# Get the current start and end of the outer range
dateRangeComponent = event.source.parent.getComponent('Date Range')
startDate = dateRangeComponent.outerRangeStartDate
endDate = dateRangeComponent.outerRangeEndDate

# Calculate the new start and end dates for the outer range
cal = Calendar.getInstance();
cal.setTime(startDate);
cal.add(Calendar.DAY_OF_MONTH, 2);
newStart = cal.getTime();

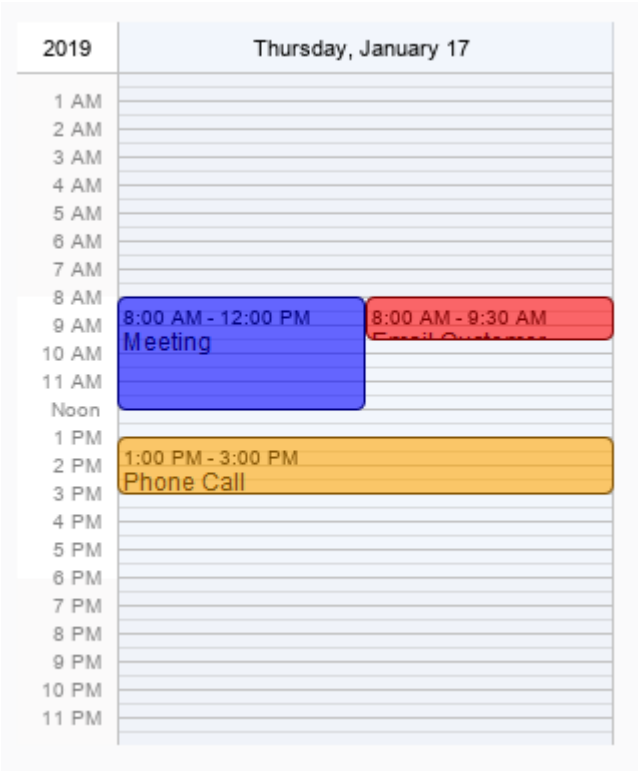
cal.setTime(endDate);
cal.add(Calendar.DAY_OF_MONTH, 2);
newEnd = cal.getTime();

# Set the new outer range for the component.
dateRangeComponent.setOuterRange(newStart, newEnd)
```

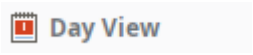
Extension Functions

This component does not have extension functions associated with it.

Vision - Day View



Component Palette Icon:



Description

This component displays a timeline for a single day, similar to what you might find in a personal planner/organizer. By filling in the Calendar Events dataset property, the component will display events that occur on this day. Each event can have custom text and a custom display color associated with it.

Properties

Property	Description	Property Type	Scripting	Category
24 Hour Format	Whether or not to show 24 hour or 12 hour format.	boolean	.twentyFourHour	Appearance
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Calendar Background Color	The color of the calendar's background. Can be chosen from	Color	.calendarBackground	Appearance

Property	Description	Property Type	Scripting	Category
	color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.			
Calendar events	Contains the calendar events.	Dataset	.events	Data
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Day	Set the calendar's day.	int	.day	Data

Property	Description	Property Type	Scripting	Category
Day Outline Color	The color of the day's outline. See **Color Selector**.	Color	.boxOutline	Appearance
Event Font	The font for all calendar events.	Font	.eventFont	Appearance
Grid marks	Set the amount of grid lines.	int	.gridMarks	Appearance
Hour Font	The font for the hour of the day.	Font	.hourFont	Appearance
Hour Foreground Color	The foreground color for hours in a day. See **Color Selector**.	Color	.hourForeground	Appearance
Hover Background Color	The background color of the hovered time. See **Color Selector**.	Color	.hoverBackground	Appearance
Hovered Event	The calendar's	int	.hoveredEvent	Data

Property	Description	Property Type	Scripting	Category
	hovered event.			
Hovered Time	The calendar's hovered time.	String	.hoveredTime	Data
Month	Set the calendar's month.	int	.month	Data
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Non-Working Hours Background Color	The background color for the non-working hours of the day. See **Color Selector**.	Color	.nonWorkingHourBackground	Appearance
Quality	The data quality code for any Tag bindings on	QualityCode	.quality	Data

Property	Description	Property Type	Scripting	Category
	this component.			
Selected Event	The calendar's selected event.	int	.selectedEvent	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Today's Background Color	The color of the today's background. See **Color Selector**.	Color	.todayBackground	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Week Day Background Color	The color of the week day's background. See **Color Selector**.	Color	.weekDaysBackground	Appearance
Week Day Font	The font of the week day's text.	Font	.weekdayFont	Appearance
Week Day Foreground Color	The color of the week day's text.	Color	.weekDaysForeground	Appearance

Property	Description	Property Type	Scripting	Category
	See **Color Selector**.			
Working End Hour	The end hour of a working day.	int	.workingEndHour	Appearance
Working Start Hour	The start hour of a working day.	int	.workingStartHour	Appearance
Year	Set the calendar's year.	int	.year	Data
Zoom	Zooms into the specified zoom time-range.	boolean	.autoZoom	Appearance
Zoomed End Hour	The end hour zoomed in.	int	.autoZoomEndHour	Appearance
Zoomed Start Hour	The start hour zoomed in.	int	.autoZoomStartHour	Appearance

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

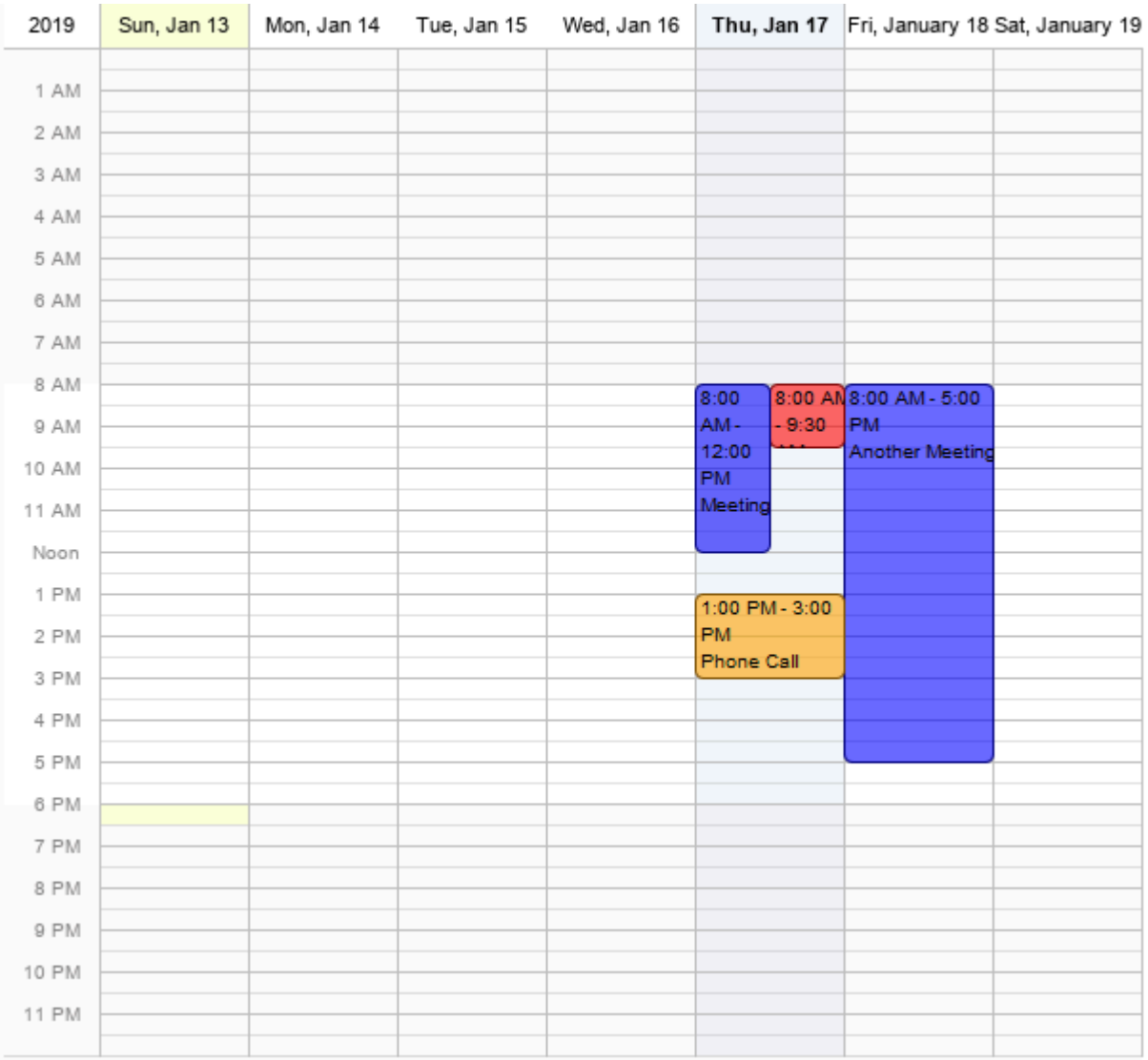
Event Handlers

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

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Vision - Week View



Component Palette Icon:



Description

Displays a full week's worth of events on a calendar. Configuration is achieved by populating the Calendar Events dataset. See the **Vision - Day View component** for details.

Properties

Property	Description	Property Type	Scripting	Category
24 Hour Format	Whether or not to show 24 hour or 12 hour format.	boolean	.twentyFourHour	Appearance
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Calendar Background Color	The color of the calendar's background. Can be chosen from	Color	.calendarBackground	Appearance

Property	Description	Property Type	Scripting	Category
	color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.			
Calendar events	Contains the calendar events.	Dataset	.events	Data
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Day	Set the calendar's day.	int	.day	Data

Property	Description	Property Type	Scripting	Category
Day Outline Color	The color of the day's outline. See **Color Selector**.	Color	.boxOutline	Appearance
Event Font	The font for all calendar events.	Font	.eventFont	Appearance
Grid marks	Set the amount of grid lines.	int	.gridMarks	Appearance
Hour Font	The font for the hour of the day.	Font	.hourFont	Appearance
Hour Foreground Color	The foreground color for hours in a day. See **Color Selector**.	Color	.hourForeground	Appearance
Hover Background Color	The background color of the hovered day and time. See **Color Selector**.	Color	.hoverBackground	Appearance
Hovered Day	The calendar's	String	.hoveredDay	Data

Property	Description	Property Type	Scripting	Category
	hovered day.			
Hovered Event	The calendar's hovered event.	int	.hoveredEvent	Data
Hovered Time	The calendar's hovered time.	String	.hoveredTime	Data
Month	Set the calendar's month.	int	.month	Data
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common

Property	Description	Property Type	Scripting	Category
Non-Working Hours Background Color	The background color for the non-working hours of the day. See **Color Selector**.	Color	.nonWorkingHourBackground	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Selected Background Color	The color of the selected day's background. See **Color Selector**.	Color	.selectedBackground	Appearance
Selected Day	The calendar's selected day.	String	.selectedDay	Data
Selected Event	The calendar's selected event.	int	.selectedEvent	Data
Show Event Time?	Whether or not to show the event time.	boolean	.showEventTime	Appearance

Property	Description	Property Type	Scripting	Category
Show Weekend?	Whether or not to show Saturday and Sunday.	boolean	.showWeekend	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Today's Background Color	The color of the today's background. See **Color Selector**.	Color	.todayBackground	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Week Day Background Color	The color of the week day's background. See **Color Selector**.	Color	.weekDaysBackground	Appearance
Week Day Font	The font of the week day's text.	Font	.weekdayFont	Appearance
Week Day Foreground Color	The color of the week day's text. See **Color Selector**.	Color	.weekDaysForeground	Appearance

Property	Description	Property Type	Scripting	Category
Working End Hour	The end hour of a working day.	int	.workingEndHour	Appearance
Working Start Hour	The start hour of a working day.	int	.workingStartHour	Appearance
Year	Set the calendar's year.	int	.year	Data
Zoom	Zooms into the specified zoom time range.	boolean	.autoZoom	Appearance
Zoomed End Hour	The end hour zoomed in.	int	.autoZoomEndHour	Appearance
Zoomed Start Hour	The start hour zoomed in.	int	.autoZoomStartHour	Appearance

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

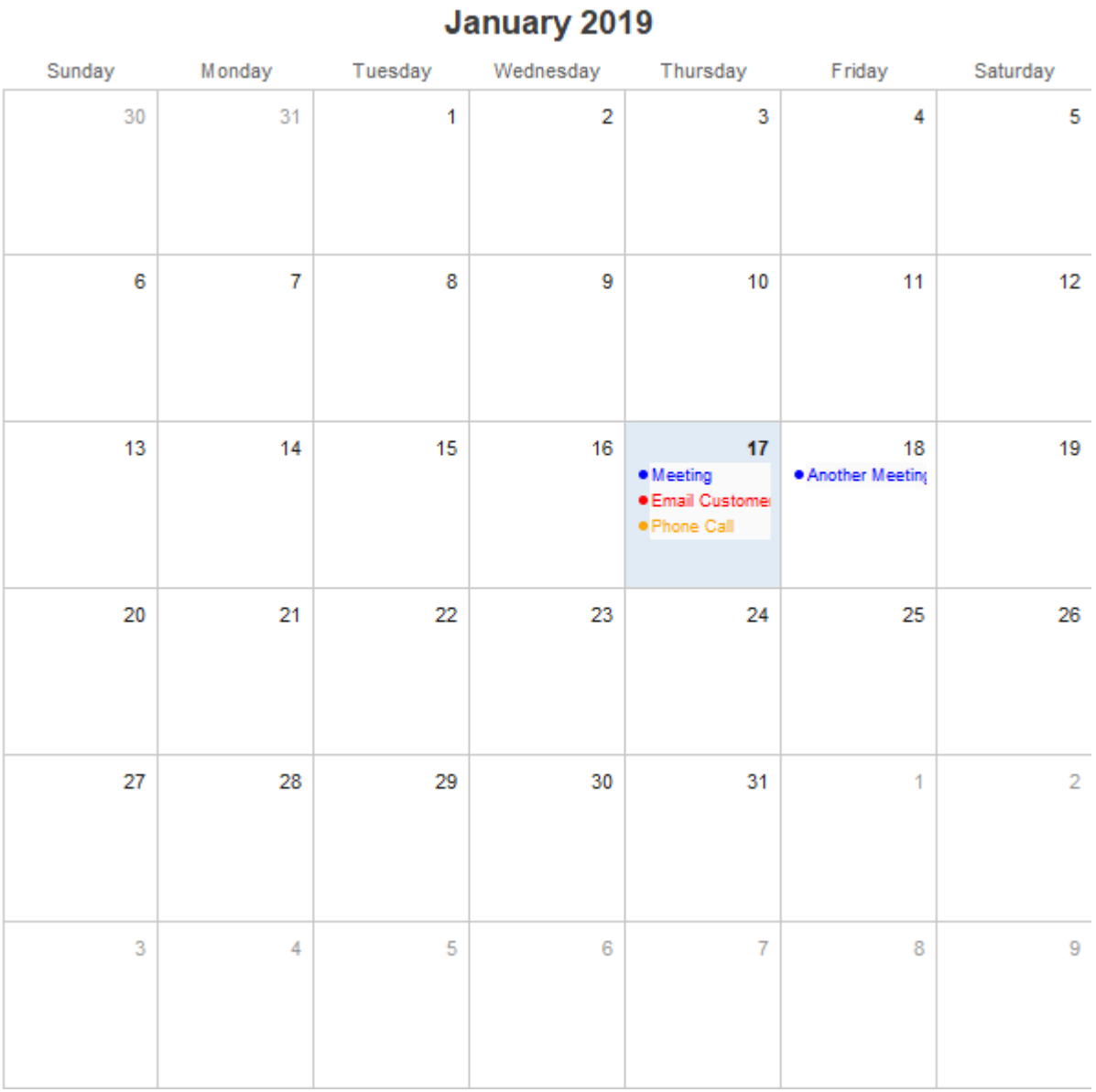
Event Handlers

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

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Vision - Month View



Component Palette Icon:



Description

This component displays events for an entire month. By filling in the Calendar Events dataset property, the component will display events that occur for each day of the month. Each event can have custom text and a custom display color associated with it.

Properties

Property	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), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Calendar Background Color	The color of the calendar's background. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL	Color	.calendarBackground	Appearance

Property	Description	Property Type	Scripting	Category
	value. See **Color Selector**.			
Calendar events	Contains the calendar events.	Dataset	.events	Data
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Day Font	The font for the number representing the day of the month.	Font	.dayFont	Appearance
Day Foreground Color	The foreground color for days in this month. See **Color Selector**.	Color	.dayOfMonthForeground	Appearance

Property	Description	Property Type	Scripting	Category
Day Other Foreground Color	The foreground color for days not in this month. See **Color Selector**.	Color	.dayOfMonthOtherForeground	Appearance
Day Outline Color	The color of the day's outline. See **Color Selector**.	Color	.boxOutline	Appearance
Event Background Color	The background color of the selected event. See **Color Selector**.	Color	.itemSelBackground	Appearance
Event Display Mode	Affects how events are displayed. **Standard Mode**: Displays each event **Highlight Mode**: Highlights each day that contains events using the event	int	.displayMode	Appearance

Property	Description	Property Type	Scripting	Category
	highlight background color.			
Event Font	The font for all calendar events.	Font	.eventFont	Appearance
Event Highlight Background	The background color of a day with events. Used only in highlight mode.	Color	.highlightBackground	Appearance
Header Background Color	The color of the header's background. See **Color Selector**.	Color	.monthHeaderBackground	Appearance
Header Font	The font of the header's text.	Font	.headerFont	Appearance
Header Foreground Color	The color of the header's text. See **Color Selector**.	Color	.monthHeaderForeground	Appearance
Hover Background Color	The background color of the hovered day. See	Color	.hoverBackground	Appearance

Property	Description	Property Type	Scripting	Category
	Color Selector.			
Hovered Day	The calendar's hovered day.	String	.hoveredDay	Data
Month	Set the calendar's month.	int	.month	Data
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data

Property	Description	Property Type	Scripting	Category
Selected Background Color	The color of the selected day's background. See **Color Selector**.	Color	.selectedBackground	Appearance
Selected Day	The calendar's selected day.	String	.selectedDay	Data
Selected Event	The calendar's selected event.	int	.selectedEvent	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Today's Background Color	The color of the today's background. See **Color Selector**.	Color	.todayBackground	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Week Day Background Color	The color of the week day's background.	Color	.weekDaysBackground	Appearance

Property	Description	Property Type	Scripting	Category
	See **Color Selector**.			
Week Day Font	The font of the week day's text.	Font	.weekdayFont	Appearance
Week Day Foreground Color	The color of the week day's text. See **Color Selector**.	Color	.weekDaysForeground	Appearance
Year	Set the calendar's year.	int	.year	Data

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Vision - Charts Palette

The following components give you various charts for displaying data.

Easy Chart

Chart

Sparkline Chart

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

Bar Chart

Radar Chart

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

Status Chart

Pie Chart

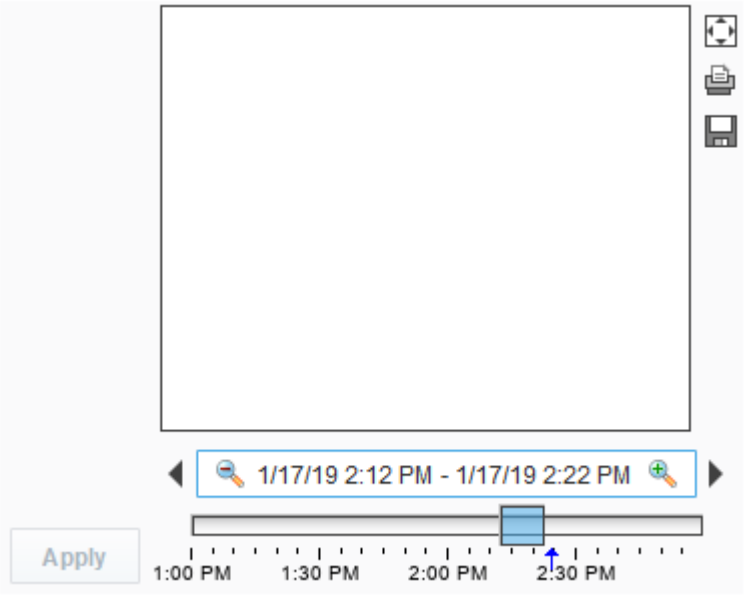
Box and Whisker Chart

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

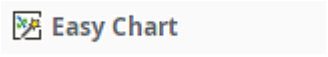
Equipment Schedule

Gantt Chart

Vision - Easy Chart



Component Palette Icon:



Description

This component is used to make runtime-configurable time-series charts. It is configured by defining a set of pens and axes. Pens can be many different styles, such as line, area, bar, and shape. This chart automatically creates controls for picking the time range and for hiding or displaying pens.

Features

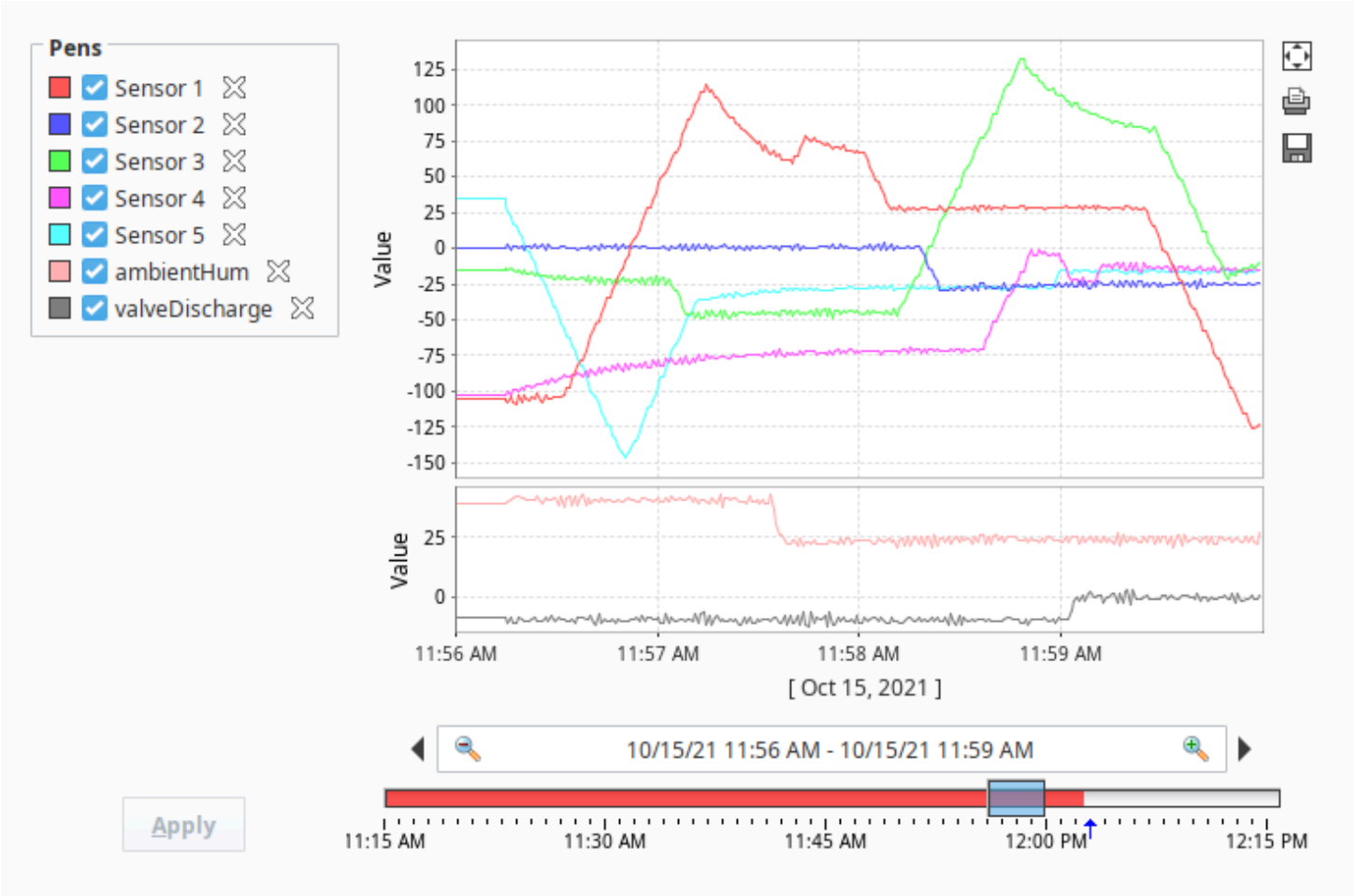
- Easy configuration
- User-selectable set of pens
- Automatic time-selection controls
- SQL Query and/or SQLTags Historian data sources
- Automatic SPC and calculated pen support
- Zoom, Pan, X-Trace modes
- Any number of Y-axes and subplots
- Realtime or Historical

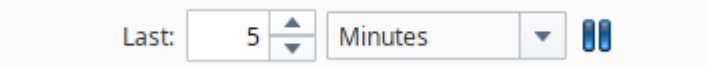
Pens

There are three kinds of pens in the Easy Chart:

1. Tag Historian Pens. These pens pull their data from the **Historian** system.
2. Database Pens. These pens will automatically create SQL SELECT queries to pull data from a database table. Typically, this is a table that is the target of a **Historical Transaction Group**.
3. Calculated Pens: These pens display a calculated dataset based off another pen, such as a moving average or Statistical Process Control (SPC) function such as the Upper Control Limit (UCL).

Interface Elements



Element	Description
Pens	After dragging a tag onto the Easy Chart, a Pen corresponding to the tag's value will appear in the Pens panel. Pens can be selected and deselected using the checkboxes. Clicking the X next to a pen will remove it from the Easy Chart.
Date Range	In **Historical** mode, the user is shown a **Vision - Date Range** component to pick the range of data to fetch and display. The initial values of this component are set through properties on the chart. In historical mode, the chart does not poll. In **Realtime** mode, the user is instead given the ability to select an amount of time before the current timestamp:  The chart will poll at a rate according to the Poll Rate parameter. In **Manual** mode, no user input controls are displayed. The chart will use the values of its Start Date and End Date parameters to govern what data is displayed. The chart will poll at a rate according to the Poll Rate parameter.
Maximize 	Clicking this button will maximize the chart to fill the area of the component.
Print 	Clicking this button will print the chart.
Save to Excel 	Clicking this button will save the chart data as an Excel spreadsheet.

NOTE

You can bring up a context menu for this component when right-clicking on it either in the Designer's Preview Mode or in a Vision Client. See the **Charting - Right Click Menu** page for more details.

Properties

Name	Description	Property Type	Scripting
3D X Offset	The offset to use in the x direction for the '3D Line' pen style.	int	.xOffset3D
3D Y Offset	The offset to use in the y direction for the '3D Line' pen style.	int	.yOffset3D
Allow Color Changes	If true, pen colors can be set to different values.	boolean	.allowColorChange
Allow Tag History Interpolation	If enabled and the query mode is not raw, the data will be interpolated for time spans with no data available.	boolean	tagHistoryAllowInte
Auto Apply	If true, user changes to pen visibility will occur immediately.	boolean	.autoApply
Auto Axis Positioning	If true, axes alternate automatically between left and right, rather than being placed explicitly.	boolean	.autoPositionAxes
Auto Color List	The list of colors to use if auto pen coloring is enabled.	Color[]	.autoColorList
Auto Pen Coloring	If true, pens are assigned different colors automatically.	boolean	.autoColorPens
Axes	This Dataset defines all axes that can be used by the pens.	Dataset	.axes
Axis Font	The font for axis labels.	Font	.axisLabelFont
Background Color	The background color of the component. See **Color Selector**.	Color	.background

Name	Description	Property Type	Scripting
Bar Margin	The margin to use for the 'Bar' pen style.	double	.barMargin
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Field Border, and Line Border. **Note:** The border is unaffected by rotation.	Border	.border
Box Fill	For historical-mode date range. The fill color for the selection box. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.boxFill
Button Size	The size of the utility button icons.	int	.utilityButtonSize
Bypass Tag History Cache	If true, tag history queries will not use the client history cache.	boolean	.tagHistoryBypass
Calculated Pens	This Dataset defines the calculated pens for the chart.	Dataset	.calcPens
Chart Border	The border for the chart itself.	Border	.chartBorder
Chart Mode	Affects the mode that the chart operates in; Manual Mode, Historical Mode, Realtime Mode.	int	.chartMode

<table>
<tr><th>Name</th><th>Description</th><th>Property Type</th><th>Scripting</th></tr>
<tr><td></td><td><table>
<tr><th>Integer Value</th><th>Corresponding Mode</th></tr>
<tr><td>0</td><td>Manual</td></tr>
<tr><td>1</td><td>Historical</td></tr>
<tr><td>2</td><td>Realtime</td></tr>
</table></td><td></td><td></td></tr>
<tr><td>Chart Title</td><td>Sets an optional title to be displayed above the chart.</td><td>String</td><td>.title</td></tr>
<tr><td>Cursor</td><td>The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.</td><td>int</td><td>.cursorCode</td></tr>
<tr><td>DB Pens</td><td>This Dataset defines all of the database pens for the chart.</td><td>Dataset</td><td>.pens</td></tr>
<tr><td>Date Editor Background</td><td>The background color for the date editor. See Color Selector.</td><td>Color</td><td>.editorBackground(</td></tr>
<tr><td>Date Editor Foreground</td><td>The foreground color for the date editor. See Color Selector.</td><td>Color</td><td>.editorForeground(</td></tr>
<tr><td>Date Range</td><td>Affects the position of the date range control.</td><td>int</td><td>.dateRangeLocatio</td></tr>
<tr><td>Date Range Border</td><td>The border for the date range control, if visible.</td><td>Border</td><td>.dateRangeBorder</td></tr>
<tr><td>Date Style</td><td>The style to display dates in. For international support.</td><td>int</td><td>.dateStyle</td></tr>
</table>

Name	Description	Property Type	Scripting
Digital Gap	The size of the gap to use between digital pens.	double	.digitalGap
Empty Group Name	The group name to use for pens that are not in a pen group.	String	.emptyGroupName
End Date	For manual-mode. The end date to use for selecting pen data	Date	.endDate
Font	Font of text on this component.	Font	.font
Foreground Color	The foreground color of the component. See **Color Selector**.	Color	.foreground
Gap Threshold	The relative threshold to use for determining continuity breaks for the 'Discontinuous Line' pen style.	double	.gapThreshold
Gridline Color	The color of the gridlines. See **Color Selector**.	Color	.gridlineColor
Gridline Dash Pattern	Enter a string of comma-delimited numbers which indicate the stroke pattern for a dashed line. For instance, "3,5" means three pixels on, five pixels off.	String	.gridlineDashPatter
Gridline Width	The width (thickness) of the gridlines.	float	.gridlineWidth
Group Pens	If true, pens will be grouped by their group name.	boolean	.penGrouping
High Density Color	For historical-mode date range. The color used to indicate high data density. See **Color Selector**.	Color	.highDensityColor
Horiz Gap	The horizontal spacing to use for the pen checkboxes.	int	.hGap

Name	Description	Property Type	Scripting
Ignore Bad Quality Data	If true, causes the system to ignore any bad quality data.	boolean	tagHistoryIgnoreBa
Invert Time Axis	If true, the time axis values will increase from the right to left or from top to bottom depending on the Plot Orientation.	boolean	.invertTimeAxis
Legend	Where the legend should appear, if any.	int	.legend
Max Selection	For historical-mode date range. The maximum size of the selected date range.	String	.maxSelectionSize
Maximize Plot	If true, displays maximized plot.	boolean	.currentlyMaximize
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText
Name	The name of this component.	String	.name
Outer Range End	For historical-mode date range. The end date for the outer range. Used in cases when an explicit point in time should be used to define the outer range of the chart. Competes with Startup Range.	Date	.outerRangeEnd
Outer Range Start	For historical-mode date range. The start date for the outer range. Used in cases when an explicit point in time should be used to define the outer range of the chart. Competes with Startup Range.	Date	.outerRangeStart
Pen Control Border	The border for the pen control panel, if visible.	Border	.penBorder
Pen Control Mode	The style in which the pen control panel alters the chart configuration. In heavyweight mode, unchecked pens are	int	.penControlMode

Name	Description	Property Type	Scripting
	not queried, so checking and unchecking pens refreshes the chart. In lightweight mode, all pens are constantly queried, so checking and unchecking pens is quick.		
Pen Control?	Controls whether or not end-users can turn on and off pens.	boolean	.allowPenManipula
Plot Background	The background color for all plots, unless they override it. See **Color Selector**.	Color	.plotBackground
Plot Orientation	The plot orientation for all plots.	int	.plotOrientation
Plot Outline	The color to use for the plot outline. See **Color Selector**.	Color	.plotOutlineColor
Poll Rate	The rate (in milliseconds) at which this chart's queries poll. Historical charts don't use this property.	int	.pollRate
Properties Loading	The number of properties currently being loaded. (Read only. Usable in bindings and scripting.)	int	.propertiesLoading
Realtime Text	For realtime-mode date range. The text to display on the realtime date control.	String	.rtLabel
Selected X Value	The selected domain axis value for X-Trace and Mark modes. (Read only. Usable in bindings and scripting.)	String	.selectedXValue
Selection Highlight	For historical-mode date range. The focus highlight color for the selection box. See **Color Selector**.	Color	.selectionHighlight
Show Density	For historical-mode date range. If true, a data density histogram will be shown in the date range.	boolean	.showHistogram

Name	Description	Property Type	Scripting
	Note: Not available for tags stored by the **Core Historian** or **Internal Historian (Legacy)**, or when **Enable Stale Data Detection** is disabled.		
Show Loading	If true, an animated indicator will be shown when data is loading.	boolean	.showLoading
Show Maximize Button?	If true, a small maximize button will be displayed next to the chart.	boolean	.showMaximize
Show Popup?	If true, a popup menu will be shown on right-click that allows the user to change mode, print, save, etc.	boolean	.showPopup
Show Print Button?	If true, a small print button will be displayed next to the chart..	boolean	.showPrint
Show Save Button?	If true, a small save button will be displayed next to the chart.	boolean	.showSave
Show Tooltips?	If true, tooltips showing point values will be displayed on the chart.	boolean	.tooltips
Show Warnings	If true, warnings generated during chart configuration will be printed to the console.	boolean	.showWarnings
Sort Pens	If true, pens visibility checkboxes will be sorted.	boolean	.alphabetizePens
Start Date	For manual-mode. The start date to use for selecting pen data.	Date	.startDate
Startup Range	For historical-mode date range, this will be the starting range of time available for selection. Useful in cases where the chart should range a period of time on window	String	.startupRange

Name	Description	Property Type	Scripting
	open (i.e., show the last 8 hours). Competes with the Outer Range Start and Outer Range End properties.		
Startup Selection	For historical-mode date range, this value will be used for the starting selection range.	String	.startupSelection
Subplot Gap	The gap between subplots.	double	.subplotGap
Subplots	This Dataset defines all subplots' relative size and color.	Dataset	.subplots
Tag History Resolution	When Tag History Resolution Mode is set to "Fixed", this setting is used to specify the number of data points that should be returned by tag history queries. When a Tag History Provider has pre-processed partitions enabled, this setting can be used in conjunction with the chart's range to specify if the tag history data should use pre-processed partitions or not. For example, if the chart is displaying a range of 60 minutes, and resolution is set to 6, the resulting window size would be 10 minutes each. Since this is greater than the default **Pre-processed Window Size** of 60 seconds, the pre-processed table is queried. If the resulting window size is less than the Pre-processed Window Size, the raw data table will be queried. Setting this to -1 is equivalent to using the Raw Resolution Mode. Setting this to 0 is equivalent to using the Natural Resolution Mode.	int	.tagHistoryResoluti

Name	Description	Property Type	Scripting
Tag History Resolution Mode	The mode used for the number of requested points. Fixed will use the Tag History Resolution Size, Natural will return a value per tag group execution, Chart Width will be based on the actual width of the chart component, and Raw will be the raw data.	int	tagHistoryResolutic
Tag Pens	This Dataset defines all of the Tag History pens for the chart.	Dataset	.tagPens
Tick Density	For historical-mode date range. This is multiplied by the width to determine the current ideal tick unit.	float	.tickDensity
Tick Font	The font for tick labels.	Font	.axisTickLabelFont
Time Style	The style to display times of day. For international support.	int	.timeStyle
Title Font	The font for the optional chart title.	Font	.titleFont
Today Color	For historical-mode date range. The color of the "Today Arrow" indicator. See **Color Selector**.	Color	.todayIndicatorColo
Total Datapoints	The number of datapoints being displayed by the graph. (Read only. Usable in bindings and scripting.)	int	.datapoints
Track Margin	For historical-mode date range. The amount of room on either side of the slider track. May need adjusting of default font is changed.	int	.trackMargin
Unit	For realtime-mode date range. The selected unit of the realtime date control.	int	.unit

Name	Description	Property Type	Scripting
Unit Count	For realtime-mode date range. The number of units back to display.	int	.unitCount
Validate Scan Class Executions	Causes the tag history query to verify the scan class execution records, generating bad data for the time periods where the scanclasses did not execute.	boolean	tagHistoryValidateS
Vert Gap	The vertical spacing to use for the pen checkboxes.	int	.vGap
Visible	If disabled, the component will be hidden.	boolean	.visible
Where Clause	A snippet of where clause that will be applied to all database pens, like "TankNum = 2".	String	.globalWhereClaus
X Axis AutoRange?	If true, the X axis will automatically fit the range of available data, if false, it will display a fixed range based on the start date and end date.	boolean	.xAxisAutoRange
X Axis Label	The label shown on the X Axis (time axis).	String	.xAxisLabel
X Axis Margin	A margin for the upper and lower ends of the x axis, expressed as a percentage of the total range.	double	.xAxisMargin
X Axis Visible	Should the x-axis be displayed?	boolean	.xAxisVisible
X-Trace Large Number Format	The large decimal format for the x-trace value in the Easy Chart.	String	.xTraceLargeNumb
X-Trace Number Format Threshold	If the magnitude of the to-be-formatted value is below this threshold, then the X-Trace Small Number Format will be used.	double	.xTraceNumberFor

Name	Description	Property Type	Scripting
X-Trace Small Number Format	The small decimal format for the x-trace value in the Easy Chart.	String	.xTraceSmallNumb
X-Trace Track Mouse	If set enabled, and the chart is set to X-Trace mode, the X-Trace will auto track the mouse position while the cursor is over the component. This is particularly useful when displaying the Easy Chart on a touchscreen. `.xTraceTrackMouse` and `.XTraceTrackMouse` can also be used interchangeably to read/write to the property.	boolean	.XTraceTrackMous

Scripting

See the **Vision - Easy Chart Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

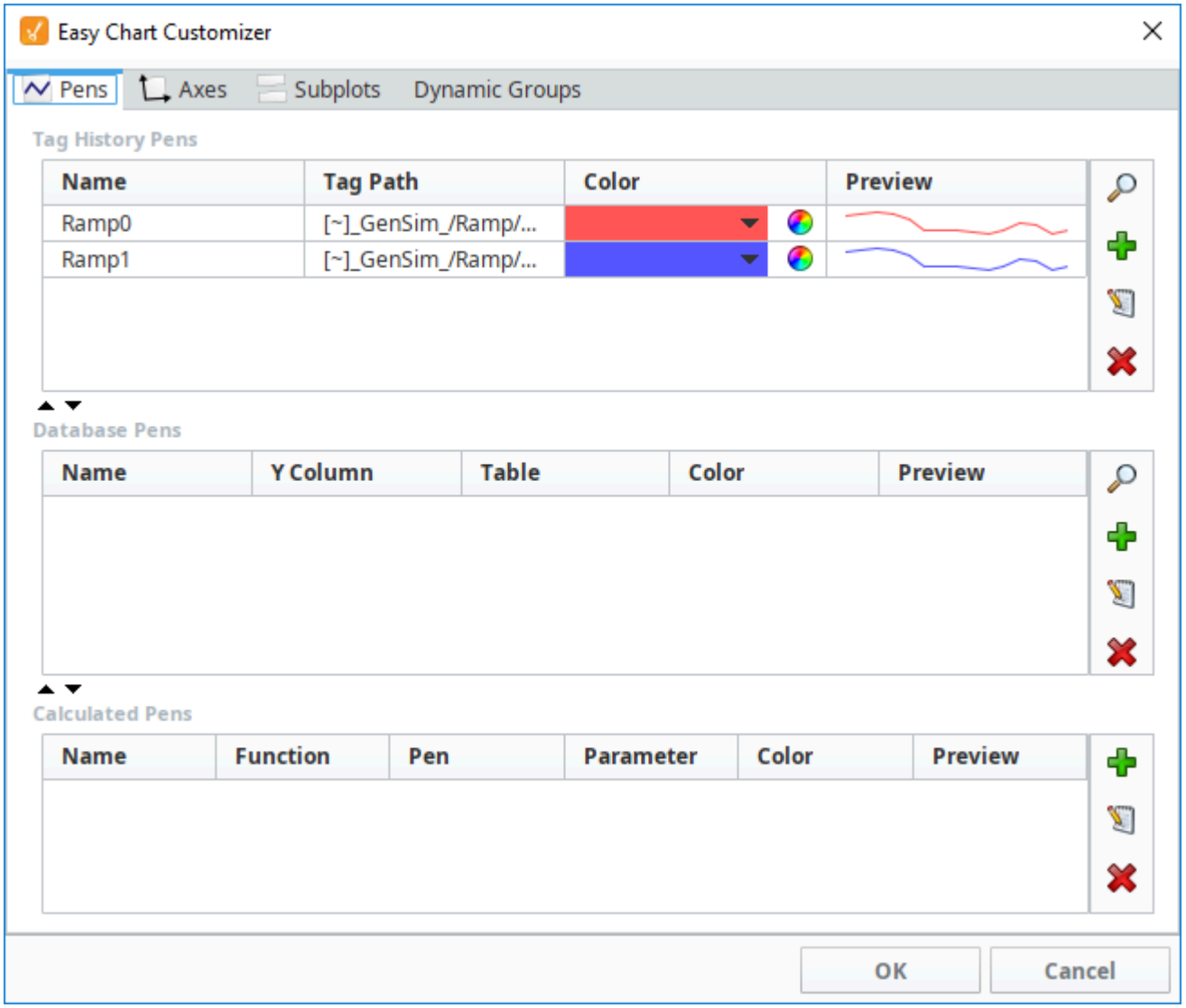
Customizers

Refer to the **Vision - Easy Chart Customizer** and the **Using the Vision Easy Chart** sections of the manual for examples and tutorials on how to use the Easy Chart Customizer. With the customizer, you can set up:

- **Axes**
- **Subplots**
- **Pen Groups**
- **Pen Display**
- **Offsets**
- **Calculated Pens**

- Ad-Hoc Charting
- Indirection

Vision - Easy Chart Customizer



Description

The **Easy Chart** component allows you to display the history of your Tags on a chart. When you drag and drop Tags onto an Easy Chart, it automatically trends the data for you. It has a special customizer that has some default settings to help you get started.

Customizers

The Easy Chart Customizer allows you to easily modify the chart to your own style. You can add pens and modify the contents of your pens, and create new axes, subplots, and pen groups. When you open the customizer, you'll notice four tabs at the top of the window: Pens, Axes, Subplots, and Dynamic Groups. Each have their own properties.

Shown below is each tab in the Easy Chart Customizer listing all its properties along with a brief description.

Pens Tab

The Pens tab is where you can add new pens, create custom names for your pens, and group pens. There are three types of pens, and each functions in a similar manner, but what makes them different is how their data is collected. Each pen type has a few unique properties and is listed at the bottom of the table.

- **Tag pens** - Pens are driven by the Tag history system. (Data from any historical provider can be used).
- **Database pens** - Pens that are driven by an SQL query. They can query for data in any connected SQL database.
- **Calculated pens** - Pens that derive their data from calculations performed on other pens.

For more information, refer to the following sections:

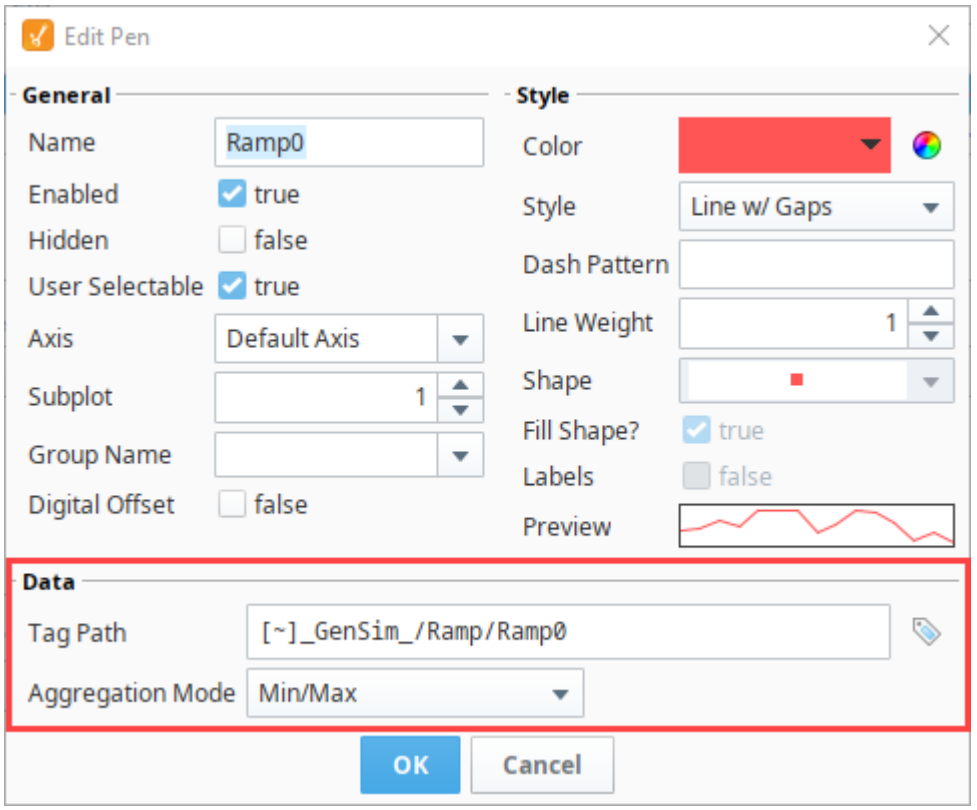
- **Easy Chart - Pen Names and Groups**
- **Easy Chart - Calculated Pens**
- **Using the Easy Chart**

Property	Description
	Add pen (Browse for Tags).
	Add a pen manually.
	Edit pen.
	Delete pen.

Edit Pen Panel

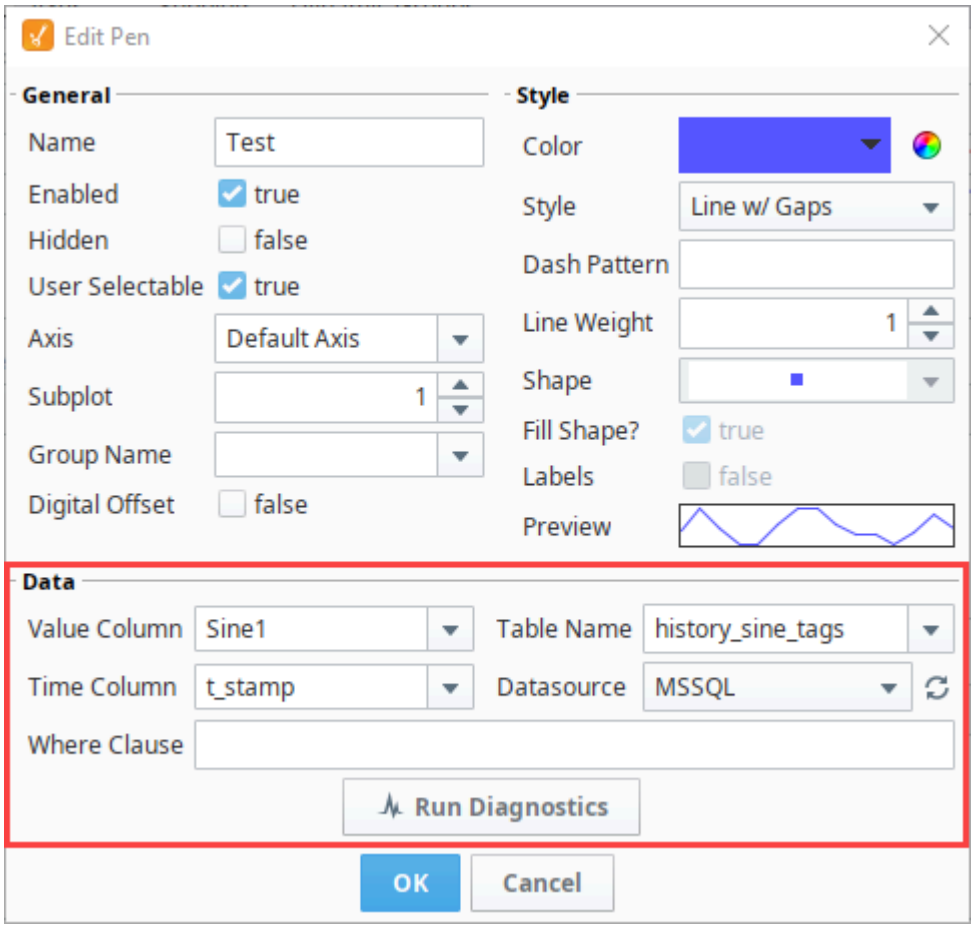
Property	Description
Name	The name of the pen is what the user will see in the legend and the pen panel.
Enabled	If false, this pen will not show up on the chart and the data will not be generated. The user will be able to enable it via the pen control panel.
Hidden	If true, the pen will not show up on the chart or the pen control panel. The data will be generated.
User Selectable	If false, the pen will show up on the chart, but not the pen control panel.
Axis	Select the Y axis this pen will use.
Subplot	Putting pens on separate subplots can increase chart clarity.
Group Name	The group name is used for logical grouping in the pen panel and for advanced dynamic grouping.
Digital Offset	If true, a small gap will be placed between this and other digital pens so they don't overlap each other.
Color	Pen color.
Style	The style of the pen determines how it looks in the chart.
Dash Pattern	Uses a dash pattern like "5,5" to specify 5 pixels on, 5 pixels off.
Line Weight	The thickness of the pen's line.
Shape	If the renderer style uses shapes, this will be the shape for each point.
Fill Shape	If true, the shape will be filled in rather than an outline.
Labels	If true, shows a label of the value above each bar.
Preview	Field where you can view the pen style.

Tag History Pens Properties



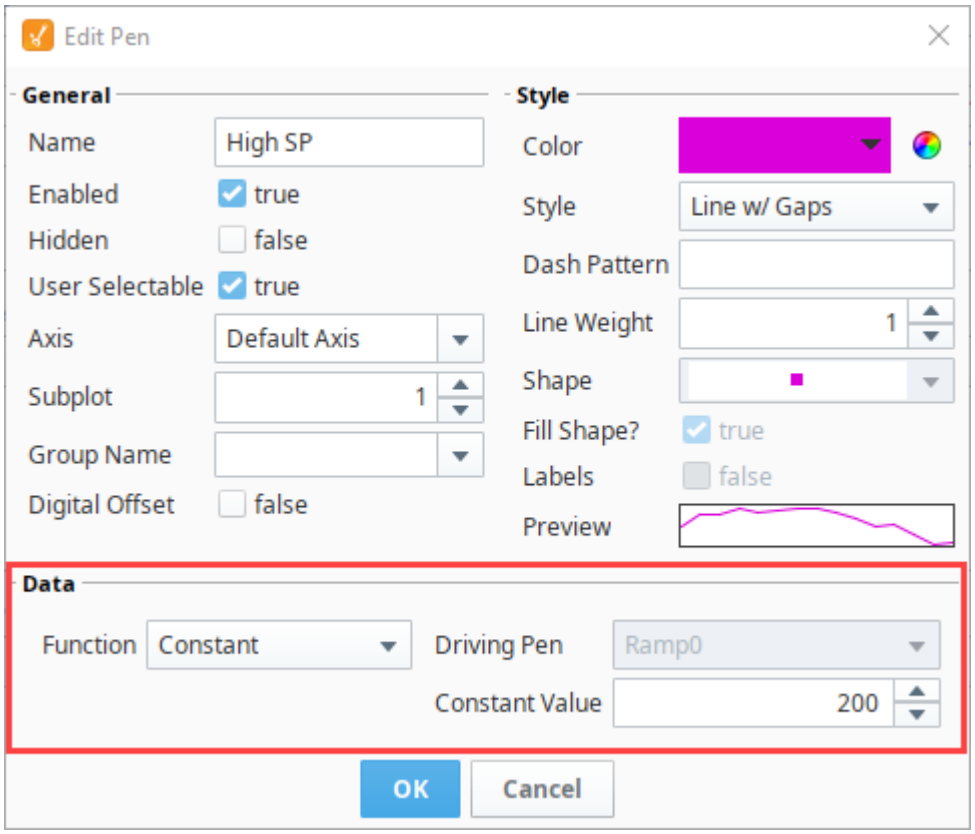
Property	Description
Tag Path	String-based path where the Tag is located.
Aggregation Mode	Type of calculation. See **Aggregation Mode** for more details.

Database Pens Properties



Property	Description
Volume Column	The name of the column for the pen's value (Y value).
Time Column	The name of the column for the pen's timestamp (X value).
Table Name	The name of the table where the pen will be found.
Datasource	The name of the data source to use for this pen (MySQL).
Where Clause	You can specify a snippet of WHERE clause here, like "TankNum = 16."
Run Diagnostics	Test this pen for data configuration for validity.

Calculated Pens Properties

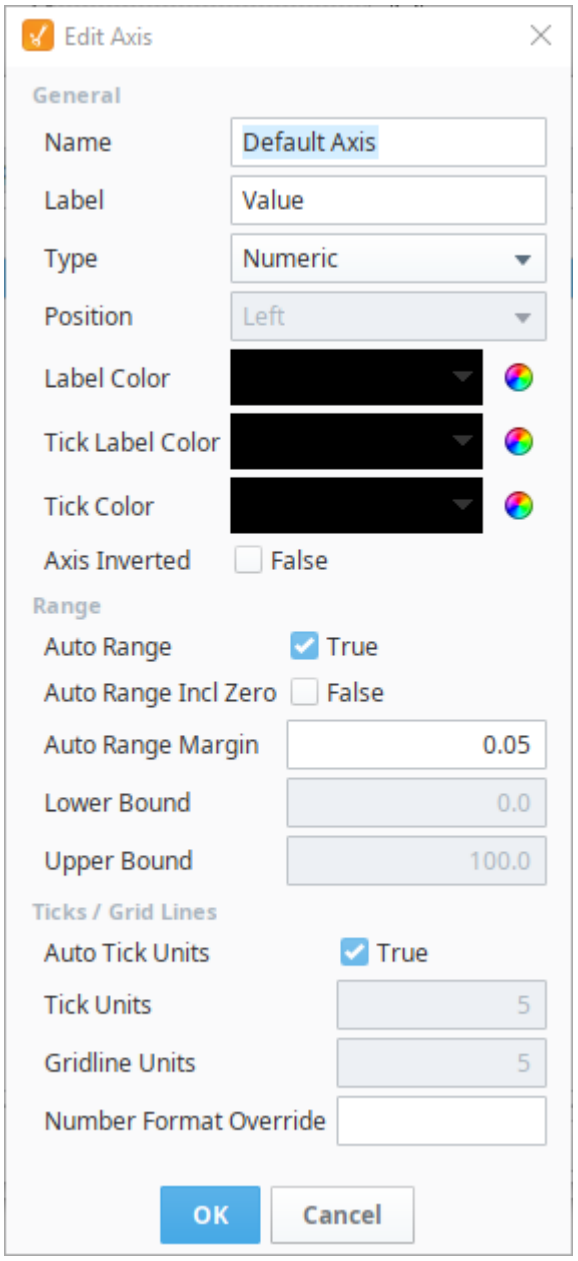


Property	Description
Function	Function is the type of calculation (i.e., Constant, UCL, UWL, Avg, LWL, LCL, MovingAvg, Multiply, Min, Max).
Driving Pen	Dedicated pen that will drive the value.
Parameter	Value which is the horizontal line drawn on the graph. The parameter type can be different for the Function used: • **Constant Value** - constant value of the pen. (Used with the Constant function). • **Window Size** - the size of the moving average window, specified as a multiplier of the chart's date range. It's the percentage of time that you're going to do the moving average on. (Used with MovingAvg function). • **Factor** - multiply by 'X' factor (Used with Multiply function). • **Secondary pen** - another pen added to the chart to show the sum and/or the difference. (Used with the Sum and Difference functions).

Axes Tab

The Axes tab is where you can configure multiple axes on the Easy Chart component.

Property	Description
Name	The name of the axis is what pens use to refer to it.
Label	The label will be displayed on the chart next to the axis.
Type	The type of axis determines the plotting behavior. (i.e., Numeric, Logarithmic, Symbol)
Position	The position of the axis, if automatic, axis positioning is turned off.
Label Color	Color of the label.
Tick Label Color	Color of the tick label.
Tick Color	Color of the tick mark.
Axis Inverted	If true, inverts the axis.
Auto Range	If true, the axis will automatically scale itself to the data, rather than display a fixed range.
Auto Range Incl Zero	If true, forces the auto range to include zero.
Auto Range Margin	The extra margin (as percent of the total range) for the top and bottom of an auto range axis.
Lower Bound	The lower bound of a non-auto-ranging axis.
Upper Bound	The upper bound of a non-auto-ranging axis.
Auto Tick Units	If true, the distance between the tick marks and the gridlines will be automatically calculated rather than a fixed number.
Tick Units	If false, this amount will be used as the distance between tick marks.
Gridline Units	If false, this amount will be used as the distance between gridlines.
Number Form Override	Specifies a number format pattern to use for tick labels. Leave blank for automatic number formatting.

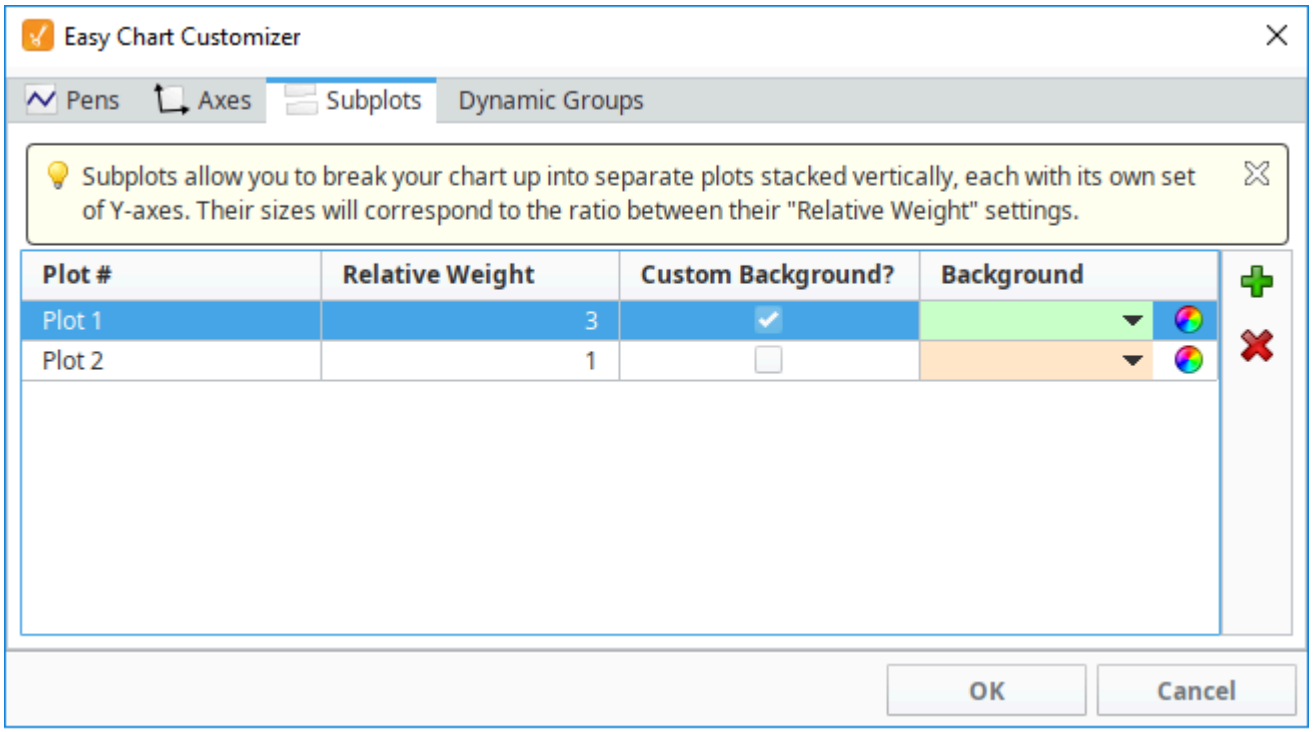


For more information, refer to the **Easy Chart - Axes** page.

Subplots Tab

The Subplots tab is where you can break up a chart's plot area into multiple distinct subplots that share the X axis, and also where you can add additional subplots.

Property	Description
Plot Number	Number of plots in a chart plot area.
Relative Weight	Ratio between all subplots. (If you have two subplots, and Plot 1's weight is 3 and Plot 2's weight is 1, then Plot 1 will be 3 times larger than Plot 2).
Custom Background	If false, the default background is white.
Background	Color of the plot area's background.

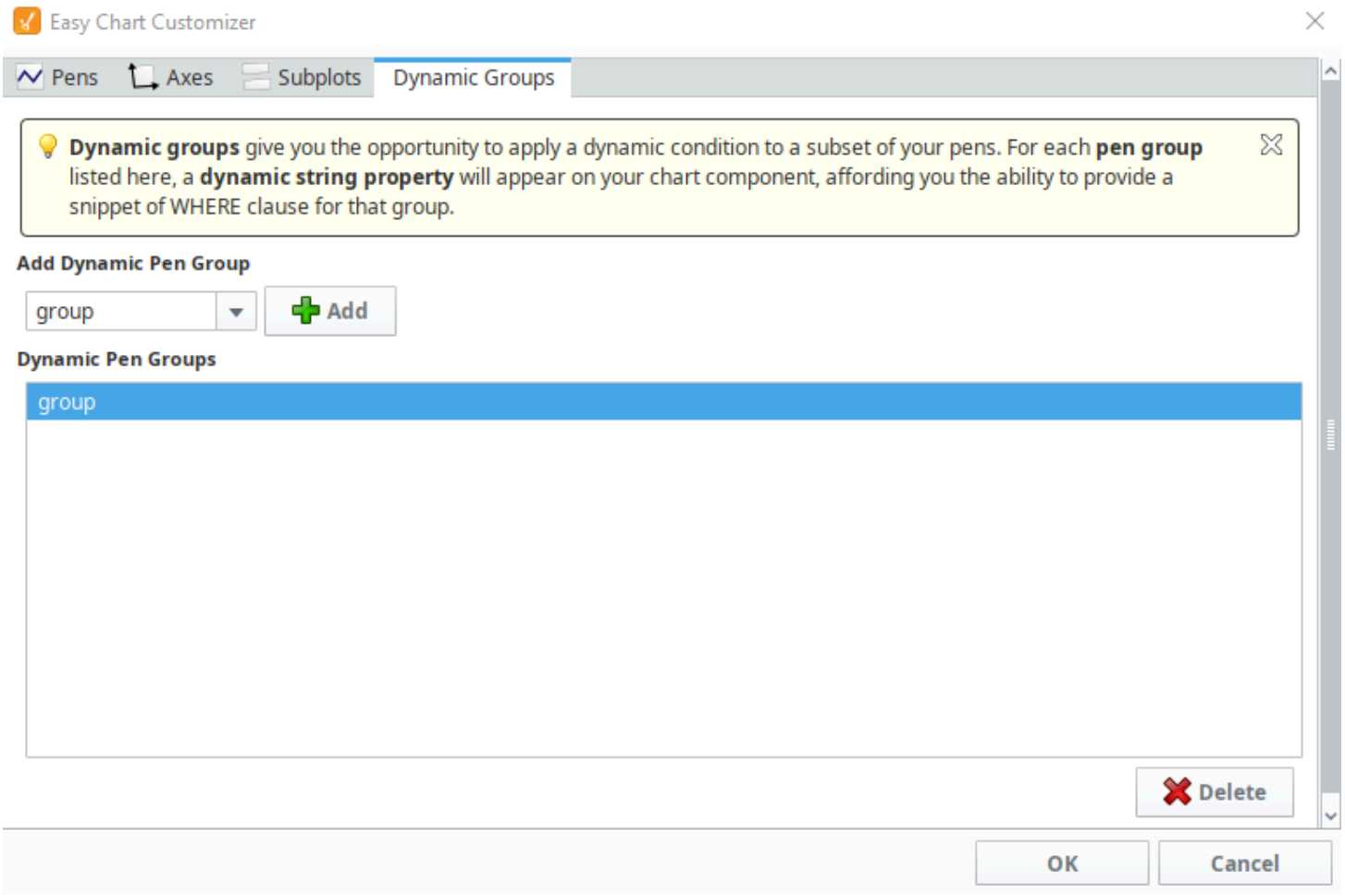


> **IN THE PENS TAB**
>
> Once you add a subplot, go to the Pens Tab, edit your pen, and put your pen into a different subplot.

For more information, go to **Easy Chart - Subplots**.

Dynamic Groups Tab

Dynamic Groups are used with Database pens. They allow you to apply a dynamic condition, like using a WHERE clause, to a subset of pens. For each pen group, a dynamic string property will appear in the Property Editor under Custom Properties of your Easy Chart component. You can create a WHERE clause that will search the database and return values if the pens meet a true condition.



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

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



To learn more about Dynamic Groups, refer to the **Easy Chart** section.

Vision - Easy Chart Scripting Functions

Scripting Functions

exportExcel(filename)

- Description
  - This function saves the chart's datasets as an Excel file. Returns a String of the complete file path chosen by the user, or None if the user canceled the save.
- Parameters
  - String **filename** - The default file name for the Save dialog.
- Return
  - String

print()

- Description
  - This function will print the chart.
- Parameters
  - None
- Return
  - None

setMode(mode)

- Description
  - Sets the current mode for the chart.

- Parameters
    - Integer **mode** - The mode to set the chart to. The mode options are as follows:
        - 0 : Zoom Mode. This is the default mode, where the user can draw a zoom rectangle with the mouse pointer.
        - 1 : Pan Mode. This mode lets the user use the mouse pointer to pan the chart to the left and right.
        - 3 : Mark mode. This mode lets the user click near a datapoint to annotate the point with its X and Y value.
        - 4 : X-Trace mode. This mode lets the user click and drag on the chart to see all values that fall along that X value.
- Return
    - None

exportDatasets()

- Description
    - Returns an Array List of datasets, representing the time series data of each type of pen.
- Parameters
    - None
- Return
    - Array List of datasets. Each dataset represents timeseries data for set of pens. The order of the datasets are listed below.

Index order of datasets

Index	Dataset
0	Tag Pens
1	Calculated Pens
2	Database Pens

Python - Accessing the Tag Pens Dataset

```python
# This example extracts the Easy Chart's Tag Pens dataset and pushes it to a Power Table.
# Place this script on the Easy Chart's propertyChange event.

if event.propertyName == 'tagPens':
    # Capture references up front.
    # This is safer than using `event` directly inside the delayed call,
    # since by the time the function executes, the event object may no longer be valid.
    chart = event.source
    table = chart.parent.getComponent('Power Table')

    # Wrap our dataset behavior in a function so we can pass it to invokeLater
    def update_table():
        # Export datasets; index 0 contains Tag Pens series data
        datasets = chart.exportDatasets()
        table.data = datasets[0]

    # Delay until after the chart finishes loading the new tag
    system.vision.invokeLater(update_table, 1000)
```

Extension Functions

configureChart

- Description
    - Provides an opportunity to perform further chart configuration via scripting. Doesn't return anything.

- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - JFreeChart **chart** - A JFreeChart object. Refer to the JFreeChart documentation for API details.
- Return
  - None

getXTraceLabel

- Description
  - Provides an opportunity to configure the x-trace label. Return a string to override the default label.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - JFreeChart **chart** - A JFreeChart object. Refer to the JFreeChart documentation for API details.
  - String **penName** - The name of the pen the x-trace label applies to.
  - Integer **yValue** - The y-value of the pen at the x-trace location.
- Return
  - None

onPowerTableRowsDropped

- Description
  - Called when the user has dropped rows from a power table on the chart. The source table must have dragging enabled.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - Component **sourceTable** - A reference to the table that the rows were dragged from.

- List **rows** - An array of the row indicies that were dragged, in the order they were selected.
- Dataset **rowData** - A dataset containing the rows that were dragged.

- Return
  - None

onTagsDropped

- Description
  - Called when the user has dropped tags from the tag tree onto the chart. Normally, the chart will add pens automatically when tags are dropped, but this default behavior will be suppressed if this extension function is implemented.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - List **paths** - A list of the tag paths that were dropped on the chart.
- Return
  - None

Example - Pen Name Replacement

```
# This will take a tag that gets dropped from a Tag Browse Tree set in Realtime
Tag Tree mode,
# and will replace the underscores in the name of the tag "_" and replace them
with spaces.
tagPens = self.tagPens

for tag in paths:
    tagPath = tag.replace("default", "~")
    splitTag = tag.split("/")
    name = splitTag[-1].replace("_", " ")

    newRow = [name, tagPath, "MinMax", "Default Axis", 1, 1,
system.vision.color(255, 85, 85, 255), "", 1, 1, 0, 1, 0, "", 0, 0, 0, 1, 0, 0]

    self.tagPens = system.dataset.addRow(tagPens, newRow)
```

Example - Group Name

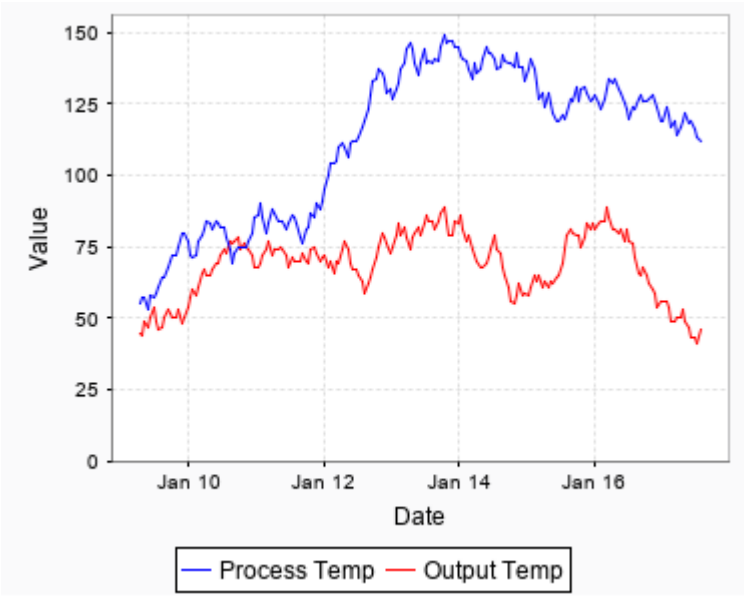
```
# This will take a tag that gets dropped from a Tag Browse Tree set in Realtime
Tag Tree mode,
# and will place it into a Pen Group titled "My Group Name".

tagPens = self.tagPens
groupName = "My Group Name"
for tag in paths:

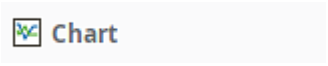
    newRow = [name, tagPath, "MinMax", "Default Axis", 1, 1,
system.vision.color(255, 85, 85, 255), "", 1, 1, 0, 1, 0, groupName, 0, 0, 0, 1,
0, 0]

    self.tagPens = system.dataset.addRow(tagPens, newRow)
```

Vision - Chart



Component Palette Icon:



Description

The Chart component (also called the Classic Chart when contrasted with the Easy Chart) provides a flexible way to display either timeseries or X-Y charts that are powered by any number of datasets. Typically, these datasets are bound to **SQL Query Bindings in Vision**.

Features

- SQL Query and/or SQLTags Historian data sources
- Zoom, Pan, X-Trace modes
- Any number of Y-axes and subplots
- Realtime or Historical
- Many different rendering styles

Configuration

The basic idea behind configuring the classic chart is simple: add datasets, and fill them in with data in a format that the chart understands. You add datasets to the chart using the chart's customizer. You then use standard property bindings to put data into these charts. Commonly you'll use **SQL Query**

Bindings in Vision. Since these datasets are just normal dynamic properties, you can also access them via scripting.

The Customizer also lets you add additional X and Y axes. There are various types of axes, and they each have a large number of properties. Lastly, you can configure additional properties for each dataset, such as which axes it maps to, its visual style, subplot, etc.

Datasets

Each dataset should define one or more "series" (a.k.a "pens"). The format for these datasets is quite simple. Each series in a dataset shares common X-values, defined by the first column. Each additional column are the Y-values for a series.

Chart Type: XY vs Category

The classic chart is typically in XY Plot mode. This means that the X-axis is either date or numeric, and the Y-axes are numeric. If your X-axis is categorical (names, not numbers), you can switch the Chart Type property to Category Chart in the Property Editor. Don't be surprised when you get a few errors - you'll need to go and switch your X-axis to be a Category Axis, and fill your dataset in with valid category data, that is, String-based X-values. This is most often used with the Bar Renderer (see the **Vision - Chart Customizer**).

> **NOTE**
>
> You can bring up a context menu for this component when right-clicking on it either in the Designer's Preview Mode or in a Vision Client. See the **Charting - Right Click Menu** page for more details.

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), Field Border, and Line Border. **Note:** The	Border	.border	Common

Name	Description	Property Type	Scripting	Category
	border is unaffected by rotation.			
Chart Orientation	The orientation of the domain axis of the chart.	int	.orientation	Appearance
Chart Title	An optional title that will appear at the top of the chart.	String	.title	Appearance
Chart Type	Choose the type for this chart: XY (Numeric X-axis) or Category (String X-axis).	int	.chartType	Behavior

Name	Description	Property Type	Scripting	Category
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Extract Order	Extract order for how category datasets should be interpreted.	int	.extractOrder	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of	String	.toolTipText	Common

Name	Description	Property Type	Scripting	Category
	this component.			
Name	The name of this component.	String	.name	Common
Plot Background	The background color for all plots, unless they override it.	Color	.plotBackground	Appearance
Properties Loading	The number of properties currently being loaded. (Read only. Usable in bindings and scripting.)	int	.propertiesLoading	Uncategorized
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Selected Datapoint	The currently selected datapoint. (Read only. Usable in bindings and scripting.)	String	.selectedData	Uncategorized

Name	Description	Property Type	Scripting	Category
Selected X Value	The selected domain axis value for X-Trace and Mark modes. (Read only. Usable in bindings and scripting.)	String	.selectedXValue	Uncategorized
Selection Enabled?	If true, the user will be able to select datapoints on the chart. The selected datapoint will be highlighted, and the selectedData property will reflect it.	boolean	.selectionEnabled	Behavior
Selection Highlight Color	The color of the selection highlight.	Color	.selectionHighlightColor	Appearance
Selection Highlight Width	The line width of the selection highlight.	float	.selectionHighlightWidth	Appearance
Show Legend?	If true, a legend will be shown for the series	boolean	.legend	Appearance

Name	Description	Property Type	Scripting	Category
	displayed in the chart.			
Show Popup?	If true, a popup menu will be shown on right-click that allows the user to change mode, print, save, etc.	boolean	.showPopup	Behavior
Show Tooltips?	If true, tooltips showing point values will be displayed.	boolean	.tooltips	Behavior
Subplot Mode	The axis that subplots share if more than one subplot.	int	.subplotMode	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

See the **Vision - Chart Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

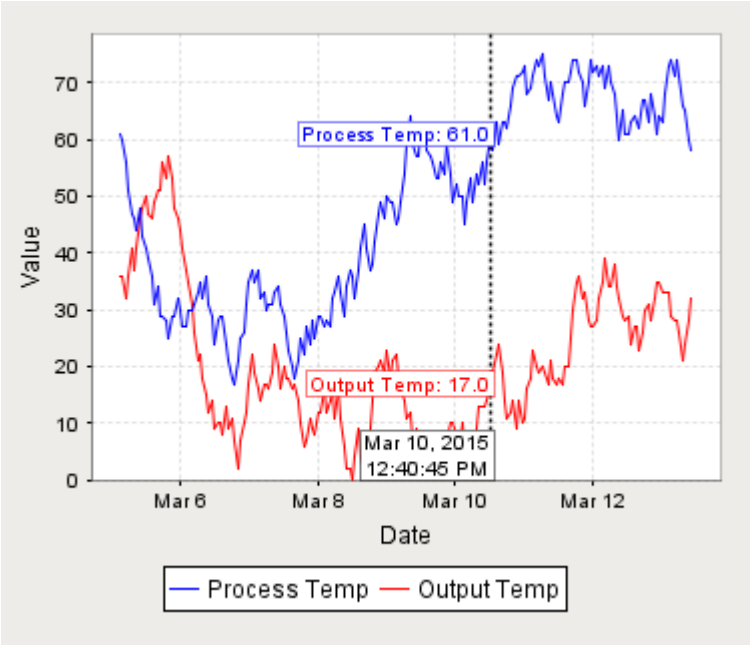
Customizers

The Chart component uses its own customizer called the **Vision - Chart Customizer**. You can add datasets and additional XY axes to a chart using the tabs in the chart customizer. You can configure additional properties for each dataset, like what axes it maps to as well as select from a host of visual styles. It also has six axis types to choose from, each with an extensive list of properties.

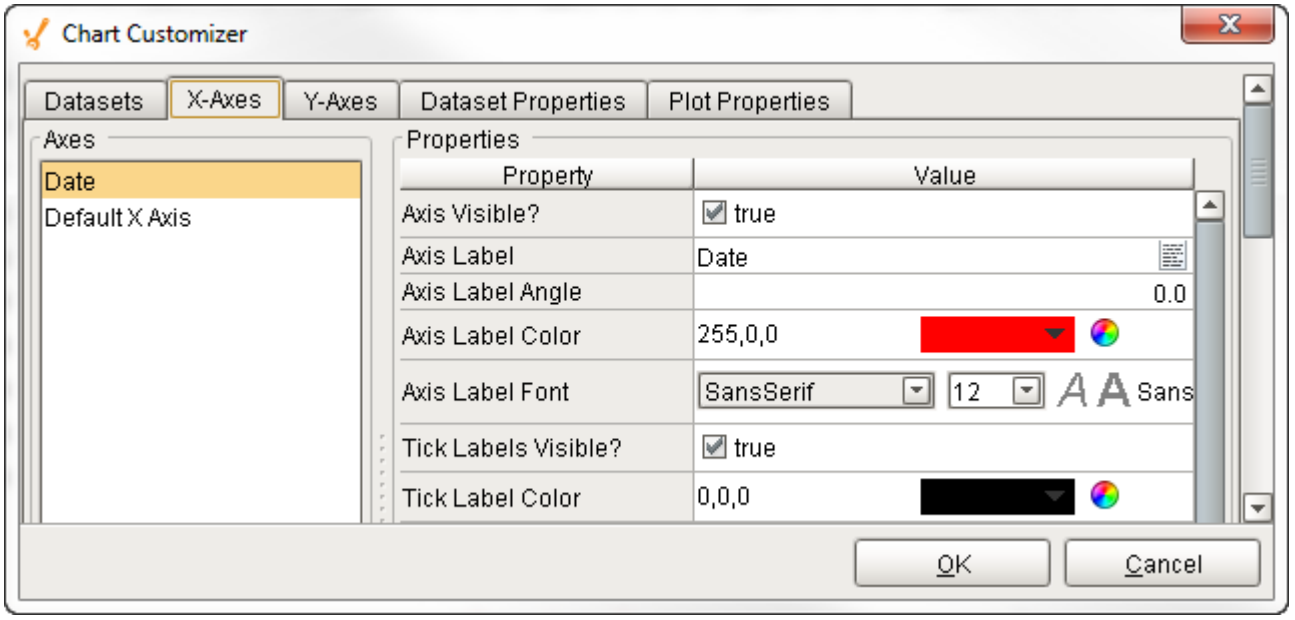
The customizer already has some default styles in place to help you get started, but you can modify these default settings to your own style. Refer to the **Vision - Chart Customizer** section for property descriptions and examples of chart axis types.

- **Vision - Chart Customizer**
- **Vision Component Customizers**

Example


Process Temp: 61.0
Output Temp: 17.0
Mar 10, 2015
12:40:45 PM
Value
Date
Mar 6
Mar 8
Mar 10
Mar 12
Process Temp
Output Temp

Vision - Chart Customizer



Description

The **Chart component**, also known as the Classic Chart, can be used to make almost any kind of chart. It provides a flexible way to display XY charts using a host of built-in properties. All you need to do to create a chart is add datasets, fill them in with data, configure a property binding, and setup the chart properties using the customizer.

Customizer

The Chart component has its own special customizer called the Chart Customizer. When you open the customizer, you'll notice five tabs at the top: Dataset, X-Axes, Y-Axes, Dataset Properties, and Plot Properties. Each tab has its own set of properties and defaults.

To get started, first add your dataset(s) and any additional XY axes using the appropriate tabs in the customizer. You can configure additional properties for each dataset, like what axes the data maps to, as well as select from a host of visual styles.

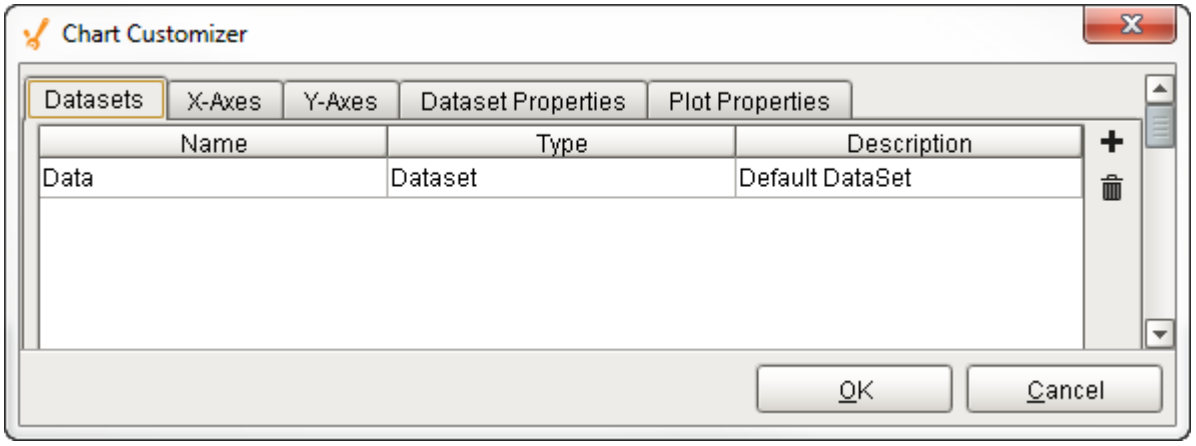
There are six types of axes to choose from when configuring a chart, each having its own list of properties: Number Axis, Date Axis, Category Axis, Logarithmic Axis, Elapsed Axis, and Symbols Axis. Most of the X and Y axes properties are used in the customizer, and some properties are specific to the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

The customizer already has some default styles in place to help you get started, but you can modify these default settings to your own style using the XY properties, Axes Type, Renderer and Plot styles. If you don't like one style, try another.

Shown below is each tab in the Chart Customizer with all its properties, description, and what axes type it supports. Note: Not all properties are available for all axes type charts.

Datasets Tab

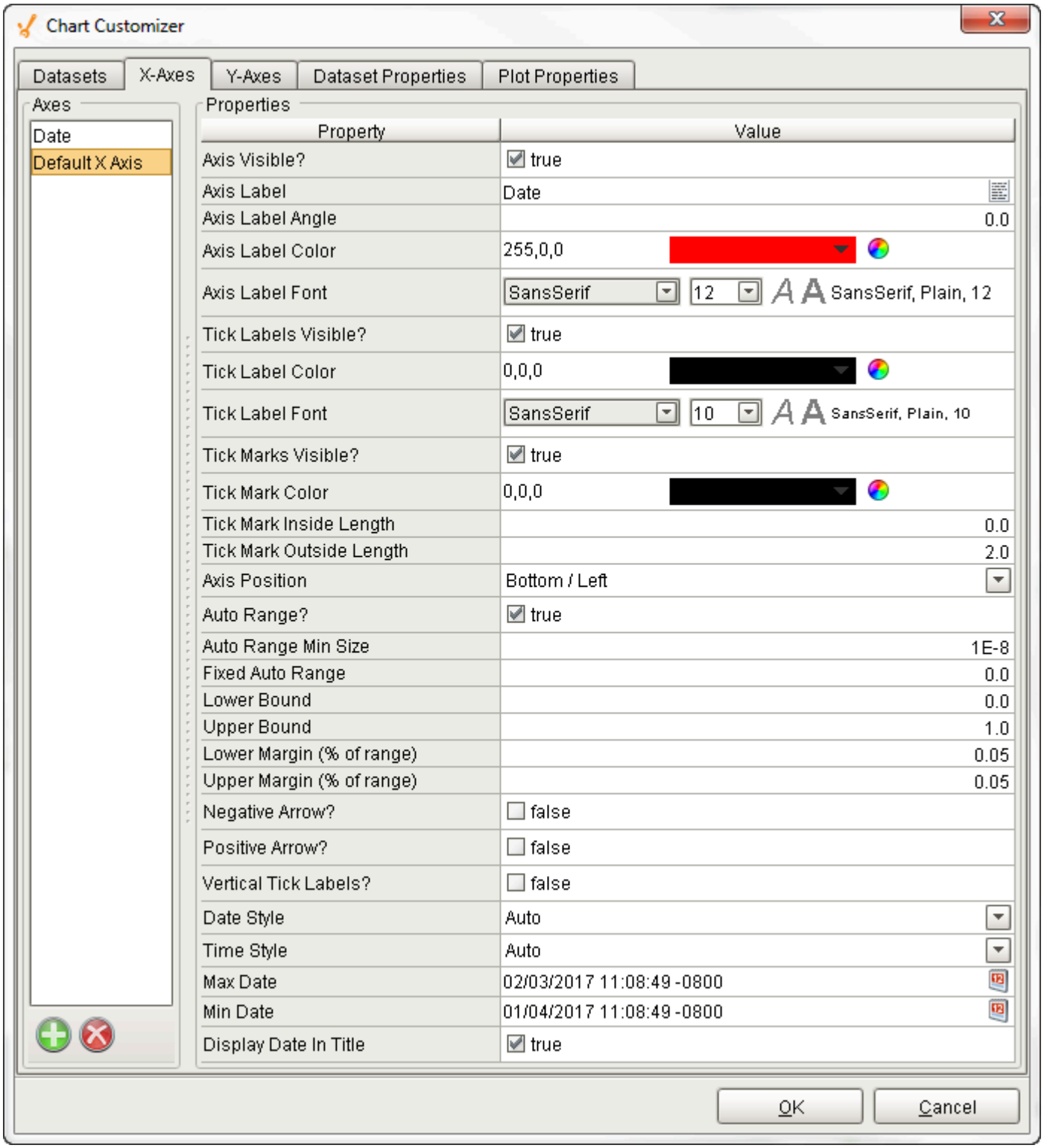
The Dataset tab is where you set up, add, and remove datasets.

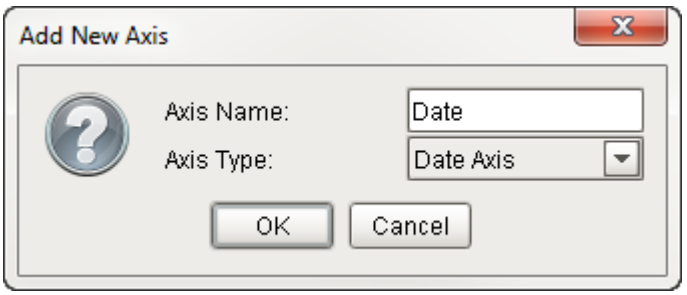


Property	Description
Data	Default dataset property.
+ Add	Adds a new dataset. Click the plus icon a new row will be added. Enter the dataset Name and Description.
Delete	Deletes an existing dataset. Click the Delete icon to delete an existing dataset.
Name	Name of the dataset. Double click in the field to rename the dataset.
Type	Default type is "Dataset."
Description	Describes the dataset.

X-Axes Tab

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


Chart Customizer
Datasets
X-Axes
Y-Axes
Dataset Properties
Plot Properties
Axes
Date
Default X Axis
Properties
Property
Value
Axis Visible?
true
Axis Label
Date
Axis Label Angle
0.0
Axis Label Color
255,0,0
Axis Label Font
SansSerif
12
SansSerif, Plain, 12
Tick Labels Visible?
true
Tick Label Color
0,0,0
Tick Label Font
SansSerif
10
SansSerif, Plain, 10
Tick Marks Visible?
true
Tick Mark Color
0,0,0
Tick Mark Inside Length
0.0
Tick Mark Outside Length
2.0
Axis Position
Bottom / Left
Auto Range?
true
Auto Range Min Size
1E-8
Fixed Auto Range
0.0
Lower Bound
0.0
Upper Bound
1.0
Lower Margin (% of range)
0.05
Upper Margin (% of range)
0.05
Negative Arrow?
false
Positive Arrow?
false
Vertical Tick Labels?
false
Date Style
Auto
Time Style
Auto
Max Date
02/03/2017 11:08:49 -0800
Min Date
01/04/2017 11:08:49 -0800
Display Date In Title
true
OK
Cancel

Property	Description	Supports Axes Types
	Add X axis. When you add an X axis, you can select from one of the following axis types: Number, Date, Category, Logarithmic, Elapsed, and Symbol. Click the green plus icon, select an Axis Type, enter an Axis Name, and click OK. 	All
	Delete an existing axis. Select the axis, and click the Delete icon.	All
Axis Visible	If false, the axis will be hidden.	All
Axis Label	Name of the axis.	All
Axis Label Angle	Angle of the value on the axis label.	All
Axis Label Color	Color of axis label.	All
Axis Label Font	Font type and size of text on axis label.	All
Tick Labels Visible	If false, the tick labels will be hidden.	All
Tick Label Color	Color of tick labels.	All
Tick Label Font	Font type and size of text on tick labels.	All
Tick Marks Visible	If false, the tick marks will be hidden.	All
Tick Mark Color	Color of tick marks.	All
Tick Mark Inside Length	Length of tick marks inside the chart.	All

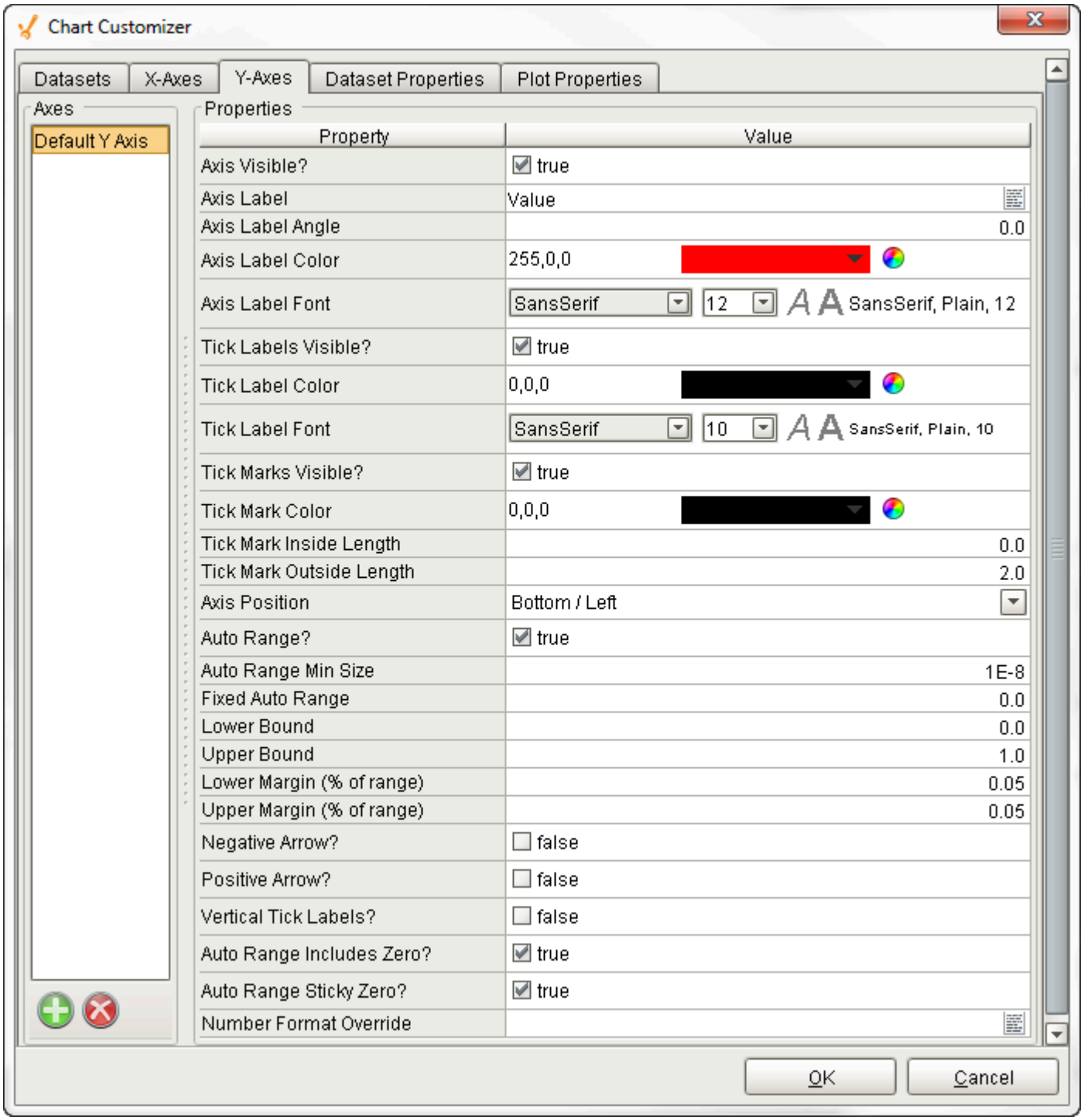
Property	Description	Supports Axes Types
Tick Mark Outside Length	Length of tick marks outside the chart.	All
Axis Position	Depends on the axis selected. X-axis label alternates between top and bottom. Y-axis label alternates between left and right. You many need to change both X and Y axis properties to get your intended axis position.	All
Auto Range	If true, the value axis range will be determined automatically. If false, the specified Lower and Upper bounds will be used.	All
Auto Range Min Size	If true, the minimum value range is used.	Date, Number, Logarithmic, Symbol, Elapsed
Fixed Auto Range	Sets an axis up for dynamic graphs.	Date, Number, Logarithmic, Symbol, Elapsed
Lower Bound	Lower bound value. Used only when Auto Range is false.	Date, Number, Logarithmic, Symbol, Elapsed
Upper Bound	Upper bound value. Used only when Auto Range is false.	Date, Number, Logarithmic, Symbol, Elapsed
Lower Margin (% of range)	Lower margin represented as a percentage. Used only when Auto Range is true.	Date, Number, Logarithmic, Symbol, Elapsed
Upper Margin (% of range)	Upper margin represented as a percentage. Used only when the Auto Range is true.	Date, Number, Logarithmic, Symbol, Elapsed

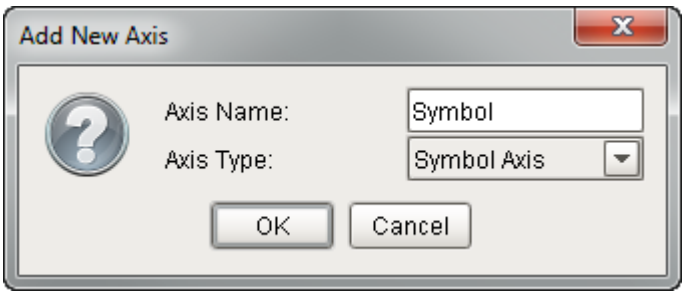
Property	Description	Supports Axes Types
Negative Arrow	If true, negative arrow is visible.	Date, Number, Logarithmic, Symbol, Elapsed
Positive Arrow	If true, positive arrow is visible.	Date, Number, Logarithmic, Symbol, Elapsed
Vertical Tick Labels	Vertical orientation for tick labels.	Date, Number, Logarithmic, Symbol, Elapsed
Auto Range Includes Zero	If true, the range includes a zero.	Date, Number, Logarithmic, Symbol, Elapsed
Auto Range Sticky Zero	If true, the zero is on both the XY axes.	Date, Number, Logarithmic, Symbol, Elapsed
Number Format Override	Overwrites the current number format.	Date, Number, Logarithmic, Symbol
Date Style	The style of the date displayed on the axis.	Date
Time Style	The style of the time displayed on the axis.	Date
Max Date	Max value in a series of dates.	Date
Min Date	Min value in a series of dates.	Date
Display Date in Title	If true, the date will be displayed in the title when the range is zoomed into the hour range.	Date
Label Angle	The angle for the value axis labels.	Category

Property	Description	Supports Axes Types
"1e#"-style tick labels	If true, uses scientific notation format (i.e.,1e5, 1e6, etc.,).	Logarithmic
"10^n"-style tick labels	If true, uses power notation format (i.e., 10 to the "X" power).	Logarithmic
Symbols String	Sequence of characters such as a literal constant. (i.e., On,Off,Auto)	Symbols
Grid Bands Visible	If true, grid bands will be hidden.	Symbols
Grid Bands Color	Color of grid bands.	Symbols
Grid Bands Alternate Color	Backup color of grid bands.	Symbols
Format String	Specified sequence of characters.	Elapsed

Y-Axes Tab

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


Chart Customizer
Datasets
X-Axes
Y-Axes
Dataset Properties
Plot Properties
Axes
Default Y Axis
Properties
Property
Value
Axis Visible?
true
Axis Label
Value
Axis Label Angle
0.0
Axis Label Color
255,0,0
Axis Label Font
SansSerif
12
SansSerif, Plain, 12
Tick Labels Visible?
true
Tick Label Color
0,0,0
Tick Label Font
SansSerif
10
SansSerif, Plain, 10
Tick Marks Visible?
true
Tick Mark Color
0,0,0
Tick Mark Inside Length
0.0
Tick Mark Outside Length
2.0
Axis Position
Bottom / Left
Auto Range?
true
Auto Range Min Size
1E-8
Fixed Auto Range
0.0
Lower Bound
0.0
Upper Bound
1.0
Lower Margin (% of range)
0.05
Upper Margin (% of range)
0.05
Negative Arrow?
false
Positive Arrow?
false
Vertical Tick Labels?
false
Auto Range Includes Zero?
true
Auto Range Sticky Zero?
true
Number Format Override
OK
Cancel

Property	Description	Supports Axes Types
	Add axis. When you add a Y axis, you can select from one of the following axis types: Number, Date, Category, Logarithmic, Elapsed, and Symbol. Click the green plus icon, select an Axis Type, enter an Axis Name, and click OK. 	All
	Delete an existing axis. Select an axis, and click the Delete icon.	All
Axis Visible	If false, the axis will be hidden.	All
Axis Label	Name of the axis.	All
Axis Label Angle	Angle of the value on the axis label.	All
Axis Label Color	Color of axis label.	All
Axis Label Font	Font type and size of text on axis label.	All
Tick Labels Visible	If false, the tick labels will be hidden.	All
Tick Label Color	Color of tick labels.	All
Tick Label Font	Font type and size of text on tick labels.	All
Tick Marks Visible	If false, the tick marks will be hidden.	All
Tick Mark Color	Color of tick marks.	All
Tick Mark Inside Length	Length of tick marks inside the chart.	All

Property	Description	Supports Axes Types
Tick Mark Outside Length	Length of tick marks outside the chart.	All
Axis Position	Depends on the axis selected. X-axis label alternates between top and bottom. Y-axis label alternates between left and right. You many need to change both X and Y axis properties to get your intended axis position.	All
Auto Range	If true, the value axis range will be determined automatically. If false, the specified Lower and Upper bounds will be used.	All
Auto Range Min Size	If true, the minimum value range is used.	Date, Number, Logarithmic, Symbol, Elapsed
Fixed Auto Range	Sets an axis up for dynamic graphs.	Date, Number, Logarithmic, Symbol, Elapsed
Lower Bound	Lower bound value. Used only when Auto Range is false.	Date, Number, Logarithmic, Symbol, Elapsed
Upper Bound	Upper bound value. Used only when Auto Range is false.	Date, Number, Logarithmic, Symbol, Elapsed
Lower Margin (% of range)	Lower margin represented as a percentage. Used only when Auto Range is true.	Date, Number, Logarithmic, Symbol, Elapsed
Upper Margin (% of range)	Upper margin represented as a percentage. Used only when the Auto Range is true.	Date, Number, Logarithmic, Symbol, Elapsed

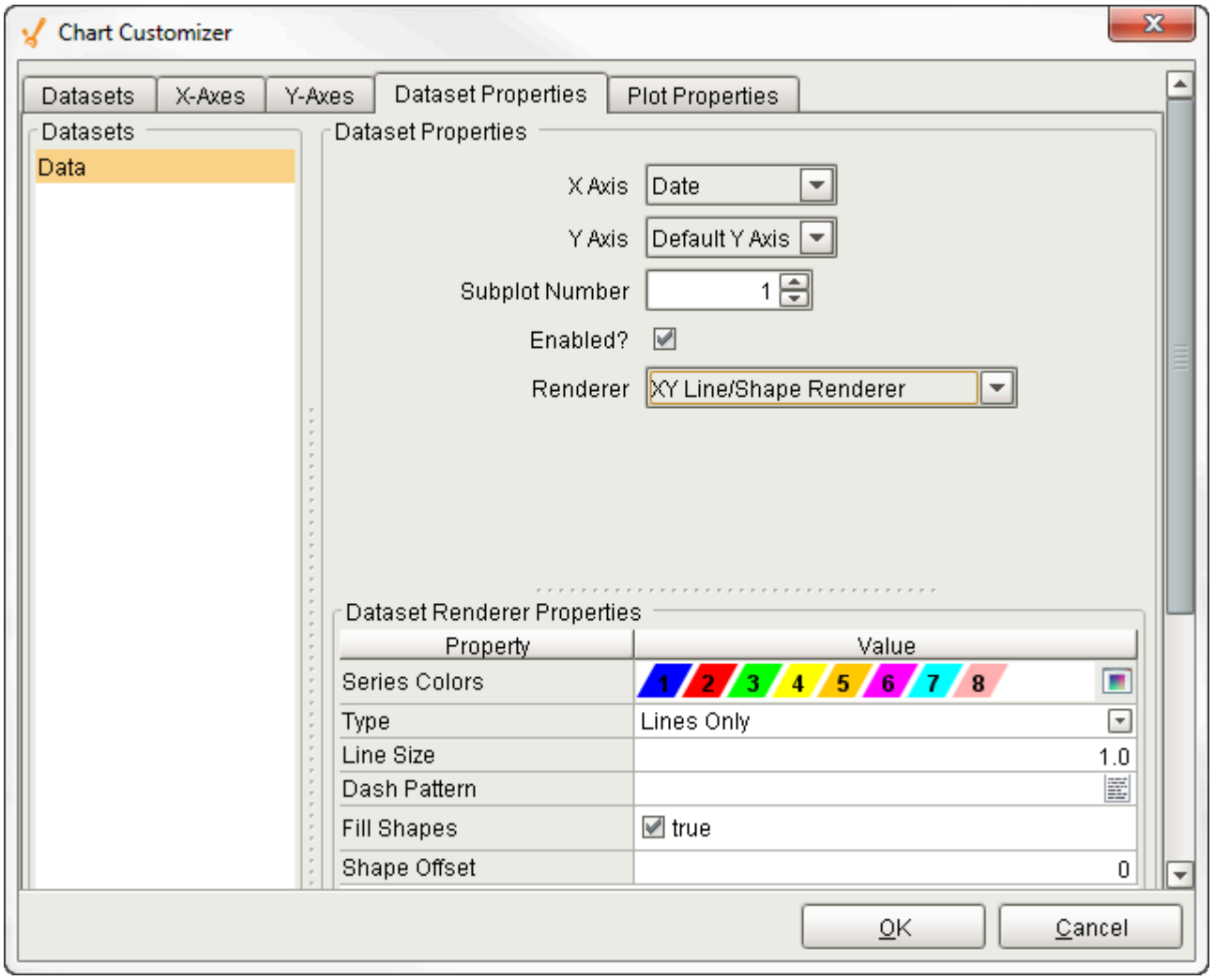
Property	Description	Supports Axes Types
Negative Arrow	If true, negative arrow is visible.	Date, Number, Logarithmic, Symbol, Elapsed
Positive Arrow	If true, positive arrow is visible.	Date, Number, Logarithmic, Symbol, Elapsed
Vertical Tick Labels	Vertical orientation for tick labels.	Date, Number, Logarithmic, Symbol, Elapsed
Auto Range Includes Zero	If true, the range includes a zero.	Date, Number, Logarithmic, Symbol, Elapsed
Auto Range Sticky Zero	If true, the zero is on both the XY axes.	Date, Number, Logarithmic, Symbol, Elapsed
Number Format Override	Overwrites the current number format.	Date, Number, Logarithmic, Symbol
Date Style	The style of the date displayed on the axis.	Date
Time Style	The style of the time displayed on the axis.	Date
Max Date	Max value in a series of dates.	Date
Min Date	Min value in a series of dates.	Date
Display Date in Title	If true, the date will be displayed in the title when the range is zoomed into the hour range.	Date
Label Angle	The angle for the value axis labels.	Category

Property	Description	Supports Axes Types
"1e#"-style tick labels	If true, uses scientific notation format (i.e.,1e5, 1e6, etc.,).	Logarithmic
"10^n"-style tick labels	If true, uses power notation format (i.e., 10 to the "X" power).	Logarithmic
Symbols String	Sequence of characters such as a literal constant. (i.e., On,Off,Auto)	Symbols
Grid Bands Visible	If true, grid bands will be hidden.	Symbols
Grid Bands Color	Color of grid bands.	Symbols
Grid Bands Alternate Color	Backup color of grid bands.	Symbols
Format String	Specified sequence of characters.	Elapsed

Dataset Properties Tab

The Dataset tab is where you can modify the visual styles of your chart. You can configure your chart with subplots, experiment with different renderer types and property types to change how the data is displayed until you find what best meets your requirements.

> **NOTE**
>
> Not all Renderer properties are available for each axis type.


Chart Customizer
Datasets
X-Axes
Y-Axes
Dataset Properties
Plot Properties
Datasets
Data
Dataset Properties
X Axis
Date
Y Axis
Default Y Axis
Subplot Number
1
Enabled?
Renderer
XY Line/Shape Renderer
Dataset Renderer Properties
Property
Value
Series Colors
1 2 3 4 5 6 7 8
Type
Lines Only
Line Size
1.0
Dash Pattern
Fill Shapes
true
Shape Offset
0
OK
Cancel

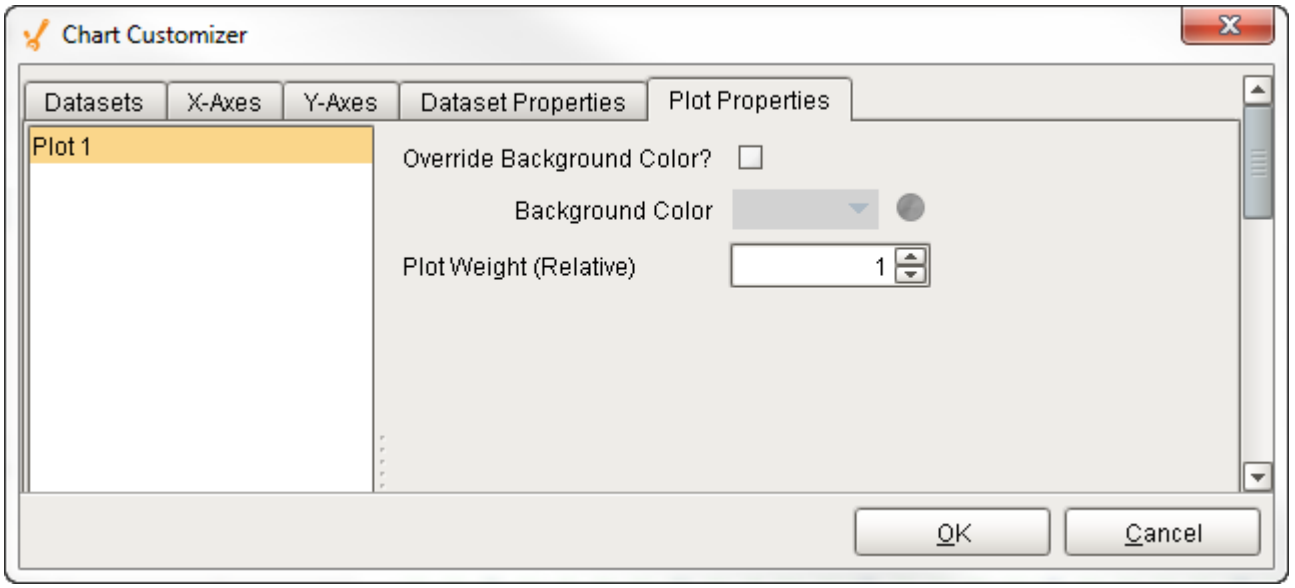
Property	Description	Supports Axes Types
Dataset	Collection of data in tabular form. Data from the dataset drives the chart.	All
X Axis	Horizontal axis.	All
Y Axis	Vertical axis.	All
Subplot Number	Number of plot areas on one chart.	All
Enabled	If true, the chart is displayed with the selected renderer properties.	All
Renderer	The visual style of the data presented on the chart. Select from various renderer styles: • XY Line/Shape Renderer • XY Bar Renderer • XY Area Renderer • XY Step Renderer • XY Step Area Renderer • XY Dot Renderer • Category Line/Shape Renderer • Category Bar Renderer	All
Series Colors	An ordered list of the colors to draw series in.	All
Type	Type of XY Item Renderer.	All
Line Size	The thickness of the line.	All
Dash Pattern	The pattern used for dashed lines.	All
Fill Shapes	If false, there is only an outline of the shape, no fill color.	All
Shape Offset	The offset into the standard shape list to start this renderer at. Offset values and respective values are	Offset

Property	Description	Supports Axes Types
	listed below. • 0 - Square • 1 - Circle • 2 - Upward triangle • 3 - Diamond • 4 - Horizontal rectangle • 5 - Downward triangle • 6 - Horizontal ellipse • 7 - Rightward triangle • 8 - Vertical rectangle • 9 - Leftward triangle	
Margin	The percentage by which the bars are trimmed using the XY Bar Renderer.	All
Shadows	If true, draws shadows under the bars using the XY Bar Renderer.	All
Outline	If true, draws an outline around the area using the XY Area Renderer.	All
Draw Lines	If true, lines will be drawn to connect the datapoints using the Category Line/Shape Renderer.	All
Draw Shapes	If true, shapes will be drawn to connect each datapoint if using the Category Line/Shape Renderer.	All
Date Style	The style of the date displayed on the axis.	Date
Time Style	The style of the time displayed on the axis.	Date
Max Date	Max value in a series of dates.	Date
Min Date	Min value in a series of dates.	Date

Property	Description	Supports Axes Types
Display Date in Title	If true, the date will be displayed in the title when the range is zoomed into the hour range.	Date
Label Angle	The angle for the value axis labels.	Category
"1e#"-style tick labels	If true, uses scientific notation format (i.e.,1e5, 1e6, etc.,).	Logarithmic
"10^n"-style tick labels	If true, uses power notation format (i.e., 10 to the "X" power).	Logarithmic
Symbols String	Sequence of characters such as a literal constant. (i.e., On,Off,Auto)	Symbols
Grid Bands Visible	If true, grid bands will be hidden.	Symbols
Grid Bands Color	Color of grid bands.	Symbols
Grid Bands Alternate Color	Backup color of grid bands.	Symbols
Format String	Specified sequence of characters.	Elapsed

Plot Properties Tab

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



Property	Description	Supports Axes Types
Plot	The chart area displaying data.	All
Override Background Color	If enabled, allows you to change the background color.	All
Background Color	Background color of the chart.	All
Plot Weight (Relative)	The chart ratio between subplots.	All

References

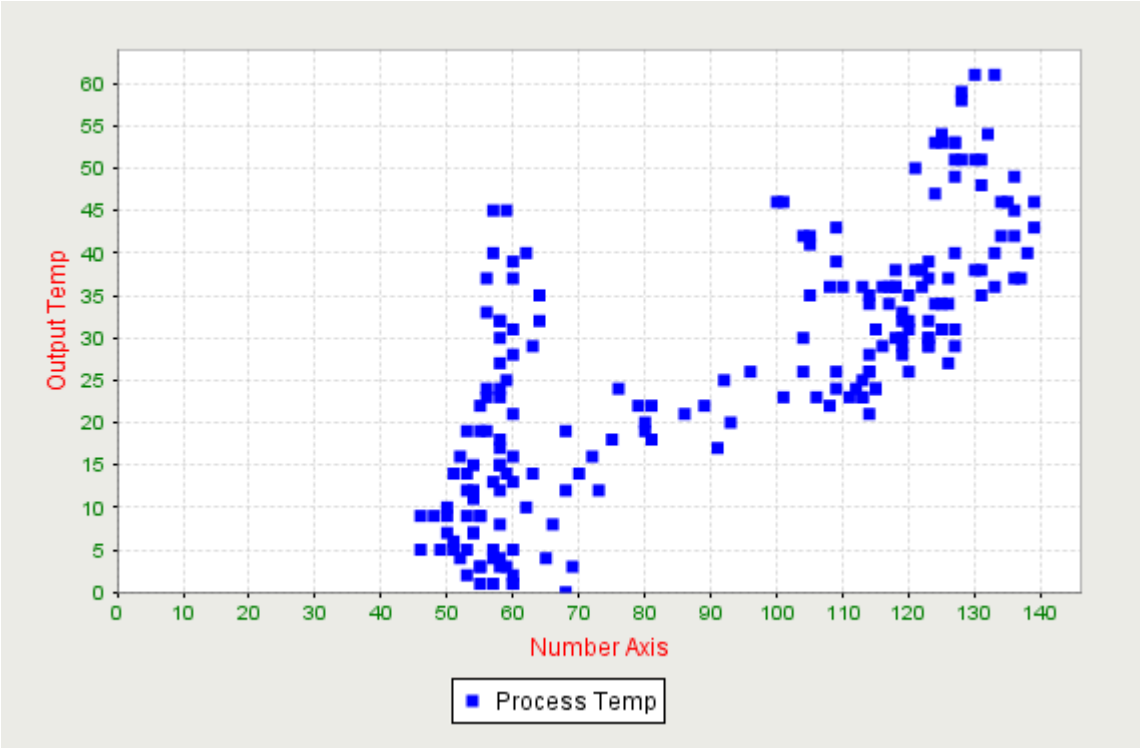
- **Vision - Chart**
- **Component Customizers**

Axis Type Examples

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

> **NOTE**
>
> Some customizer properties are specific to the axis type and have their own unique properties. Examples of all axis types are shown below along with the property settings used to create each chart.

Number Axis Chart



Binding Type

- Tag - Tag History

Chart Customizer Property Settings

Datasets Tab

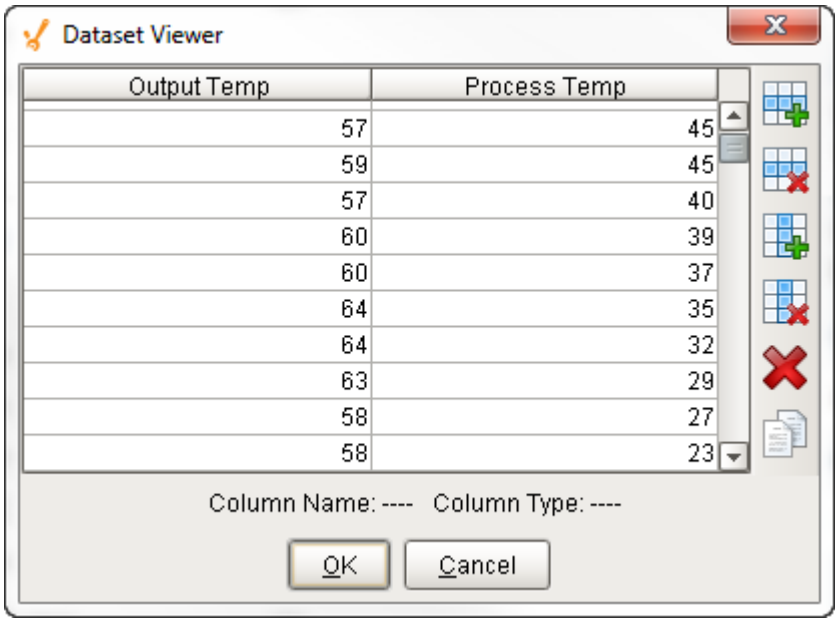
Property Name	Value
Datasets	Data
X-Axes Tab	

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

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

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

Dataset Property Dataset



Date Axis Chart

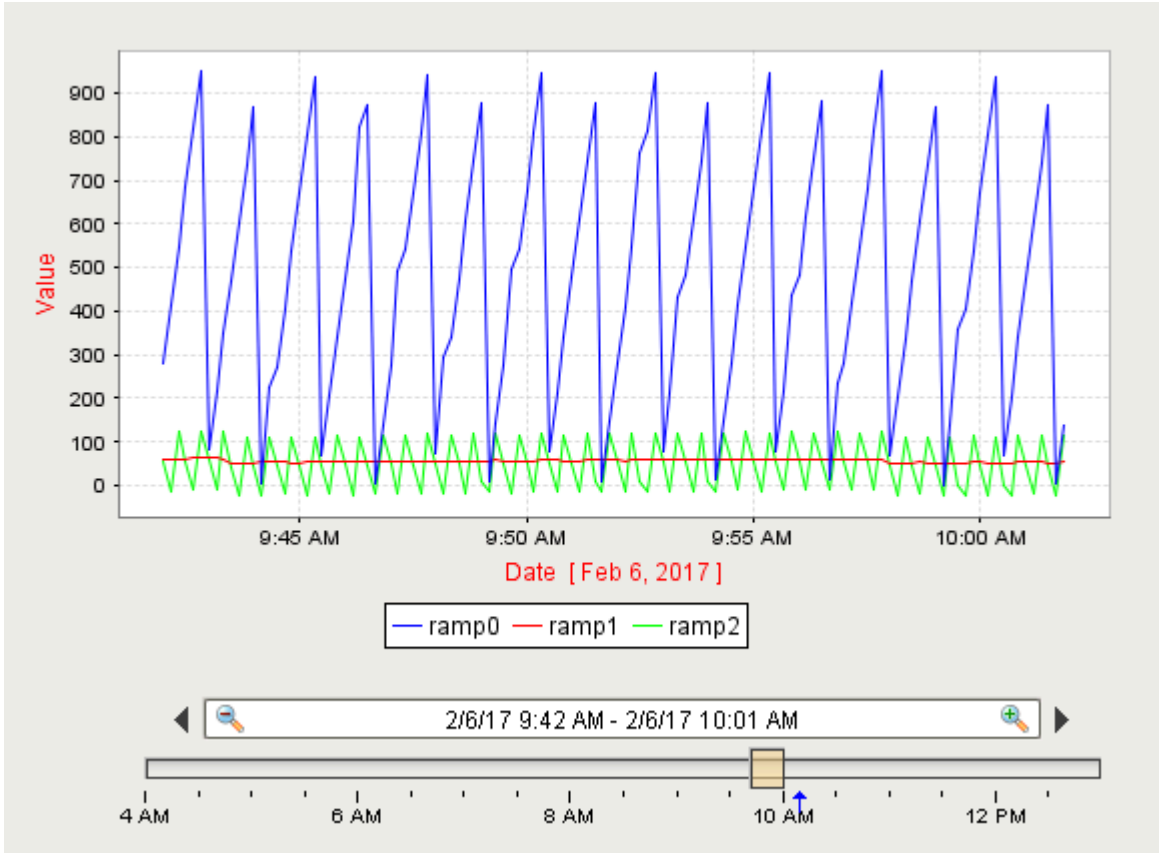


Chart Customizer Property Settings

Datasets Tab

Property Name	Value
Datasets	Data
X-Axes Tab	

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

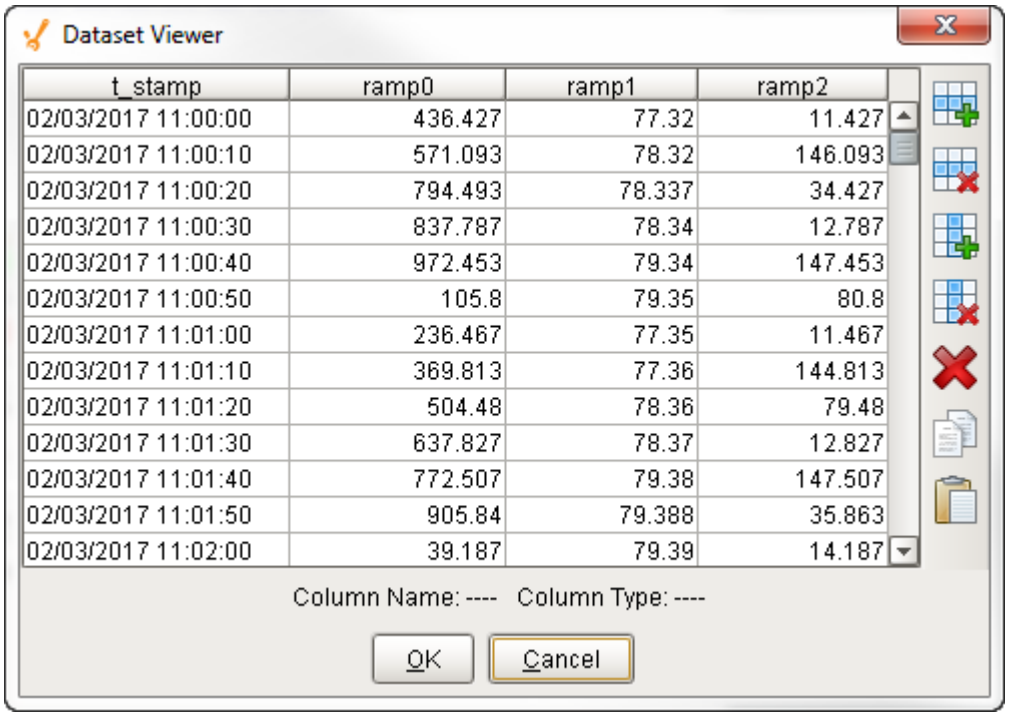
Y-Axes Tab

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

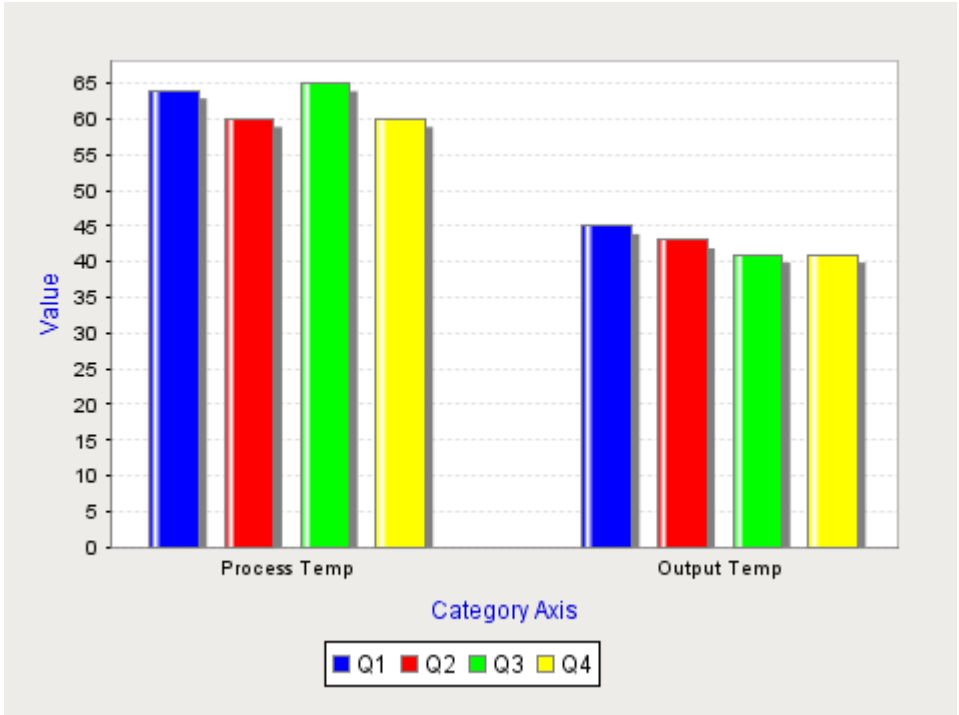
Dataset Properties Tab

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

Dataset Property Dataset



Category Axis Chart



Behavior

Property	Value
Chart Type	Category

Chart Customizer Property Settings

Datasets Tab

Property Name	Value
Datasets	Data
X-Axes Tab	

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

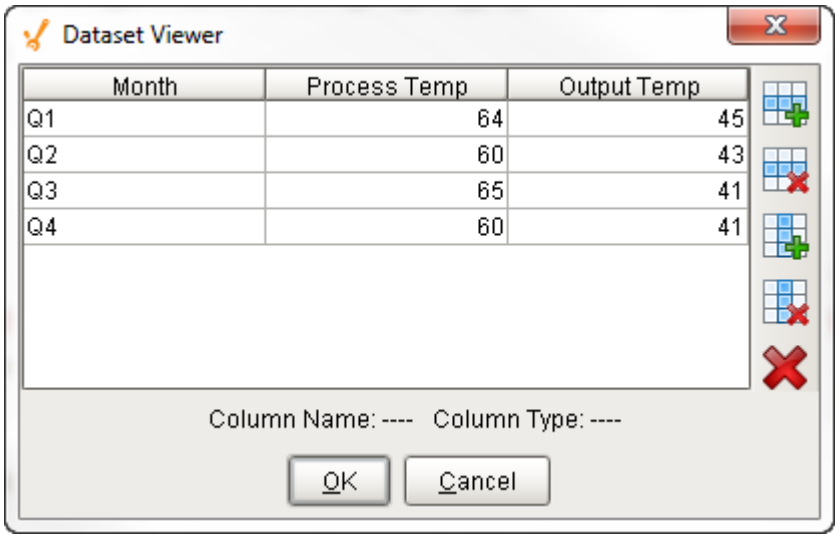
Y-Axes Tab

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

Dataset Properties Tab

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

Dataset Property Dataset



Logarithmic Axis Chart

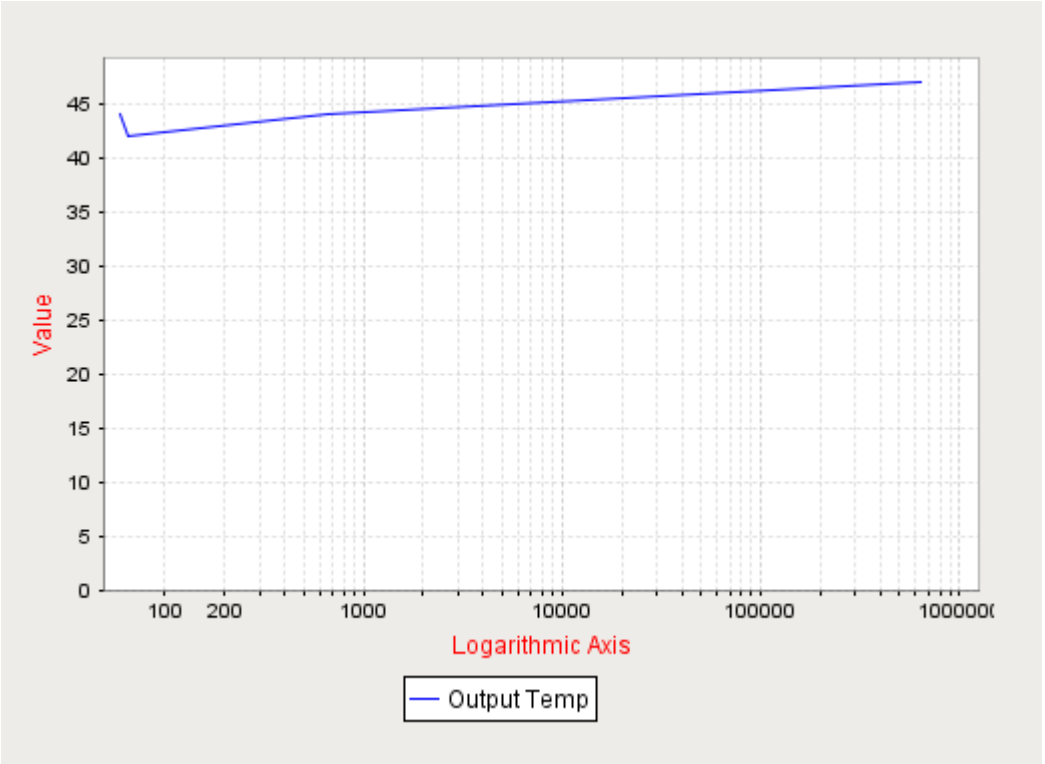


Chart Customizer Property Settings

Datasets Tab

Property Name	Value
Datasets	Data
X-Axes Tab	

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

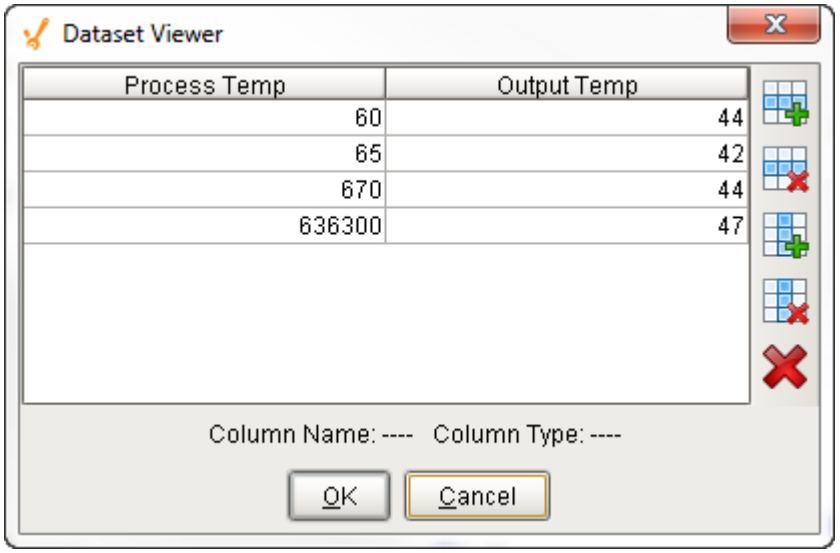
Y-Axes Tab

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

Dataset Properties Tab

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

Dataset Property Dataset



Symbols Axis Chart

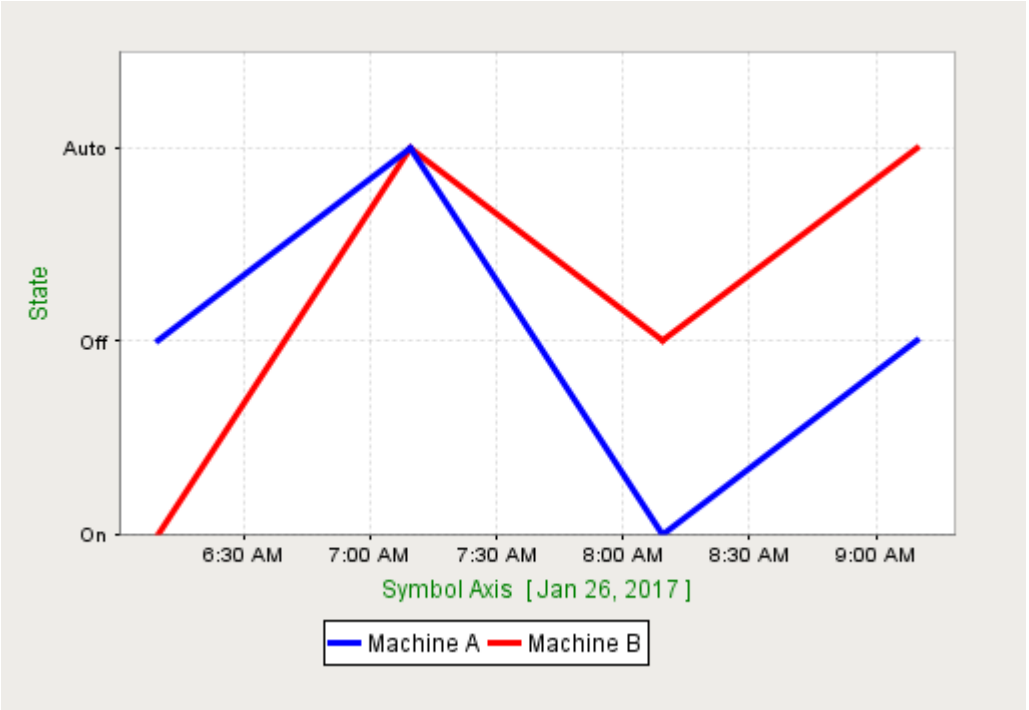


Chart Customizer Property Settings

Datasets Tab

Property Name	Value
Datasets	Data
X-Axes Tab	

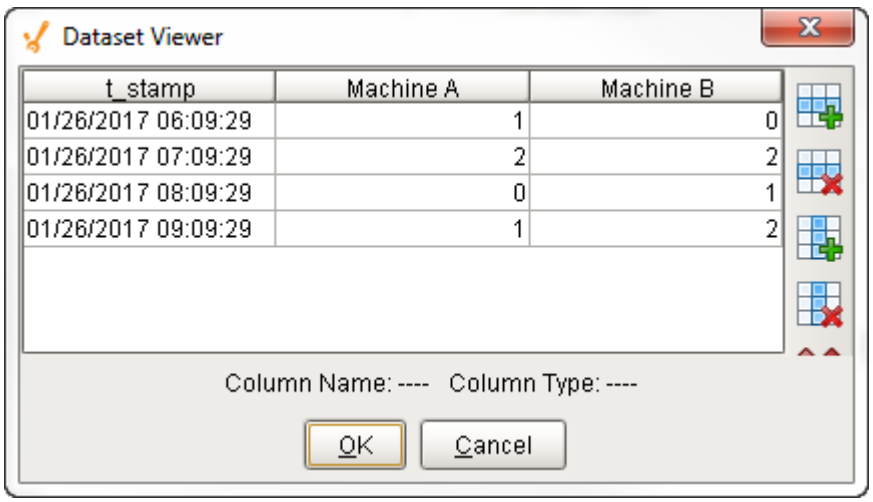
Property Name	Value
Axes	Default Axis
Axis Label	Symbol Axis
Axis Label Color	Green

Y-Axes Tab

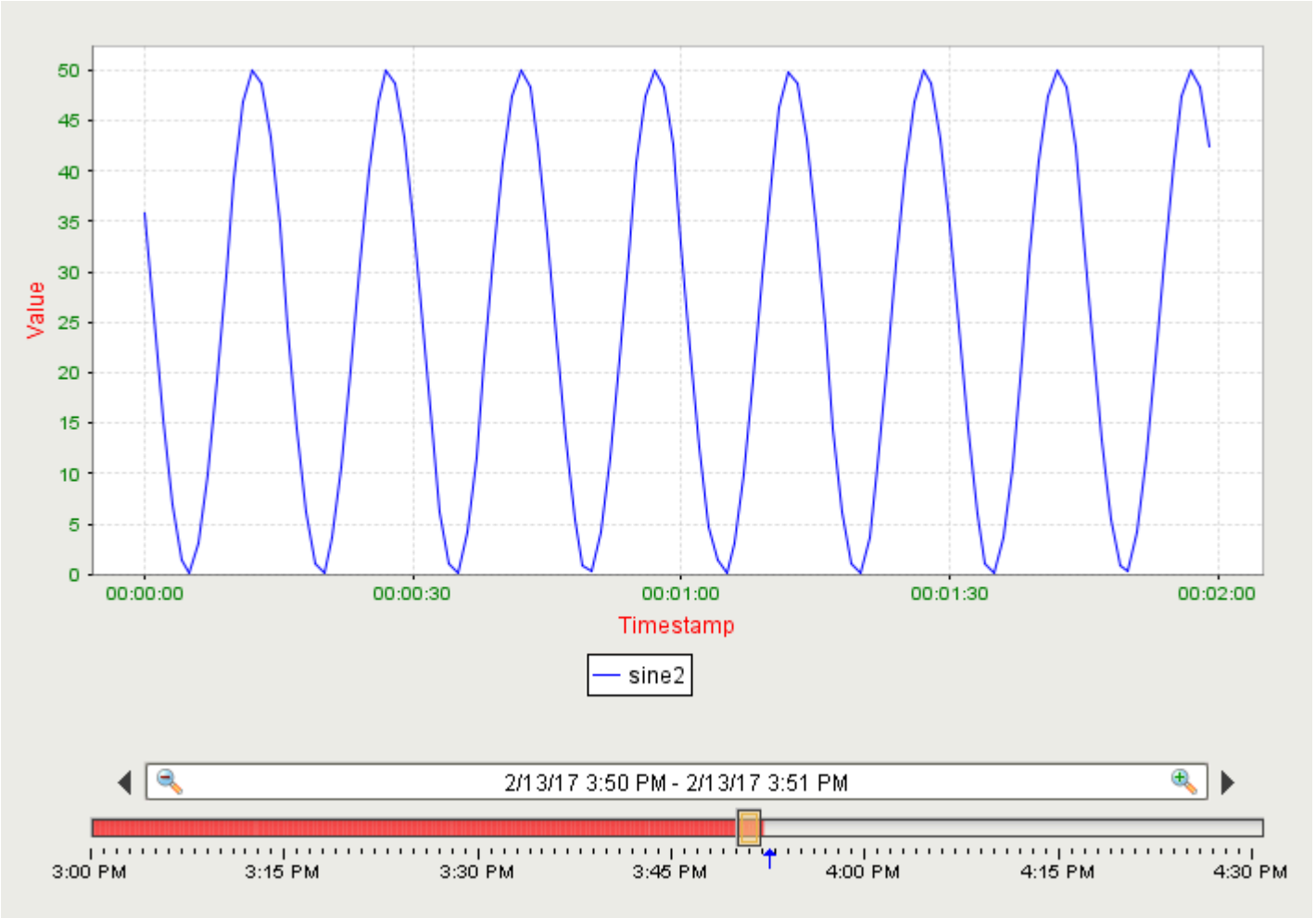
Property Name	Value
Axes	Symbol
Axis Label	State
Axis Label Color	Green
Symbols String	On,Off,Auto
Dataset Properties Tab	

Property Name	Value
Datasets	Data
X Axis	Default X Axis
Y Axis	Symbol
Renderer	XY Line/Shape Renderer
Type	Lines Only
Line Size	3

Dataset Property Dataset



Elapsed Time Axis Chart



Binding Type

- Database
- SQL Query

Chart Customizer Property Settings

Datasets Tab

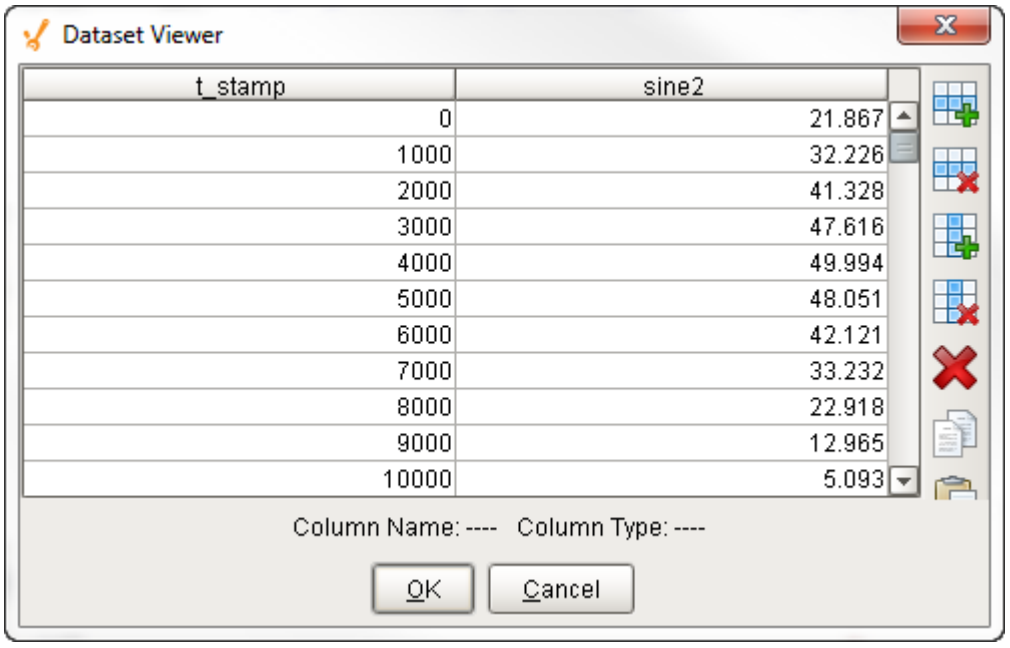
Property Name	Value
Dataset	Data
X-Axes Tab	

Property Name	Value
Axes	Elapsed Time
Axis Label	Timestamp
Axis Label Color	Red
Tick Label Color	Green
Upper Bound	60,000
Tick Size (ms)	30,000
Y-Axes Tab	

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

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

Dataset Property Dataset



Vision - Chart Scripting Functions

This page details the various component and extension functions available for **Vision's Chart component**.

Component Functions

getPlotProperties()

- Description
  - Retrieves the value of the selected PlotProperty objects that define background color and weight of each plot.
- Parameters
  - None
- Return
  - List

getProperties()

- Description
  - Retrieves the value of custom properties added to the Chart.
- Parameters
  - None
- Return
  - List

getSelectedData()

- Description
  - Returns the value of the selected chart entity as a string.
- Parameters
  - None
- Return
  - String

getSelectedEntity()

- Description
  - Returns the selected chart entity directly.
- Parameters
  - None
- Return
  - ChartEntity

getSubplotMode()

- Description
  - Retrieves the subplot mode that is currently in use:
    - 0 = Shared Domain
    - 1 = Shared Range
- Parameters
  - None
- Return
  - Int

getXAxes()

- Description
  - Returns a dictionary of the related rendering properties.
- Parameters
  - None
- Return
  - Dictionary<String, AxisConfig> - **AxisConfig** is an object that defines rendering properties

getYAxes()

- Description
  - Returns a dictionary of the related rendering properties.
- Parameters
  - None
- Return
  - Dictionary<String, AxisConfig> - **AxisConfig** is an object that defines rendering properties

refreshChart()

- Description
  - Refreshes the dataset for the specified subplot and dataset.
- Parameters
  - int **subplotIndex**
  - int **dataSetIndex**
- Return
  - None

setDatasetEnabled()

- Description
  - Sets a dataset to be enabled or not enabled.
- Parameters
  - string **dataSetName**
  - boolean **isEnabled** - A boolean representing if the dataset is enabled
- Return
  - None

setDatasetPlotNumber()

- Description
  - Sets a dataset's plot number.
- Parameters
  - string **dataSetName**
  - int **plotNumber**
- Return
  - None

setDatasetXAxis()

- Description
  - Sets a dataset's X axis name.
- Parameters
  - string **dataSetName**
  - string **axisName**
- Return
  - None

setDatasetYAxis()

- Description
  - Sets a dataset's Y axis name.
- Parameters
  - string **dataSetName**
  - string **axisName**
- Return
  - None

setSubplotMode()

- Description
  - Sets the subplot mode to be used when there is more than one subplot.
- Parameters
  - int **mode** - The mode to set the chart to. The mode options are as follows:
    - 0 - Shared Domain
    - 1 - Shared Range
- Return
  - None

setXAxes()

- Description
  - Sets defined rendering properties using AxisConfig objects.
- Parameters
  - PyDictionary string keys to **AxisConfig** objects
- Return
  - None

setYAxes()

- Description
  - Sets defined rendering properties using AxisConfig objects.
- Parameters
  - PyDictionary string keys to **AxisConfig** objects
- Return
  - None

Extension Functions

configureChart

- Description
  - Provides an opportunity to perform further chart configuration via scripting.
- Parameters
  - Component **self**- A reference to the component that is invoking this function.
  - JFreeChart **chart**- A JFreeChart object. Refer to the **JFreeChart documentation** for API details.
- Return
  - None

getXTraceLabel

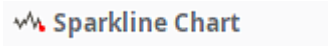
- Description
  - Provides an opportunity to configure the x-trace label. Return a string to override the default label.
- Parameters
  - Component **self**- A reference to the component that is invoking this function.

    - JFreeChart **chart** - A JFreeChart object. Refer to the **JFreeChart documentation** for API details.
    - String **penName** - The name of the pen the x-trace label applies to.
    - int **yValue** - The y-value of the pen at the x-trace location
- Return
    - None

Vision - Sparkline Chart



Component Palette Icon:



Description

The sparkline chart is a minimalistic chart component that displays a line-chart history for a single datapoint. Sparklines were invented by Edward Tufte as a way to show a great deal of contextual information in a very small amount of space. Sparklines are typically used to display the recent history (up to current time) of a datapoint so that the viewer can quickly discern the recent trend of a datapoint.

To use a sparkline, bind its Data property either to a Tag Historian realtime query, or to a database query. There should be two columns in this dataset: the first one a date column, the second a number. Each row will become a datapoint on the chart, and the dataset must be sorted by time in ascending order.

Instead of using axes to convey scale, the sparkline can display a band of color across the back of the chart which indicates the desired operating range of the datapoint. In this way, it is instantly obvious when a value is in its expected range, above that range, or below. The sparkline automatically configures its internal axes based on the data given to it. To give it a fixed range, fill in the Range High and Range Low properties.

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.background	Appearance
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Border Inset	The amount of space to inset the chart inside its border.	double	.borderInset	Appearance

Name	Description	Property Type	Scripting	Category
Chart Max	The value that corresponds to the upper edge of the chart. (Read only. Usable in bindings and scripting.)	Double	.chartMax	Uncategorized
Chart Min	The value that corresponds to the lower edge of the chart. (Read only. Usable in bindings and scripting.)	Double	.chartMin	Uncategorized
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Data	The history data to draw in the sparkline chart.	Dataset	.data	Data
Desired High	The high value of the desired operating range. If left blank (null), no desired range band will be shown.	Double	.desiredHi	Data
Desired Low	The low value of the desired operating	Double	.desiredLo	Data

Name	Description	Property Type	Scripting	Category
	range. If left blank (null), no desired range band will be shown.			
Desired Range Color	The color of the desired operating range band. Only used if the desired operating range is configured. See **Color Selector**.	Color	.desiredRangeColor	Appearance
First Marker Color	The color of the first value marker. See **Color Selector**.	Color	.firstMarkerColor	Markers
First Marker Size	The size of the first value marker.	double	.firstMarkerSize	Markers
First Marker Style	The style of the first value marker.	int	.firstMarkerStyle	Markers
First Value	The first (oldest) value in the dataset. (Read only. Usable in bindings and scripting.)	Double	.firstValue	Uncategorized
High Marker Color	The color of the high value marker. See **Color Selector**.	Color	.hiMarkerColor	Markers
High Marker Size	The size of the high value marker.	double	.hiMarkerSize	Markers
High Marker Style	The style of the high value marker.	int	.hiMarkerStyle	Markers

Name	Description	Property Type	Scripting	Category
Last Marker Color	The color of the last value marker. See **Color Selector**.	Color	.lastMarkerColor	Markers
Last Marker Size	The size of the last value marker.	double	.lastMarkerSize	Markers
Last Marker Style	The style of the last value marker.	int	.lastMarkerStyle	Markers
Last Value	The last (most recent) value in the dataset. (Read only. Usable in bindings and scripting.)	Double	.lastValue	Uncategorized
Line Color	The color of the sparkline. See **Color Selector**.	Color	.foreground	Appearance
Line Width	The width of the sparkline.	float	.lineWidth	Appearance
Low Marker Color	The color of the low value marker. See **Color Selector**.	Color	.loMarkerColor	Markers
Low Marker Size	The size of the low value marker.	double	.loMarkerSize	Markers
Low Marker Style	The style of the low value marker.	int	.loMarkerStyle	Markers

Name	Description	Property Type	Scripting	Category
Max Value	The largest value in the dataset. (Read only. Usable in bindings and scripting.)	Double	.maxValue	Uncategorized
Min Value	The smallest value in the dataset. (Read only. Usable in bindings and scripting.)	Double	.minValue	Uncategorized
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	int	.quality	Data
Range High	A fixed value for the upper edge of the chart. If left blank (null), the upper range will be calculated automatically.	Double	.rangeHi	Data
Range Low	A fixed value for the lower edge of the chart. If left blank	Double	.rangeLo	Data

Name	Description	Property Type	Scripting	Category
	(null), the lower range will be calculated automatically.			
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

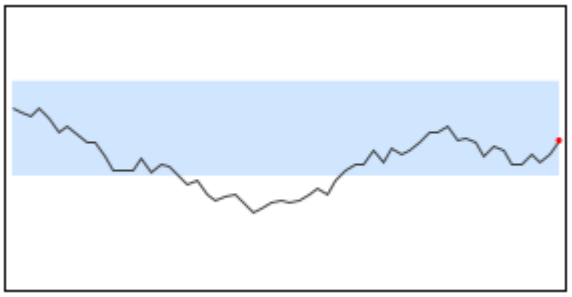
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the **Component Events** page

Customizers

- **Vision Component Customizers**
- **Style Customizer**

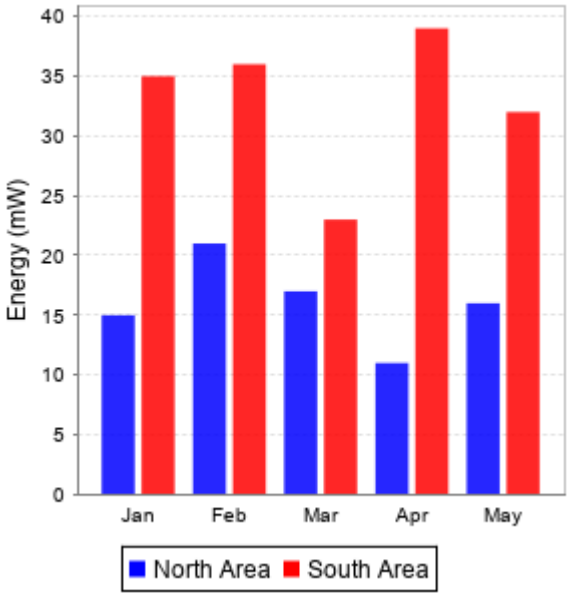
Examples

Sparkline Chart with Low and High Limits

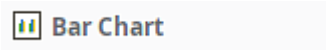


Property Name	Value
Desired Range Color	184,218,255
Range High	100
Range Low	0
Desired High	75
Desired Low	40

Vision - Bar Chart



Component Palette Icon:



Description

The Bar Chart represents numeric values in an underlying dataset. It is often configured to display as a category chart. A category chart is a chart whose X-values are categories (strings, names, groupings, etc) rather than numeric values (numbers, dates).

Like most chart components (other than the Easy Chart), the Data property drives the chart. The first column in the Data dataset defines the names of the categories. The rest of the columns define the values for each of the series (if there is more than one series per category), and thus should be numeric.

> **TIP**
>
> If the columns define the categories and rows define the series for your data, then set the Extract Order to "By Column".

> **NOTE**
>
> You can bring up a context menu for this component when right-clicking on it either in the Designer's Preview Mode or in a Vision Client. See the Charting - Right Click Menu page for

more details.

Properties

Name	Description	Property Type	Scripting	Category
Bar Label Color	The color for the bar labels. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.barLabelColor	Axes
Bar Label Font	The font for the bar labels.	Font	.barLabelFont	Axes
Bar Label Offset	The offset between the bar and the bar label.	double	.barLabelOffset	Axes
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Category Axis Label	The label for the category axis.	String	.categoryLabel	Axes

Name	Description	Property Type	Scripting	Category
Category Axis Label Angle	The angle for the value axis' labels.	int	.catAxisLabelPosition	Axes
Category Axis Label Color	The color for the category axis label. See **Color Selector**.	Color	.catAxisLabelColor	Axes
Category Axis Label Font	The font for the category axis label.	Font	.catAxisLabelFont	Axes
Category Axis Lower Margin	The lower margin, as a percentage, of the category axis.	double	.catAxisLowerMargin	Axes
Category Axis Tick Color	The color for the category axis' ticks. See **Color Selector**.	Color	.catAxisTickColor	Axes
Category Axis Tick Font	The font for the category axis' ticks.	Font	.catAxisTickFont	Axes
Category Axis Upper Margin	The upper margin, as a percentage, of the category axis.	double	.catAxisUpperMargin	Axes
Category Margin	The margin between categories as a fraction of the total space.	double	.categoryMargin	Appearance

Name	Description	Property Type	Scripting	Category
Chart Title	An optional title that will appear at the top of the chart.	String	.title	Appearance
Chart Type	Controls how the bar chart is displayed.	int	.rendererType	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Data	The data driving the chart.	Dataset	.data	Data
Extract Order	Controls whether the first row defines the categories or the series.	int	.extractOrder	Data
Foreground Transparency	The transparency of the bars (useful for 3D bars). Valid values are between 0 (0% opacity) and 1 (100% opacity).	float	.foregroundAlpha	Appearance

Name	Description	Property Type	Scripting	Category
Gradient bars?	If true, bars will be painted with a gradient 'shine'.	boolean	.gradient	Appearance
Item Margin	The margin between bars in a category as a fraction.	double	.itemMargin	Appearance
Labels?	Always display labels?	boolean	.labels	Appearance
Legend Font	The font for the legend items.	Font	.legendFont	Axes
Legend?	If true, show a legend for the chart.	boolean	.legend	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Plot Background	The background color for the plot.	Color	.plotBackground	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Series Colors	The sequence of colors used for	Color[]	.seriesColors	Appearance

Name	Description	Property Type	Scripting	Category
	series in the bar chart. See **Color Selector**.			
Shadows?	If true, bars will have a drop-shadow beneath them.	boolean	.shadows	Appearance
Title Font	The font for the chart's title.	Font	.titleFont	Axes
Tooltips?	If true, show tooltips.	boolean	.tooltips	Behavior
Value Axis Auto-Range	If true, the value axis range will be determined automatically. If false, the specified upper and lower bounds will be used.	boolean	.valAxisAutoRange	Axes
Value Axis Label	The label for the value axis	String	.valueLabel	Axes
Value Axis Label Color	The color for the value axis label. See **Color Selector**.	Color	.valAxisLabelColor	Axes
Value Axis Label Font	The font for the value axis label.	Font	.valAxisLabelFont	Axes
Value Axis Lower Bound	The lower bound of the value axis. Used only when	double	.valAxisLowerBound	Axes

Name	Description	Property Type	Scripting	Category
	auto-range is false.			
Value Axis Tick Color	The color for the value axis' ticks. See **Color Selector**.	Color	.valAxisTickColor	Axes
Value Axis Tick Font	The font for the value axis' ticks.	Font	.valAxisTickFont	Axes
Value Axis Upper Bound	The upper bound of the value axis. Used only when auto-range is false.	double	.valAxisUpperBound	Axes
Value Axis Upper Margin	The upper margin, as a percentage, of the value axis. Only used when auto-range is true.	double	.valAxisUpperMargin	Axes
Vertical	Sets the orientation of the chart to vertical (true) or horizontal(false)	boolean	.vertical	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

See the **Vision - Bar Chart Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

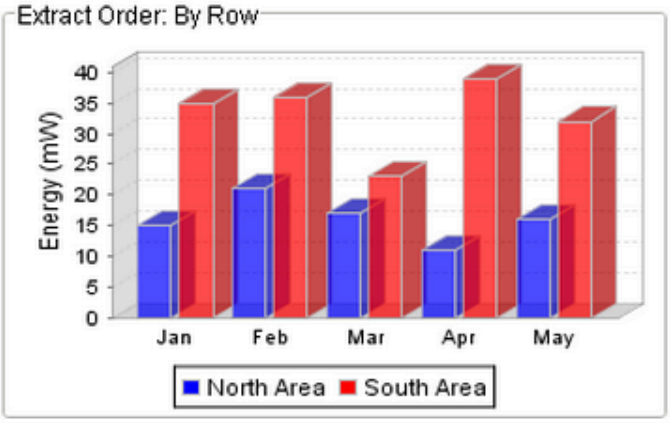
Customizers

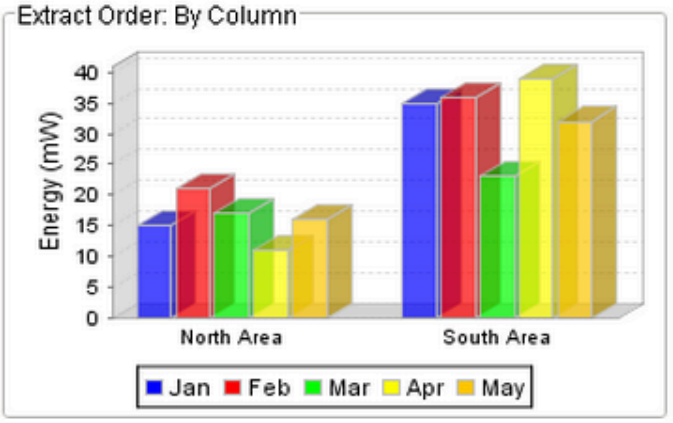
- **Vision Component Customizers**

Examples

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

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





Vision - Bar Chart Scripting Functions

This page details the various component and extension functions available for **Vision's Bar Chart** component.

Component Functions

This component does not have component functions associated with it.

Extension Functions

configureChart

- Description
  - Provides an opportunity to perform further chart configuration via scripting.
- Parameters
  - Component **self**- A reference to the component that is invoking this function.
  - JFreeChart **chart**- A JFreeChart object. Refer to the **JFreeChart documentation** for API details.
- Return
  - None

getBarColor

- Description
  - Provides a chance to override the color of each bar. Can be used to have bar colors changed based upon bar value. Returning the value None will use the default bar color for the series.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.

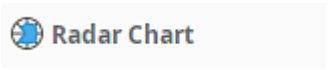
- int **series** - The series index for this bar.
- int **category** - The category index for this bar.
- int **value** - The value (a number) of this bar.
- Color **defaultColor** - The color that the bar would be if this function wasn't invoked.

- Return
  - Color

Vision - Radar Chart



Component Palette Icon:



Description

Radar charts, also known as web charts, spider charts, spider plots, and a few other names, display a dataset as a two dimensional polygon. The plot is arranged as a set of spokes with equal angles between them. Each spoke represents a value axis for the variable it corresponds to. Each dataset is then drawn as a connected polygon, where the points of the polygon are arranged on the spokes according to their value. Each row of the dataset has a minimum and maximum column -- these values are used to determine the scale of the spoke for that variable, with the midpoint representing the desired value.

The intended use of radar plots is to display realtime information in such a way that outliers can be quickly identified. This can be an efficient way to convey if a process is running on-spec or off-spec at a glance.

The radar chart gets its data from a dataset. Each row in the dataset will become a single variable (spoke) on the chart. The dataset must have a columns labeled "Value", "Min", and "Max"; other columns will be ignored. To display realtime data on a radar chart, you can use a cell-update binding to bind individual values to tag values. You can also drop tags onto a radar chart, with the EngMin binding to min and EngMax binding to max. If there are no existing cell-update bindings, the tags will replace existing data, otherwise the tags will be added to the end of the dataset. Alternatively, you can have realtime information stored by a transaction group to a database table, and drive the radar chart's dataset with a query binding.

Refer to **Radar Chart** to learn more.

Properties

Name	Description	Property Type	Scripting	Category
Actual Fill Color	Fill color for the actual polygon. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.actualFillColor	Appearance
Actual Stroke Color	Stroke color for the actual polygon. See Color Selector.	Color	.actualStrokeColor	Appearance
Actual Stroke Width	Stroke width for the actual polygon.	float	.actualStrokeWidth	Appearance
Background Color	The background color of the component. See Color Selector.	Color	.background	Appearance
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common

Name	Description	Property Type	Scripting	Category
Border Inset	The amount of area that the chart should be inset from the component bounds.	double	.borderInset	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Data	Contains the datapoints for the radar plot. Each row represents a spoke and point on the polygon.	Dataset	.data	Data
Desired Fill Color	Fill color for the desired polygon. See Color Selector.	Color	.desiredFillColor	Appearance
Desired Stroke Color	Stroke color for the desired polygon. See Color Selector.	Color	.desiredStrokeColor	Appearance
Desired Stroke Width	Stroke width for the desired polygon.	float	.desiredStrokeWidth	Appearance
Name	The name of this component.	String	.name	Common

Name	Description	Property Type	Scripting	Category
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Show Desired Shape	Display the desired shape on the chart.	boolean	.showDesiredShape	Appearance
Spoke Color	The color to use for the chart's spokes and exterior ring. See Color Selector.	Color	.foreground	Appearance
Spoke Width	The line width for the chart's spokes and exterior ring.	float	.strokeWidth	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

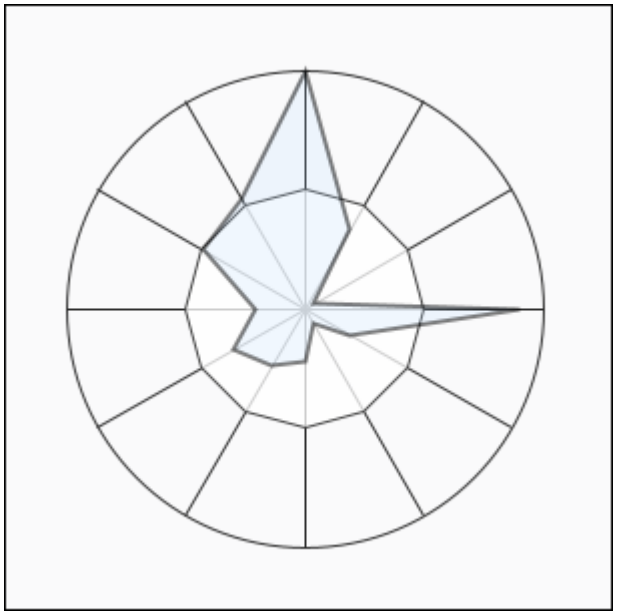
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the **Component Events** page

Customizers

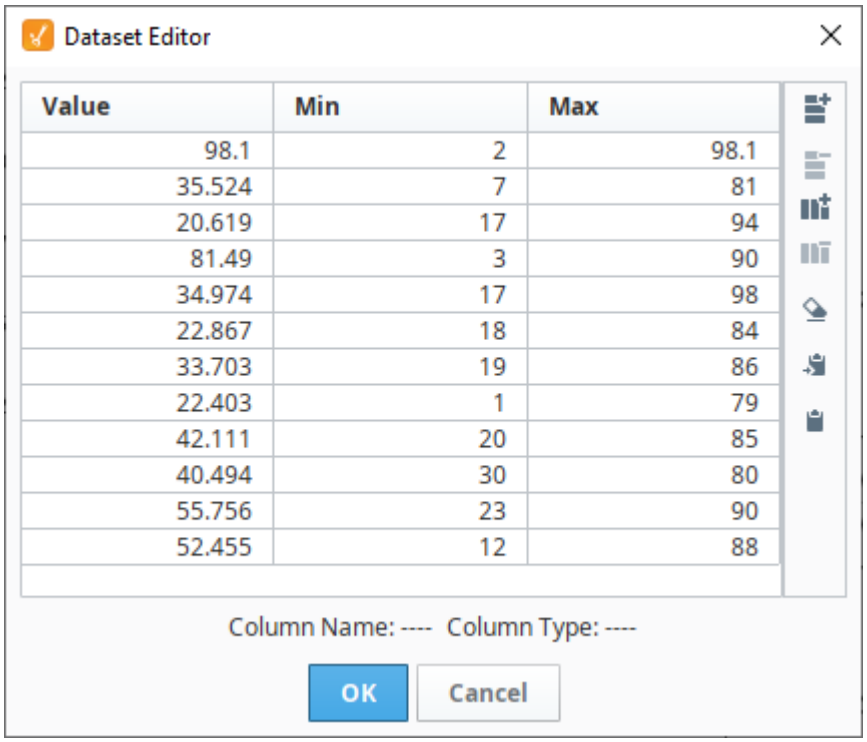
- **Vision Component Customizers**
- **Style Customizer**

Examples

Radar Charts display realtime information in such a way that outliers can be quickly identified. In this example, the Radar Chart plotted the values forming a polygon using the raw data in the code block below. You can quickly see where the process is out-of-spec and compare the values to where they should be.



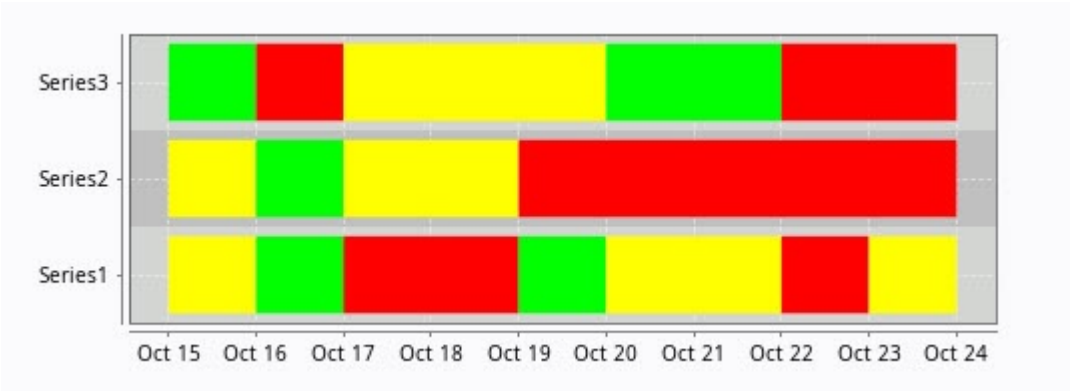
Radar Chart - Dataset Editor



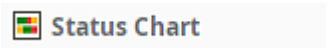
Radar Chart - Raw Data

```
"#TYPES"
"D","D","D"
"#ROWS","12"
"98.09962923575328","2.0","98.09962923575328"
"35.524092312648314","7.0","81.0"
"20.619468859704142","17.0","94.0"
"81.49014792489209","3.0","90.0"
"34.97383734960057","17.0","98.0"
"22.866686267453773","18.0","84.0"
"33.70266314329313","19.0","86.0"
"22.402620699908937","1.0","79.0"
"42.111234986669811","20.0","85.0"
"40.494873208734567","30.0","80.0"
"55.756456098723458","23.0","90.0"
"52.455123456944321","12.0","88.0"
```

Vision - Status Chart



Component Palette Icon:



Description

The Status Chart component allows you to visualize the status of one or more discrete datapoints over a time range. The X-axis is always a timeseries axis, and the Y-axis is a category axis, with one entry per data series. The chart is populated with a single dataset, the first column of which must be a datetime column.

Wide vs Tall Datasets

In Wide format, all of the columns but the first must be numeric. These "series" columns' headers will be used as the names on the y-axis. In Tall format, there should be exactly 3 columns. The first is the timestamp, the second is the series name, and the third is the value. For example:

Wide Format

t_stamp	Valve1	Valve2
2010-01-13 8:00:00	0	2
2010-01-13 8:02:00	0	2
2010-01-13 8:04:00	1	2
2010-01-13 8:06:00	1	1
2010-01-13 8:08:00	0	1

Tall Format

t_stamp	Name	Value
2010-01-13 8:00:00	Valve1	0
2010-01-13 8:00:00	Valve2	2
2010-01-13 8:02:00	Valve1	0
2010-01-13 8:02:00	Valve2	2
2010-01-13 8:04:00	Valve1	1
2010-01-13 8:04:00	Valve2	2
2010-01-13 8:06:00	Valve1	1
2010-01-13 8:06:00	Valve2	1
2010-01-13 8:08:00	Valve1	0
2010-01-13 8:08:00	Valve2	1

Color Mapping

Apart from getting the data into the series chart, the only other commonly configured option is the mapping of discrete values to colors. This is done in the Status Chart Customizer. Each named series can have its own mapping of colors, if desired. These mappings are stored in the expert-level dataset property Series Properties Data so they can be altered at runtime.

> **NOTE**
>
> You can bring up a context menu for this component when right-clicking on it either in the Designer's Preview Mode or in a Vision Client. See the **Charting - Right Click Menu** page for more details.

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), and Field Border. **Note**: The border is		.border	Common

Name	Description	Property Type	Scripting	Category
	unaffected by rotation.			
Chart Title	Title of this chart.	String	.chartTitle	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Data Format	Format of the incoming data. In "wide" format, the first column of the dataset needs to be a timestamp, and every subsequent column represents one series in the chart. In "tall" format, the first column is a timestamp,			

Name	Description	Property Type	Scripting	Category
	the second column is a series name.			
Date Style	The style to display dates in. For international support.	int	.dateStyle	Appearance
Domain Axis Color	Color used on the domain axis. See **Color Selector**.	Color	.domainAxisColor	Domain Axis
Domain Axis Font	Font used on the domain axis.	Font	.domainAxisFont	Domain Axis
Domain Axis Label	Label on the domain axis.	String	.domainAxisLabel	Domain Axis
Domain Axis Location	Location of the domain axis.	int	.domainAxisLocation	Domain Axis
Legend	Maps chart colors to descriptions.	dataset	.legend	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of	String	.toolTipText	Common

Name	Description	Property Type	Scripting	Category
	this component.			
Name	The name of this component.	String	.name	Common
Properties Loading	The number of properties currently being loaded. (Read only. Usable in bindings and scripting.)	int	.propertiesLoading	Uncategorized
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Range Axis Color	Color used on the range axis. See **Color Selector**.	Color	.rangeAxisColor	Range Axis
Range Axis Font	Font used on the range axis.	Font	.rangeAxisFont	Range Axis
Range Axis Label	Label on the range axis.	String	.rangeAxisLabel	Range Axis

Name	Description	Property Type	Scripting	Category
Range Axis Location	Location of the range axis.	int	.rangeAxisLocation	Range Axis
Range Axis Lower Margin	Lower margin of the range axis.	double	.rangeAxisLowerMargin	Range Axis
Range Axis Upper Margin	Upper margin of the range axis.	double	.rangeAxisUpperMargin	Range Axis
Series Data	Data about each series. Data can be in either "wide" or "tall" format.	Dataset	.data	Data
Series Properties Data	Properties for each series.	Dataset	.properties	Data
Series Spacing	Affects the amount of spacing between series. Can be between 0.0 and 1.0. The series present on this chart are given equal space to display themselves. Series	double	.seriesSpacing	Appearance

Name	Description	Property Type	Scripting	Category
	spacing is the percentage of that space that they use to do so.			
Show Domain Axis	Sets whether or not the domain axis is visible.	boolean	.domainAxisVisible	Domain Axis
Show Range Axis	Sets whether or not the range axis is visible.	boolean	.rangeAxisVisible	Range Axis
Time Style	The style to display times of day. For international support.	int	.timeStyle	Appearance
Title Color	Color of the chart title. See **Color Selector**.	Color	.titleColor	Appearance
Title Font	Font on the chart title.	Font	.titleFont	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

See the **Vision - Status Chart Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

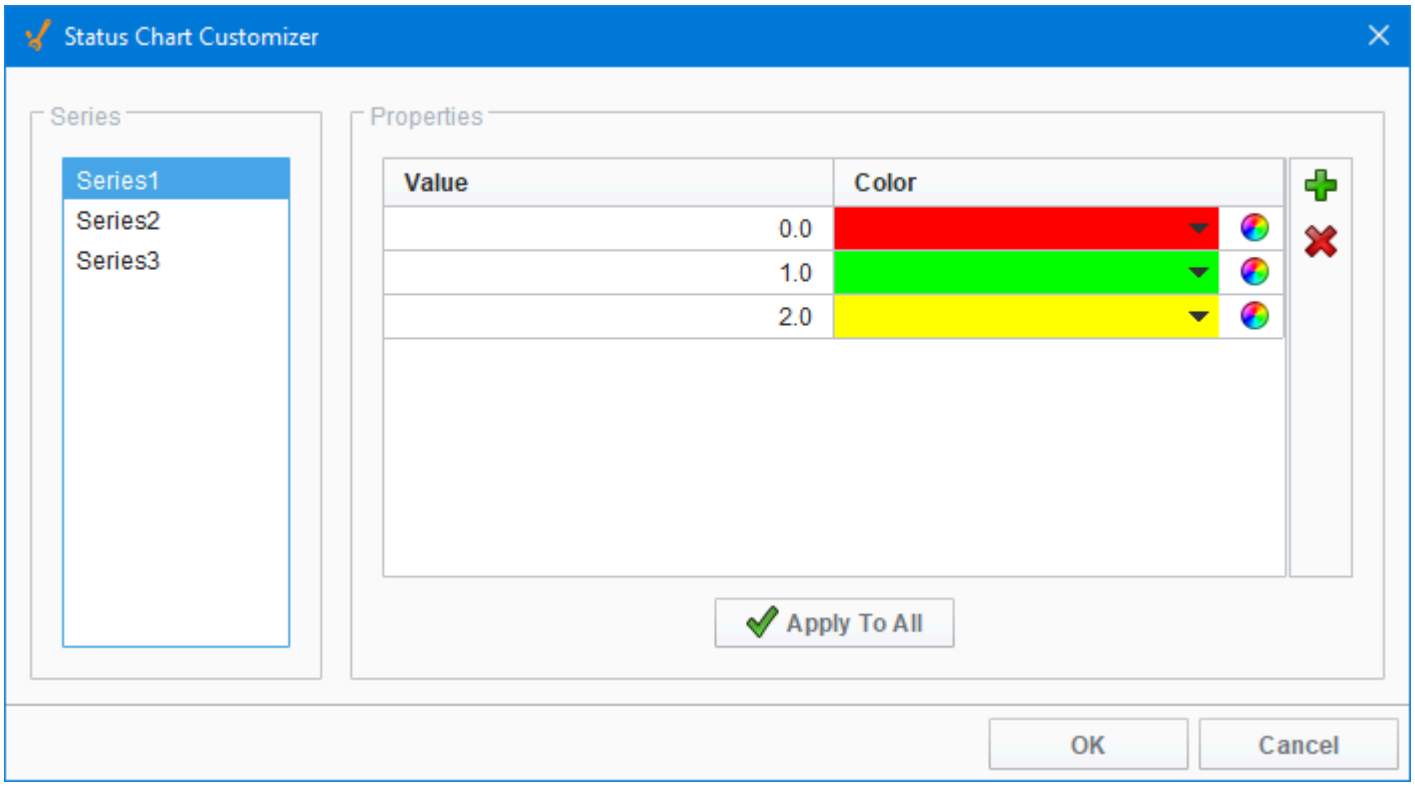
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the **Component Events** page

Customizers

The Status Chart component has its own customizer, used to set a number-to-color mapping for each series in the **Series Data** property.

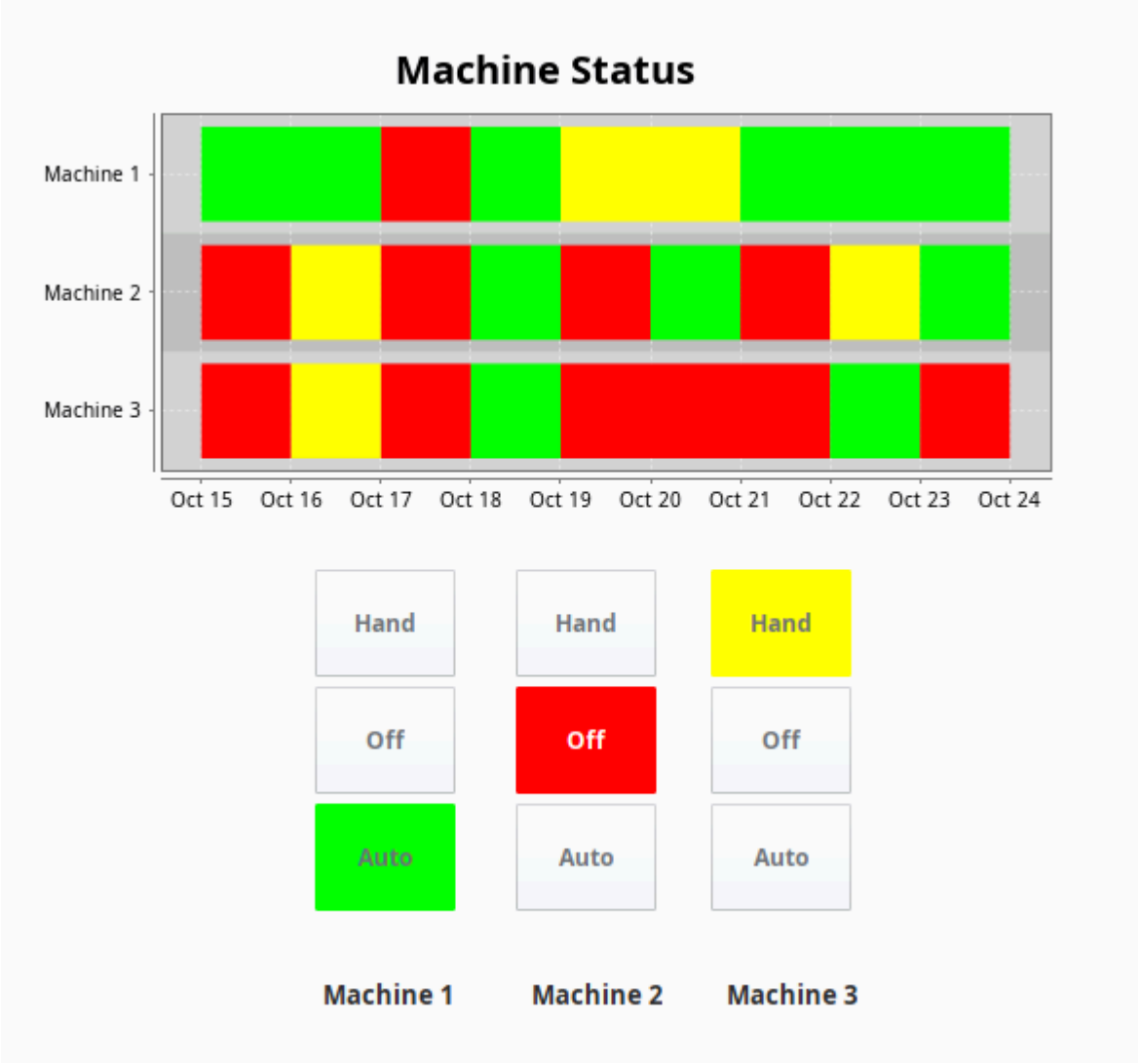
Status Chart Customizer - Property Description

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

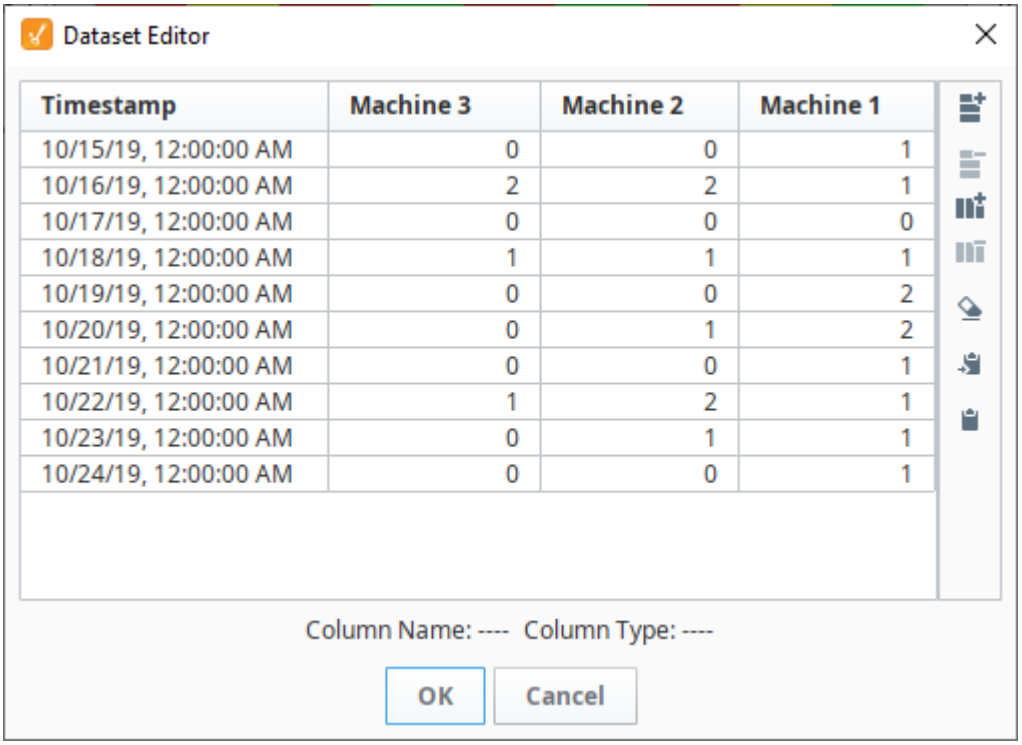


Examples

This example uses the Status Chart to display the state of each of the three machines over consecutive days using the Muli-State button. Tag History was turned on to record history HOA values. The Series Data property's dataset populates the Status Chart using a Tag History Binding. You can view the raw data by clicking on the Dataset Viewer icon to the right of the Series Data property. Each color represents a state for the machine and can be set in the Series Properties Data property. This example also has the raw data in the code block in case you want to try it for yourself.



Series Data - Dataset Viewer

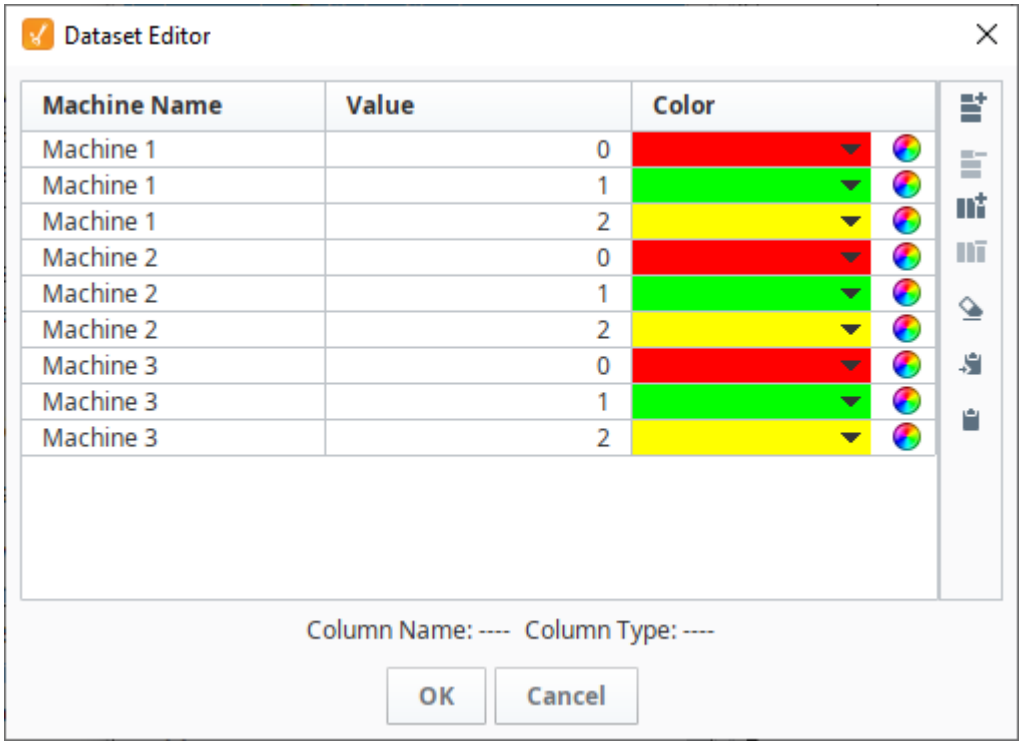


Series Raw Data

```
"#NAMES"
"Timestamp","Machine 3","Machine 2","Machine 1"
"#TYPES"
"date","I","I","I"
"#ROWS","10"
"2008-10-15 00:00:00.000","0","0","1"
"2008-10-16 00:00:00.000","2","2","1"
"2008-10-17 00:00:00.000","0","0","0"
"2008-10-18 00:00:00.000","1","1","1"
"2008-10-19 00:00:00.000","0","0","2"
"2008-10-20 00:00:00.000","0","1","2"
"2008-10-21 00:00:00.000","0","0","1"
"2008-10-22 00:00:00.000","1","2","1"
"2008-10-23 00:00:00.000","0","1","1"
"2008-10-24 00:00:00.000","0","0","1"
```

Series Properties Data - Dataset Viewer

Each machine has three states, and each of the three states (i.e., HOA) have different colors assigned representing a different state.



Series Properties Raw Data

```
"#NAMES"
"SeriesName","Value","Color"
"#TYPES"
"str","I","clr"
"#ROWS","9"
"Series1","0","color(255,0,0,255)"
"Series1","1","color(0,255,0,255)"
"Series1","2","color(255,255,0,255)"
"Series2","0","color(255,0,0,255)"
"Series2","1","color(0,255,0,255)"
"Series2","2","color(255,255,0,255)"
"Series3","0","color(255,0,0,255)"
"Series3","1","color(0,255,0,255)"
"Series3","2","color(255,255,0,255)"
```

Vision - Status Chart Scripting Functions

This page details the various component and extension functions available for **Vision's Status Chart** component.

Component Functions

This component does not have component functions associated with it.

Extension Functions

configureChart

- Description
  - Provides an opportunity to perform further chart configuration via scripting.
- Parameters
  - Component **self**- A reference to the component that is invoking this function.
  - JFreeChart **chart**- A JFreeChart object. Refer to the **JFreeChart documentation** for API details.
- Return
  - None

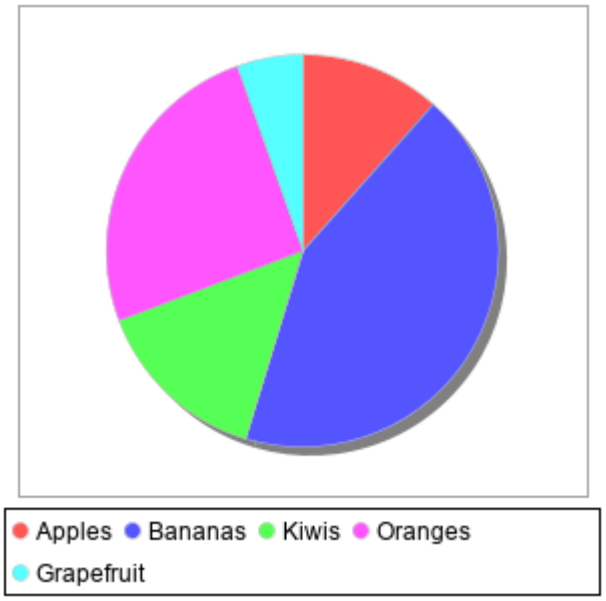
getToolTip

- Description
  - Return a formatted tool tip String
- Parameters
  - Component **self**- A reference to the component that is invoking this function.

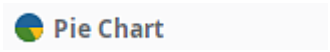
- int **seriesIndex**- The series index corresponding to the column in the series dataset.
- int **selectedTimeStamp**- The time stamp corresponding to the x value of the displayed tooltip. The time stamp is the number of seconds since the epoch.
- int **timeDiff**- The width of the current status interval measured in seconds since the epoch.
- int **seletedStatus**- The status value corresponding to the x value of the displayed tooltip.
- PyDataset **data**- The series dataset as a PyDataset.
- PyDataset **properties**- The series properties dataset as a PyDataset.
- string **defaultString**- The default tooltip string.

- Return
  - String **defaultString**

Vision - Pie Chart



Component Palette Icon:



Description

The Pie Chart component displays a familiar-looking pie chart. A Pie Chart displays a list of named items, each of which has a value that is part of a total. The total is the sum of the value of each item. The key to the Pie Chart component is the Data property, which contains the items that will be displayed as pie wedges. Typically, this dataset will be bound to a **SQL Query Binding in Vision** to pull dynamic data out of an external database.

Extract Order

Similar to other charts, the pie chart can actually accept data in two formats. You can tell the pie chart which format to use via its Extract Order property. The two extract orders are By Column or By Row. The following table shows the two styles for the data that created the pie chart in the screenshot.

By Column

Label	Value
Grapefruit	7
Apples	15
Bananas	56
Kiwis	19

By Row

Grapefruit	Apples	Bananas	Kiwis
7	15	56	19

Labels

In addition to the color-coded legend, the pie chart can annotate each wedge with a label. The format of the label is controlled via the Label Format property.

For example, the format string used in the screenshot is " 0 = 2 (3) " This is a pattern string that uses the following placeholders:

- 0 - the item label
- 1 - the item value
- 2 - the item percentage

> **NOTE**
>
> You can bring up a context menu for this component when right-clicking on it either in the Designer's Preview Mode or in a Vision Client. See the **Charting - Right Click Menu** page for more details.

Properties

Name	Description	Property Type	Scripting	Category
3D Depth Factor	The depth of a 3D pie as a factor of the chart height.	double	.depthFactor	Appearance
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Chart Title	An optional title that will appear at the top of the chart.	String	.title	Appearance
Cursor	The mouse cursor to use	int	.cursorCode	Common

Name	Description	Property Type	Scripting	Category
	when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.			
Data	The data driving the chart.	Dataset	.data	Data
Enforce Circularity?	If true, the pie cannot be an oval, even if the overall chart is.	boolean	.circular	Appearance
Extract Order	Controls whether or not a pie plot views columns as pies, or rows.	int	.extractOrder	Data

Name	Description	Property Type	Scripting	Category
Foreground Transparency	The transparency of the pie (useful for 3D pies). Valid values are between 0 (0% opacity) and 1 (100% opacity).	double	.foregroundAlpha	Appearance
Label Font	The font for labels items, if there are labels.	Font	.labelFont	Appearance
Label Format	Formatting String. '0' is the wedge name, '1' is the value, '2' is the percent.	String	.labelFormat	Appearance
Labels?	Should labels be displayed near sections?	boolean	.labels	Appearance
Legend Font	The font for legend items, if there is a legend.	Font	.legendFont	Appearance
Legend?	Should there be an item legend below the chart?	boolean	.legend	Appearance

Name	Description	Property Type	Scripting	Category
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Outline Colors	The colors to use for the pie wedge outlines. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color[]	.outlineColors	Appearance
Outline Visible	Whether to display an outline around the pie chart.	boolean	.outlineVisible	Appearance
Outline Stroke	The width for the section outline stroke.	float	.outlineStroke	Appearance

Name	Description	Property Type	Scripting	Category
Plot Background	The background color for all plots, unless they override it. See **Color Selector**.	Color	.plotBackground	Appearance
Plot Insets	The padding to use around the actual plot rendering area.	int	.plotInsets	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Rotation	Draw the wedges clockwise or counter-clockwise from the starting angle?	int	.rotation	Appearance
Section Colors	The colors to use for the pie wedge fills. See **Color Selector**.	Color[]	.sectionColors	Appearance

Name	Description	Property Type	Scripting	Category
Selected Wedge	The currently selected wedge. (Read only. Usable in bindings and scripting.)	String	.selectedData	Uncategorized
Selection Enabled?	If true, the user will be able to select wedges on the chart. The selected wedge will be highlighted, and the "selectedData" property will reflect it.	boolean	.selectionEnabled	Behavior
Selection Highlight Color	The color of the selection highlight. See **Color Selector**.	Color	.selectionHighlightColor	Appearance
Selection Highlight Width	The line width of the selection highlight.	float	.selectionHighlightWidth	Appearance
Starting Angle	The start angle to draw the pie wedges.	int	.startAngle	Appearance

Name	Description	Property Type	Scripting	Category
Style	Style of pie chart, standard, 3D, or ring.	int	.style	Appearance
Title Font	The font for the chart's title.	Font	.titleFont	Appearance
Tooltip Format	Formatting String. '0' is the wedge name, '1' is the value, '2' is the percent.	String	.tooltipFormat	Appearance
Tooltips?	Should tooltips be displayed when the mouse hovers over sections?	boolean	.tooltips	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

See the **Vision - Pie Chart Scripting Functions page** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

- Vision Component Customizers

Examples

Code Snippet

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

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

Vision - Pie Chart Scripting Functions

This page details the various component and extension functions available for **Vision's Pie Chart component**.

Component Functions

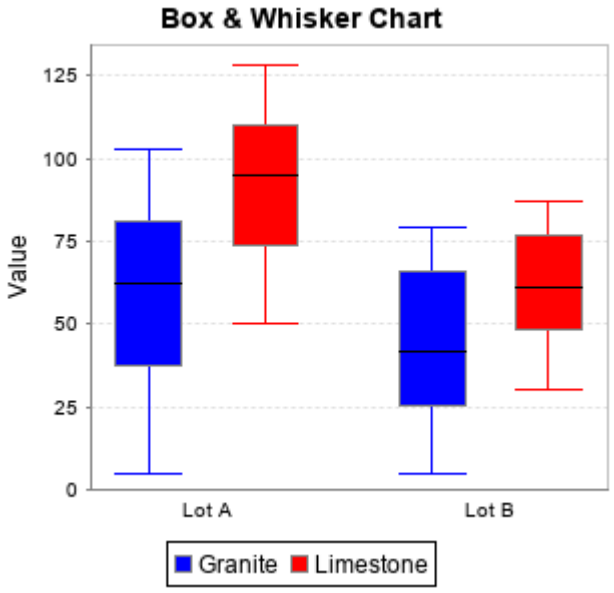
This component does not have component functions associated with it.

Extension Functions

configureChart

- Description
  - Provides an opportunity to perform further chart configuration via scripting.
- Parameters
  - Component **self**- A reference to the component that is invoking this function.
  - JFreeChart **chart**- A JFreeChart object. Refer to the **JFreeChart documentation** for API details.
- Return
  - None

Vision - Box and Whisker Chart



Component Palette Icon:

Box and Whisker Chart

Description

A Box and Whisker chart displays pertinent statistical information about sets of data. Each box represents a set of numbers. The upper and lower bounds of the box represent the 1st and 3rd quartiles. The line inside the box represents the median. The extends of the "whiskers" represent the max and min outliers. For a more detailed description, see **http://mathworld.wolfram.com/Box-and-WhiskerPlot.html**.

The configuration for setting up a box and whisker chart, like most charts, is populating the Data property. The dataset for a box and whisker chart contains sets of numbers. Each column defines a series of values, for which a "box" will be calculated. The column headers define the name for the box. You may also have an optional first column that is a String column, which can break up the series into categories.

To learn more, refer to **Box and Whisker Chart**.

> **NOTE**
>
> You can bring up a context menu for this component when right-clicking on it either in the Designer's Preview Mode or in a Vision Client. See the **Charting - Right Click Menu** page for

more details.

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), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Category Axis Title	A text label to display on the category axis.	String	.categoryAxisTitle	Appearance
Chart Title	An optional title that will appear at the top of the chart.	String	.title	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Data	The data driving the chart.	Dataset	.data	Data
Fill Boxes?	Fill the boxes with their color?	boolean	.fillBoxes	Appearance

Name	Description	Property Type	Scripting	Category
Font	Font of text on this component.	Font	.font	Appearance
Legend?	Show a legend on the chart?	boolean	.legend	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Plot Background	The background color for the plot. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.plotBackground	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Series Colors	The colors to paint each box in a series. See **Color Selector**.	Color[]	.seriesColors	Appearance
Tooltips?	Show tooltips on tasks?	boolean	.tooltips	Behavior
Value Axis Title	A text label to display on the value axis.	String	.valueAxisTitle	Appearance

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

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

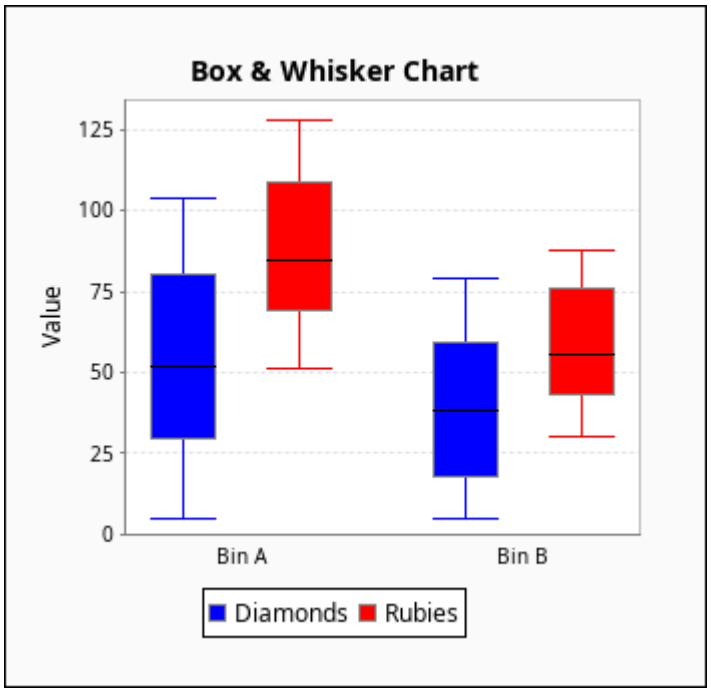
Customizers

This component does not have any custom properties.

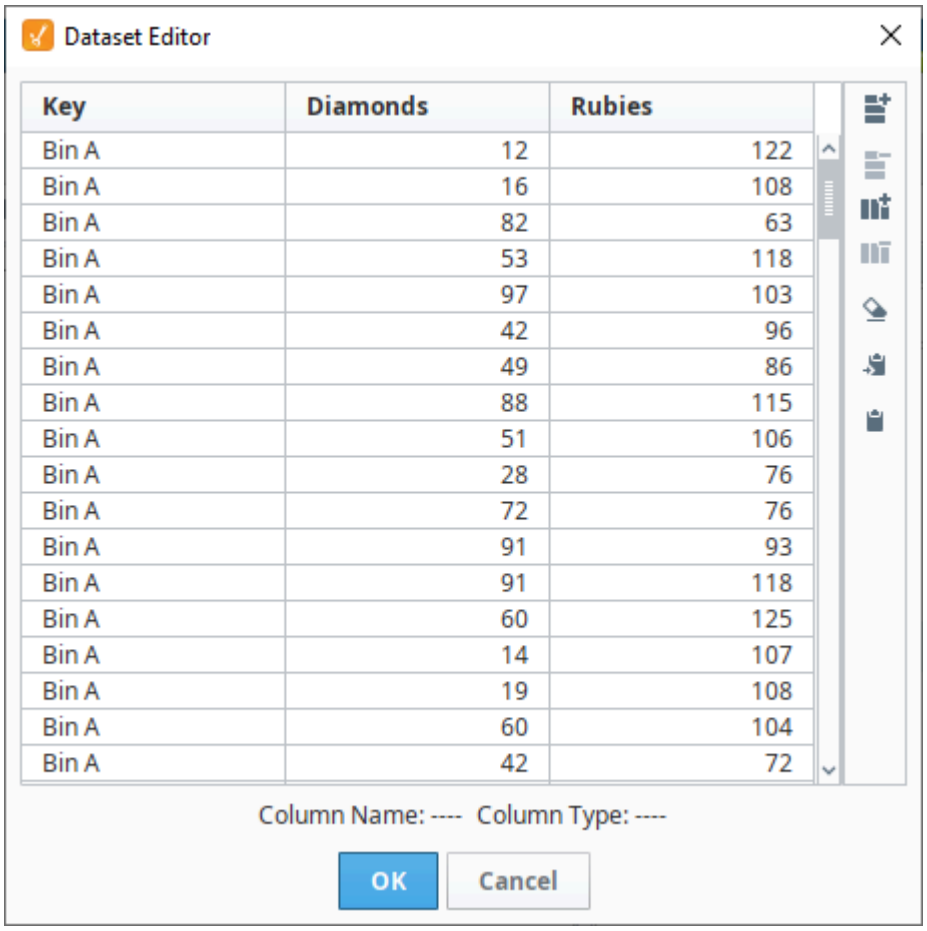
Examples

This example uses the Box & Whisker Chart to display information about two sets of data, Bin A and Bin B, and both contain Diamonds and Rubies. The Box and Whisker Chart is displaying a large amount of data as you can tell from looking at the code block below. It displays high, low, and median values which is where 50% of the data falls. The dataset contains all the raw data and calculates the upper and lower bounds of each box which are the solid colored boxes, horizontal line inside the box which represents the median value, and the whiskers which represent the minimum and maximum values which are outside the solid color boxes.

The dataset populates the chart. You can view the data in the dataset by clicking on the dataset icon. This example also has the raw data in the code block in case you want to try it for yourself.



Box and Whisker - Dataset Editor



Box and Whisker Raw Data

```
"#NAMES"
"Key","Diamonds","Rubies"
"#TYPES"
"str","I","I"
"#ROWS","200"
"Bin A","12","122"
"Bin A","16","108"
"Bin A","82","63"
"Bin A","53","118"
"Bin A","97","103"
"Bin A","42","96"
"Bin A","49","86"
"Bin A","88","115"
"Bin A","51","106"
"Bin A","28","76"
"Bin A","72","76"
"Bin A","91","93"
"Bin A","91","118"
"Bin A","60","125"
"Bin A","14","107"
"Bin A","19","108"
"Bin A","60","104"
"Bin A","42","72"
"Bin A","97","69"
"Bin A","99","69"
"Bin A","95","119"
"Bin A","76","92"
"Bin A","84","101"
"Bin A","27","99"
"Bin A","33","101"
"Bin A","12","53"
"Bin A","90","83"
"Bin A","78","61"
"Bin A","101","61"
"Bin A","50","84"
"Bin A","93","126"
"Bin A","15","85"
"Bin A","43","117"
"Bin A","37","57"
"Bin A","79","81"
"Bin A","5","53"
"Bin A","65","75"
"Bin A","94","76"
"Bin A","79","80"
"Bin A","94","97"
"Bin A","45","58"
"Bin A","104","77"
"Bin A","29","74"
```

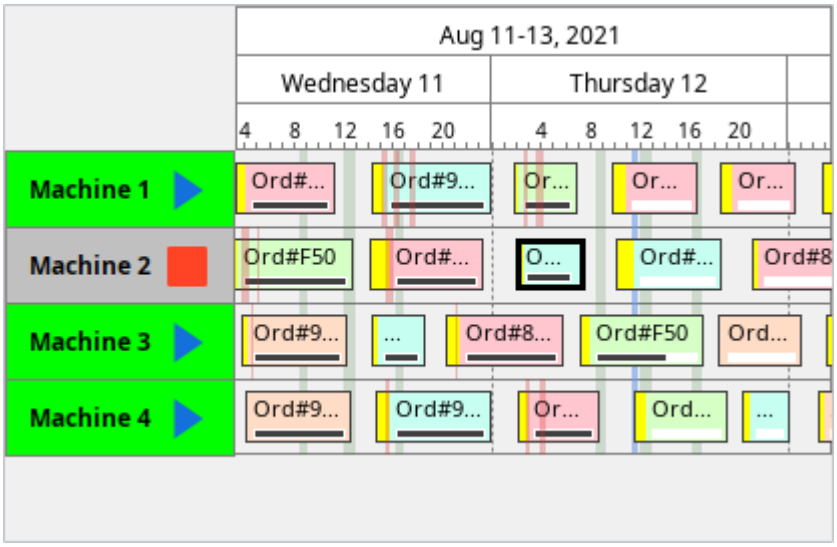
```
"Bin A","22","89"
"Bin A","20","115"
"Bin A","61","73"
"Bin A","5","70"
"Bin A","12","117"
"Bin A","36","118"
"Bin A","42","85"
"Bin A","92","87"
"Bin A","100","57"
"Bin A","42","72"
"Bin A","102","114"
"Bin A","7","90"
"Bin A","75","112"
"Bin A","36","92"
"Bin A","84","105"
"Bin A","80","69"
"Bin A","46","67"
"Bin A","48","77"
"Bin A","100","62"
"Bin A","32","72"
"Bin A","11","113"
"Bin A","23","127"
"Bin A","53","95"
"Bin A","67","108"
"Bin A","45","54"
"Bin A","47","51"
"Bin A","62","68"
"Bin A","86","72"
"Bin A","80","70"
"Bin A","77","113"
"Bin A","103","126"
"Bin A","21","57"
"Bin A","22","128"
"Bin A","11","77"
"Bin A","48","57"
"Bin A","73","118"
"Bin A","35","125"
"Bin A","57","52"
"Bin A","34","124"
"Bin A","66","68"
"Bin A","81","79"
"Bin A","43","78"
"Bin A","16","53"
"Bin A","81","109"
"Bin A","64","53"
"Bin A","94","59"
"Bin A","67","95"
"Bin A","67","57"
"Bin A","27","115"
```

```
"Bin A","18","120"
"Bin A","17","77"
"Bin A","56","87"
"Bin A","32","124"
"Bin A","30","57"
"Bin A","5","78"
"Bin A","68","82"
"Bin A","31","58"
"Bin B","66","74"
"Bin B","64","85"
"Bin B","29","86"
"Bin B","34","85"
"Bin B","16","36"
"Bin B","42","68"
"Bin B","26","33"
"Bin B","9","85"
"Bin B","27","74"
"Bin B","42","58"
"Bin B","6","72"
"Bin B","14","79"
"Bin B","40","54"
"Bin B","12","42"
"Bin B","21","34"
"Bin B","6","73"
"Bin B","46","43"
"Bin B","39","36"
"Bin B","67","42"
"Bin B","55","71"
"Bin B","42","42"
"Bin B","34","41"
"Bin B","24","54"
"Bin B","20","42"
"Bin B","66","75"
"Bin B","12","80"
"Bin B","75","84"
"Bin B","43","57"
"Bin B","62","50"
"Bin B","12","37"
"Bin B","65","32"
"Bin B","11","60"
"Bin B","5","32"
"Bin B","21","58"
"Bin B","36","53"
"Bin B","12","79"
"Bin B","37","78"
"Bin B","24","30"
"Bin B","73","87"
"Bin B","53","70"
"Bin B","70","82"
```

```
"Bin B","6","36"
"Bin B","65","72"
"Bin B","54","88"
"Bin B","10","47"
"Bin B","10","70"
"Bin B","63","41"
"Bin B","12","84"
"Bin B","77","47"
"Bin B","64","72"
"Bin B","72","84"
"Bin B","68","49"
"Bin B","23","88"
"Bin B","78","63"
"Bin B","40","57"
"Bin B","14","76"
"Bin B","7","45"
"Bin B","77","60"
"Bin B","19","86"
"Bin B","52","50"
"Bin B","64","88"
"Bin B","57","37"
"Bin B","50","69"
"Bin B","45","85"
"Bin B","27","51"
"Bin B","28","56"
"Bin B","54","54"
"Bin B","43","32"
"Bin B","11","68"
"Bin B","44","85"
"Bin B","22","55"
"Bin B","74","76"
"Bin B","51","83"
"Bin B","50","42"
"Bin B","65","77"
"Bin B","22","43"
"Bin B","34","36"
"Bin B","29","46"
"Bin B","33","51"
"Bin B","39","55"
"Bin B","17","43"
"Bin B","35","44"
"Bin B","50","31"
"Bin B","10","49"
"Bin B","78","38"
"Bin B","15","31"
"Bin B","45","78"
"Bin B","79","76"
"Bin B","22","55"
"Bin B","37","49"
```

```
"Bin B","10","50"
"Bin B","40","76"
"Bin B","40","44"
"Bin B","17","45"
"Bin B","16","87"
"Bin B","7","41"
"Bin B","67","77"
"Bin B","70","35"
"Bin B","69","52"
"Bin B","30","71"
```

Vision - Equipment Schedule



Component Palette Icon:

Equipment Schedule

Description

The Equipment Schedule component is a mix between the Status Chart, Gantt Chart, and Calendar components. It conveys a lot of information about equipment, including current status, production schedule, production status, scheduled and unexpected downtime.

The equipment schedule is powered by four datasets. Information is retrieved from the datasets by column name, case-insensitive. The order of the columns is not important. Optional columns may be omitted.

The Items Dataset

Describes the "items" or "cells" to display schedules for. Each entry in this dataset will become a row of the chart.

Name	Type	Optional	Description
ID	Any	N	The identifier for this item. May be any type, will referenced by each entry in the Scheduled Events dataset.
Label	String	N	The text to display in the header.
Foreground	Color	Y	Text color
Background	Color	Y	Background color
StatusImagePath	String	Y	A path to an image to display to the right of the header label.

The Scheduled Items Dataset

Lists the scheduled events for each item described in the "Items" dataset. Each scheduled event can have a colored lead, or change-over time, a label, a background color, and a progress.

Name	Type	Optional	Description
EventId	String	Y	An identifier for the event, used for event selection.
ItemId	Any	N	The ID of the item to correlate this event with. If no such item is found, the event won't be shown.
Label	String	N	The text to display in the event's box.
StartDate	Date	N	The start-time for the event.
EndDate	Date	N	The end-time for the event.
Foreground	Color	Y	The text color of the event.
Background	Color	Y	The background color of the event.
LeadTime	Integer	Y	Time, in seconds, to display as lead time.
LeadColor	Color	Y	The color for the lead time, if any.
PctDone	Number	Y	A value from 0 to 100 to be displayed as a progress bar, use -1 to hide progress bar.

The Downtime Dataset

Entries in this dataset will be displayed as simple colored overlays on top of the events, correlated against an item defined in the "Items" dataset.

Name	Type	Optional	Description
ItemId	Any	N	The ID of the item to correlate this downtime event with. If no such item is found, the downtime event won't be shown.
StartDate	Date	N	The start time for the downtime event.
EndDate	Date	N	The end time for the downtime event.
Color	Color	Y	The color to use, typically transparent.
Layer	Integer	Y	0 or 1, with 0 meaning that the rectangle gets painted below the events, and 1 means it will be painted above the events.

The Breaks Dataset

Entries in this dataset will be displayed as colored underlays beneath all events.

Name	Type	Optional	Description
StartDate	Date	N	The start time for the break event.
EndDate	Date	N	The end time for the break event.
Color	Color	Y	The color to use.

Properties

Name	Description	Property Type	Scripting	Category
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Break Events	Scheduled breaks, which will appear as downtime for all items.	Dataset	.breakEvents	Data
Current Time Color	The color of the current time indicator. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.nowColor	Appearance
Downtime Events	Downtime events correlated to a specific item.	Dataset	.downtimeEvents	Data

Name	Description	Property Type	Scripting	Category
Drag Enabled	Controls whether or not scheduled events can be dragged for rescheduling.	boolean	.dragEnabled	Behavior
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
End Date	The end of the time range to display.	Date	.endDate	Data
Event Border	The normal border for a scheduled event.	Border	.eventBorder	Appearance
Event Font	The font to use for the event labels.	Font	.eventFont	Appearance
Event Margin	The margin to leave visible above and below a scheduled event.	int	.scheduledEventMargin	Appearance
Header Background	The color of the background for the header timeline. See **Color Selector**.	Color	.headerBackground	Appearance
Header Font	The font of the text in the header timeline.	Font	.headerFont	Appearance

Name	Description	Property Type	Scripting	Category
Header Item Font	The font to use for the header items' labels.	Font	.itemFont	Appearance
Header Text Color	The color of the text in the header timeline. See **Color Selector**.	Color	.headerTextColor	Appearance
Items	The cells, or equipment items, to have their schedules displayed.	Dataset	.items	Data
Line Color	The color of separating lines in the schedule.	Color	.lineColor	Appearance
Name	The name of this component.	String	.name	Common
Progress Bar Background	The background color for the event progress bars. See **Color Selector**.	Color	.progressBackground	Appearance
Progress Bar Border	The border color for the event progress bars. See **Color Selector**.	Color	.progressBorder	Appearance
Progress Bar Fill	The color for 'done' portion the event progress bars. See **Color Selector**.	Color	.progressFill	Appearance

Name	Description	Property Type	Scripting	Category
Resize Enabled	Controls whether or not scheduled events resized for duration changes.	boolean	.resizeEnabled	Behavior
Row Height	The height of each event's schedule row.	int	.lineHeight	Appearance
Schedule Background	The background color of the schedule area. See **Color Selector**.	Color	.scheduleBackground	Appearance
Scheduled Events	The scheduled events for all configured items.	Dataset	.scheduledEvents	Data
Selected Event Border	The border for a selected scheduled event.	Border	.selectedEventBorder	Appearance
Selected Event ID	The ID of the selected event.	String	.selectedEvent	Data
Start Date	The beginning of the time range to display.	Date	.startDate	Data
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

See the **Vision - Equipment Schedule Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

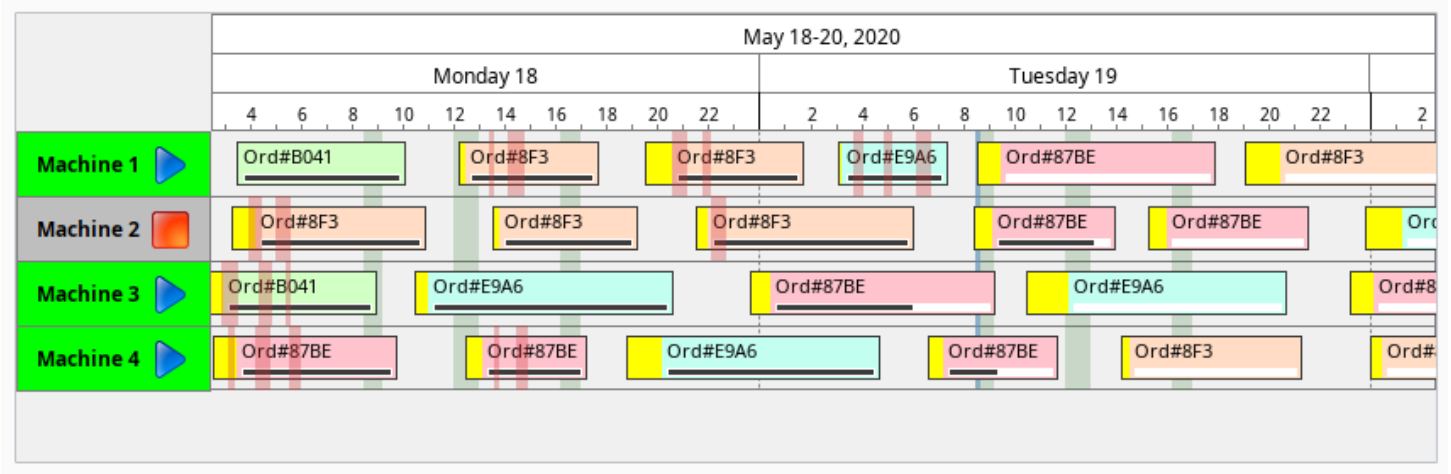
Customizers

- **Vision Component Customizers**

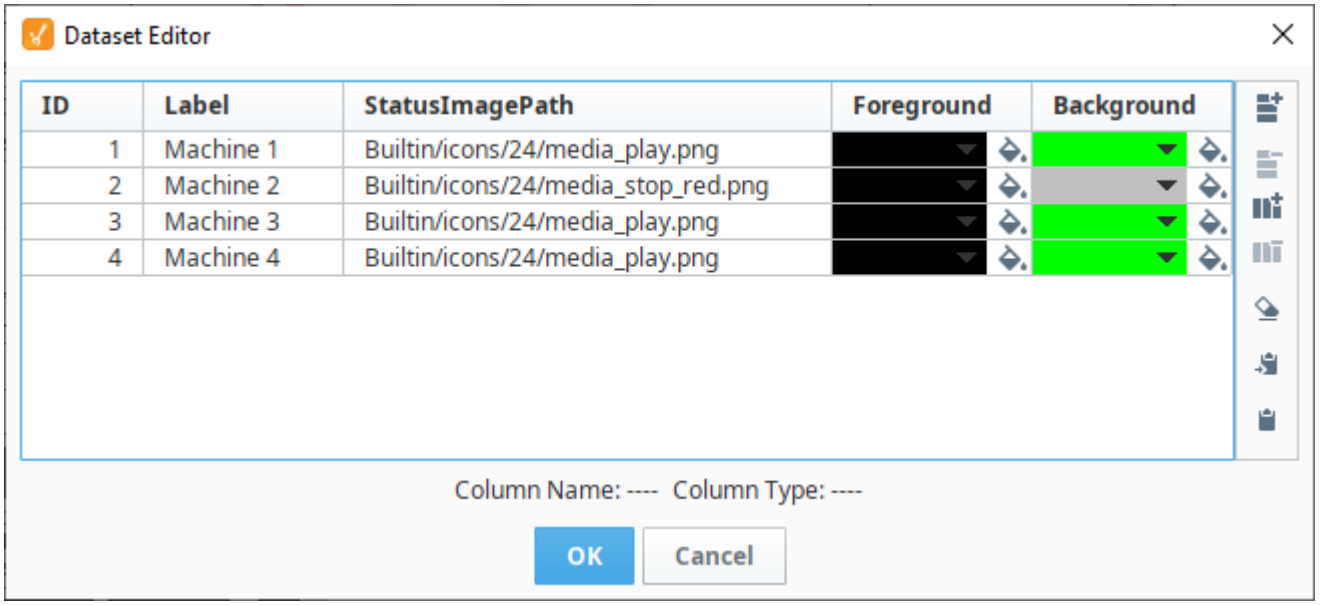
Examples

The Equipment Schedule contains a lot information about Machines 1-4 from May 18 through May 20 such as equipment status, the production schedule, production status, and schedule and unscheduled downtime. It provides a view into the status of equipment on the production floor in realtime and scheduled work planned for three days. It uses four datasets: Items, Scheduled Events, Downtime Events, and Break Events. Each dataset is shown below with it's associated raw data.

You'll notice each piece of equipment has a lead time or change-over time, a unique Order number for the run, background color and displays a progress bar. Equipment downtime entries are displayed as colored overlays on top of the events. Break events with a start and end time are displayed as colored underlays beneath the events.



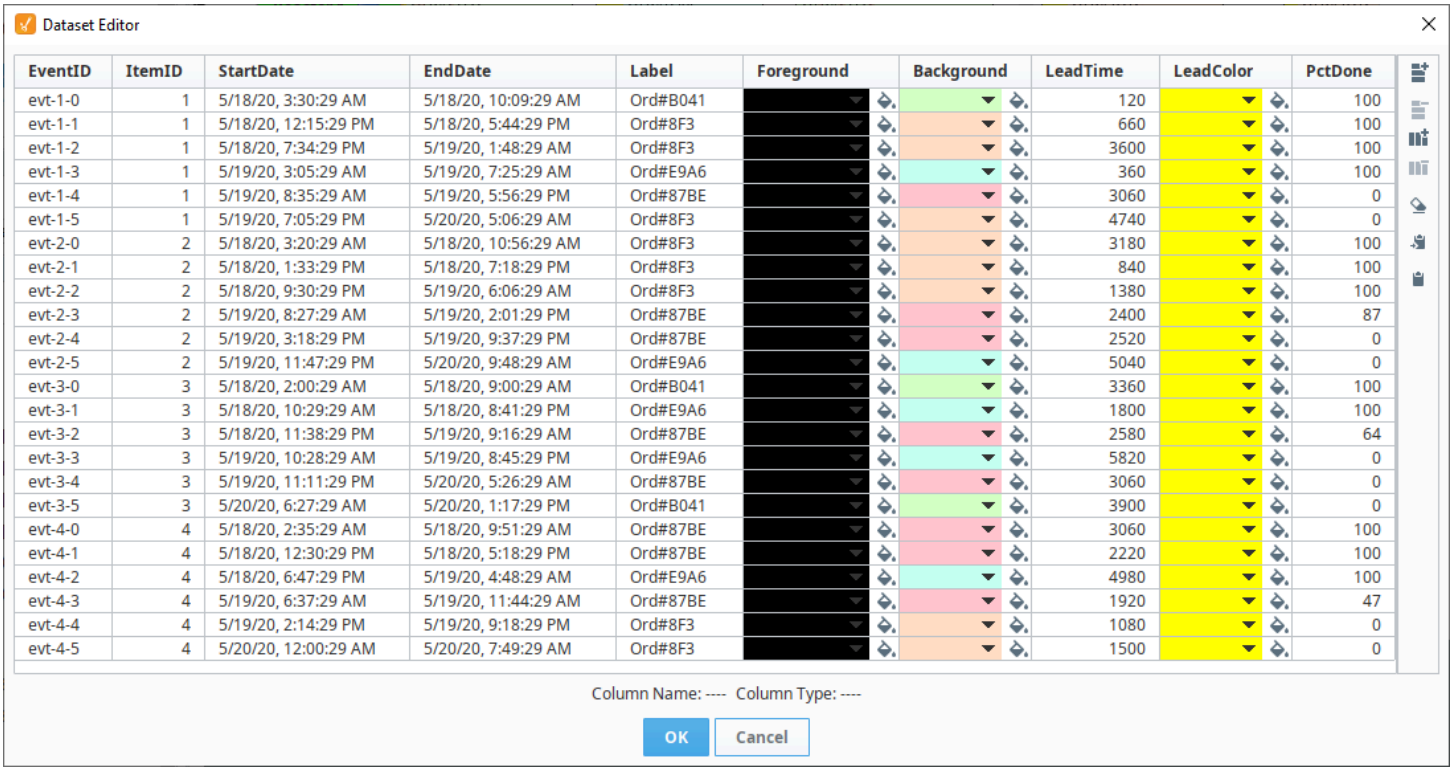
Equipment Schedule - Items Dataset



Equipment Schedule - Items Raw Data

```
"#NAMES"
"ID","Label","StatusImagePath","Foreground","Background"
"#TYPES"
"I","str","str","clr","clr"
"#ROWS","4"
"1","Machine
1","Builtin/icons/24/media_play.png","color(0,0,0,255)","color(0,255,0,255)"
"2","Machine
2","Builtin/icons/24/media_stop_red.png","color(0,0,0,255)","color(192,192,192,255)"
"3","Machine
3","Builtin/icons/24/media_play.png","color(0,0,0,255)","color(0,255,0,255)"
"4","Machine
4","Builtin/icons/24/media_play.png","color(0,0,0,255)","color(0,255,0,255)"
```

Equipment Schedule - Scheduled Items Dataset


Dataset Editor

EventID	ItemID	StartDate	EndDate	Label	Foreground	Background	LeadTime	LeadColor	PctDone
evt-1-0	1	5/18/20, 3:30:29 AM	5/18/20, 10:09:29 AM	Ord#B041			120		100
evt-1-1	1	5/18/20, 12:15:29 PM	5/18/20, 5:44:29 PM	Ord#8F3			660		100
evt-1-2	1	5/18/20, 7:34:29 PM	5/19/20, 1:48:29 AM	Ord#8F3			3600		100
evt-1-3	1	5/19/20, 3:05:29 AM	5/19/20, 7:25:29 AM	Ord#E9A6			360		100
evt-1-4	1	5/19/20, 8:35:29 AM	5/19/20, 5:56:29 PM	Ord#87BE			3060		0
evt-1-5	1	5/19/20, 7:05:29 PM	5/20/20, 5:06:29 AM	Ord#8F3			4740		0
evt-2-0	2	5/18/20, 3:20:29 AM	5/18/20, 10:56:29 AM	Ord#8F3			3180		100
evt-2-1	2	5/18/20, 1:33:29 PM	5/18/20, 7:18:29 PM	Ord#8F3			840		100
evt-2-2	2	5/18/20, 9:30:29 PM	5/19/20, 6:06:29 AM	Ord#8F3			1380		100
evt-2-3	2	5/19/20, 8:27:29 AM	5/19/20, 2:01:29 PM	Ord#87BE			2400		87
evt-2-4	2	5/19/20, 3:18:29 PM	5/19/20, 9:37:29 PM	Ord#87BE			2520		0
evt-2-5	2	5/19/20, 11:47:29 PM	5/20/20, 9:48:29 AM	Ord#E9A6			5040		0
evt-3-0	3	5/18/20, 2:00:29 AM	5/18/20, 9:00:29 AM	Ord#B041			3360		100
evt-3-1	3	5/18/20, 10:29:29 AM	5/18/20, 8:41:29 PM	Ord#E9A6			1800		100
evt-3-2	3	5/18/20, 11:38:29 PM	5/19/20, 9:16:29 AM	Ord#87BE			2580		64
evt-3-3	3	5/19/20, 10:28:29 AM	5/19/20, 8:45:29 PM	Ord#E9A6			5820		0
evt-3-4	3	5/19/20, 11:11:29 PM	5/20/20, 5:26:29 AM	Ord#87BE			3060		0
evt-3-5	3	5/20/20, 6:27:29 AM	5/20/20, 1:17:29 PM	Ord#B041			3900		0
evt-4-0	4	5/18/20, 2:35:29 AM	5/18/20, 9:51:29 AM	Ord#87BE			3060		100
evt-4-1	4	5/18/20, 12:30:29 PM	5/18/20, 5:18:29 PM	Ord#87BE			2220		100
evt-4-2	4	5/18/20, 6:47:29 PM	5/19/20, 4:48:29 AM	Ord#E9A6			4980		100
evt-4-3	4	5/19/20, 6:37:29 AM	5/19/20, 11:44:29 AM	Ord#87BE			1920		47
evt-4-4	4	5/19/20, 2:14:29 PM	5/19/20, 9:18:29 PM	Ord#8F3			1080		0
evt-4-5	4	5/20/20, 12:00:29 AM	5/20/20, 7:49:29 AM	Ord#8F3			1500		0

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

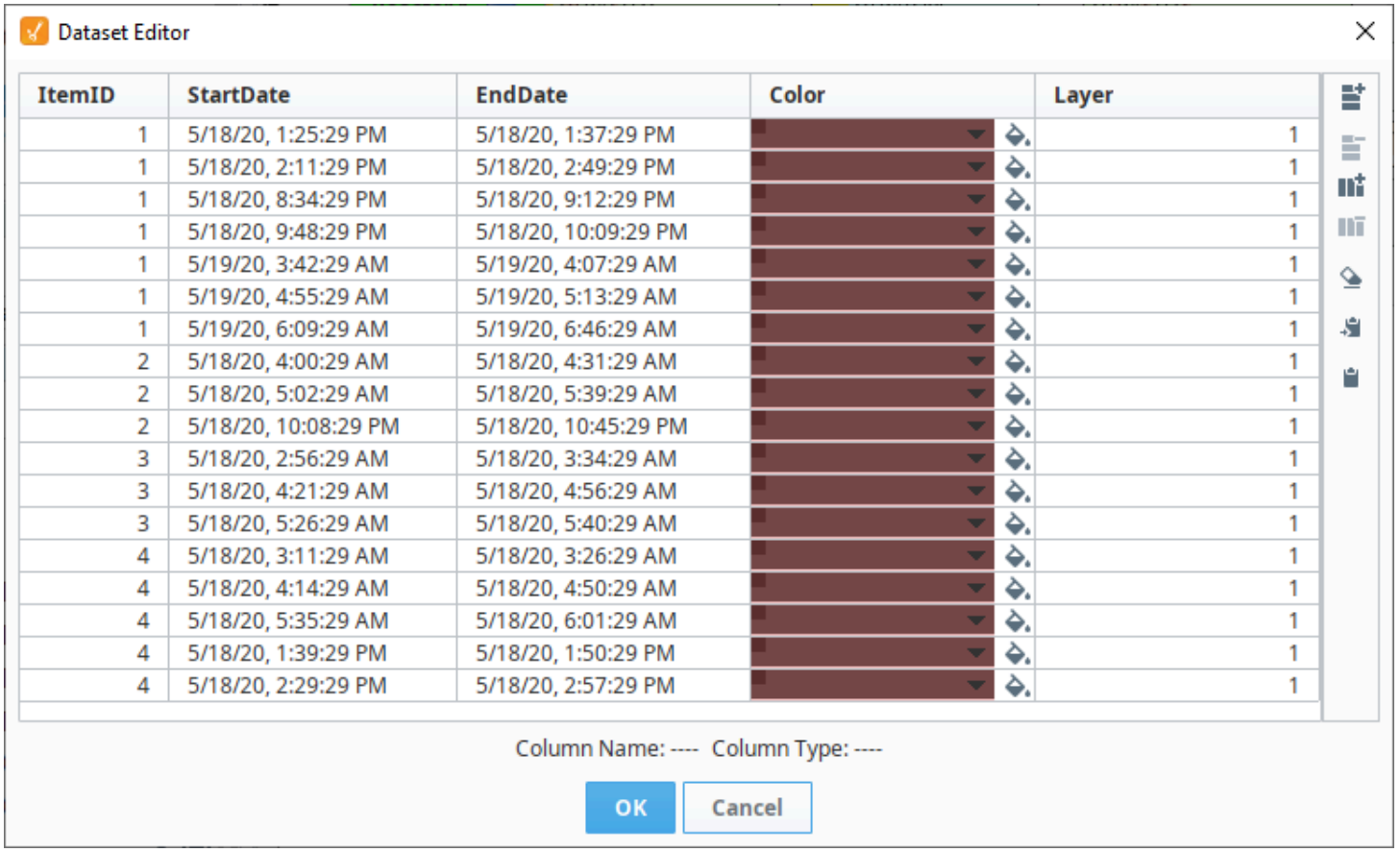
OK Cancel

Equipment Schedule - Scheduled Items Raw Data

```
"#NAMES"
"EventID","ItemID","StartDate","EndDate","Label","Foreground","Background","LeadTime"
"#TYPES"
"str","I","date","date","str","clr","clr","I","clr","D"
"#ROWS","24"
"evt-1-0","1","2020-05-18 03:30:29.002","2020-05-18
10:09:29.002","Ord#B041","color(0,0,0,255)","color(214,255,198,255)","120","color(255
"evt-1-1","1","2020-05-18 12:15:29.002","2020-05-18
17:44:29.002","Ord#8F3","color(0,0,0,255)","color(255,220,198,255)","660","color(255,
"evt-1-2","1","2020-05-18 19:34:29.002","2020-05-19
01:48:29.002","Ord#8F3","color(0,0,0,255)","color(255,220,198,255)","3600","color(255
"evt-1-3","1","2020-05-19 03:05:29.002","2020-05-19
07:25:29.002","Ord#E9A6","color(0,0,0,255)","color(198,255,242,255)","360","color(255
"evt-1-4","1","2020-05-19 08:35:29.002","2020-05-19
17:56:29.002","Ord#87BE","color(0,0,0,255)","color(255,198,207,255)","3060","color(25
"evt-1-5","1","2020-05-19 19:05:29.002","2020-05-20
05:06:29.002","Ord#8F3","color(0,0,0,255)","color(255,220,198,255)","4740","color(255
"evt-2-0","2","2020-05-18 03:20:29.002","2020-05-18
10:56:29.002","Ord#8F3","color(0,0,0,255)","color(255,220,198,255)","3180","color(255
"evt-2-1","2","2020-05-18 13:33:29.002","2020-05-18
19:18:29.002","Ord#8F3","color(0,0,0,255)","color(255,220,198,255)","840","color(255,
"evt-2-2","2","2020-05-18 21:30:29.002","2020-05-19
06:06:29.002","Ord#8F3","color(0,0,0,255)","color(255,220,198,255)","1380","color(255
"evt-2-3","2","2020-05-19 08:27:29.002","2020-05-19
14:01:29.002","Ord#87BE","color(0,0,0,255)","color(255,198,207,255)","2400","color(25
"evt-2-4","2","2020-05-19 15:18:29.002","2020-05-19
21:37:29.002","Ord#87BE","color(0,0,0,255)","color(255,198,207,255)","2520","color(25
"evt-2-5","2","2020-05-19 23:47:29.002","2020-05-20
09:48:29.002","Ord#E9A6","color(0,0,0,255)","color(198,255,242,255)","5040","color(25
"evt-3-0","3","2020-05-18 02:00:29.002","2020-05-18
09:00:29.002","Ord#B041","color(0,0,0,255)","color(214,255,198,255)","3360","color(25
"evt-3-1","3","2020-05-18 10:29:29.002","2020-05-18
20:41:29.002","Ord#E9A6","color(0,0,0,255)","color(198,255,242,255)","1800","color(25
"evt-3-2","3","2020-05-18 23:38:29.002","2020-05-19
09:16:29.002","Ord#87BE","color(0,0,0,255)","color(255,198,207,255)","2580","color(25
"evt-3-3","3","2020-05-19 10:28:29.002","2020-05-19
20:45:29.002","Ord#E9A6","color(0,0,0,255)","color(198,255,242,255)","5820","color(25
"evt-3-4","3","2020-05-19 23:11:29.002","2020-05-20
05:26:29.002","Ord#87BE","color(0,0,0,255)","color(255,198,207,255)","3060","color(25
"evt-3-5","3","2020-05-20 06:27:29.002","2020-05-20
13:17:29.002","Ord#B041","color(0,0,0,255)","color(214,255,198,255)","3900","color(25
"evt-4-0","4","2020-05-18 02:35:29.002","2020-05-18
09:51:29.002","Ord#87BE","color(0,0,0,255)","color(255,198,207,255)","3060","color(25
"evt-4-1","4","2020-05-18 12:30:29.002","2020-05-18
17:18:29.002","Ord#87BE","color(0,0,0,255)","color(255,198,207,255)","2220","color(25
"evt-4-2","4","2020-05-18 18:47:29.002","2020-05-19
04:48:29.002","Ord#E9A6","color(0,0,0,255)","color(198,255,242,255)","4980","color(25
"evt-4-3","4","2020-05-19 06:37:29.002","2020-05-19
```

```
11:44:29.002","Ord#87BE","color(0,0,0,255)","color(255,198,207,255)","1920","color(25
"evt-4-4","4","2020-05-19 14:14:29.002","2020-05-19
21:18:29.002","Ord#8F3","color(0,0,0,255)","color(255,220,198,255)","1080","color(255
"evt-4-5","4","2020-05-20 00:00:29.002","2020-05-20
07:49:29.002","Ord#8F3","color(0,0,0,255)","color(255,220,198,255)","1500","color(255
```

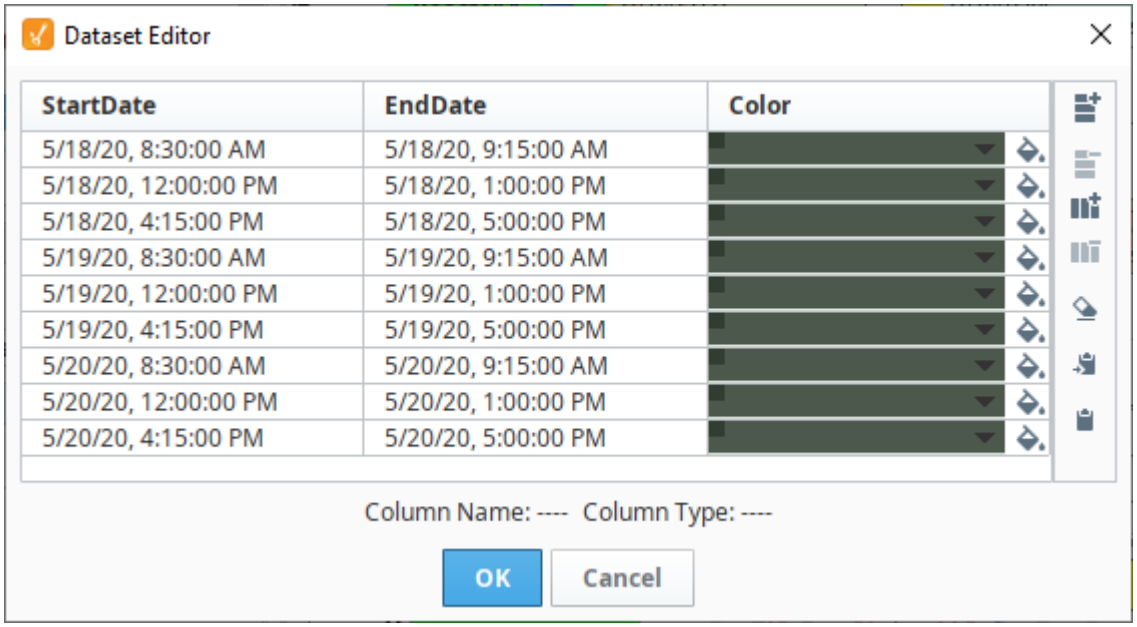

Equipment Schedule - Downtime Events Dataset



Equipment Schedule - Downtime Events Raw Data

```
"#NAMES"
"ItemID","StartDate","EndDate","Color","Layer"
"#TYPES"
"I","date","date","clr","I"
"#ROWS","18"
"1","2020-05-18 13:25:29.002","2020-05-18 13:37:29.002","color(212,49,49,75)","1"
"1","2020-05-18 14:11:29.002","2020-05-18 14:49:29.002","color(212,49,49,75)","1"
"1","2020-05-18 20:34:29.002","2020-05-18 21:12:29.002","color(212,49,49,75)","1"
"1","2020-05-18 21:48:29.002","2020-05-18 22:09:29.002","color(212,49,49,75)","1"
"1","2020-05-19 03:42:29.002","2020-05-19 04:07:29.002","color(212,49,49,75)","1"
"1","2020-05-19 04:55:29.002","2020-05-19 05:13:29.002","color(212,49,49,75)","1"
"1","2020-05-19 06:09:29.002","2020-05-19 06:46:29.002","color(212,49,49,75)","1"
"2","2020-05-18 04:00:29.002","2020-05-18 04:31:29.002","color(212,49,49,75)","1"
"2","2020-05-18 05:02:29.002","2020-05-18 05:39:29.002","color(212,49,49,75)","1"
"2","2020-05-18 22:08:29.002","2020-05-18 22:45:29.002","color(212,49,49,75)","1"
"3","2020-05-18 02:56:29.002","2020-05-18 03:34:29.002","color(212,49,49,75)","1"
"3","2020-05-18 04:21:29.002","2020-05-18 04:56:29.002","color(212,49,49,75)","1"
"3","2020-05-18 05:26:29.002","2020-05-18 05:40:29.002","color(212,49,49,75)","1"
"4","2020-05-18 03:11:29.002","2020-05-18 03:26:29.002","color(212,49,49,75)","1"
"4","2020-05-18 04:14:29.002","2020-05-18 04:50:29.002","color(212,49,49,75)","1"
"4","2020-05-18 05:35:29.002","2020-05-18 06:01:29.002","color(212,49,49,75)","1"
"4","2020-05-18 13:39:29.002","2020-05-18 13:50:29.002","color(212,49,49,75)","1"
"4","2020-05-18 14:29:29.002","2020-05-18 14:57:29.002","color(212,49,49,75)","1"
```

Equipment Schedule - Breaks Events



Equipment Schedule - Breaks Events Raw Data

```
"#NAMES"
"StartDate","EndDate","Color"
"#TYPES"
"date","date","clr"
"#ROWS","9"
"2020-05-18 08:30:00.002","2020-05-18 09:15:00.002","color(55,120,55,50)"
"2020-05-18 12:00:00.002","2020-05-18 13:00:00.002","color(55,120,55,50)"
"2020-05-18 16:15:00.002","2020-05-18 17:00:00.002","color(55,120,55,50)"
"2020-05-19 08:30:00.002","2020-05-19 09:15:00.002","color(55,120,55,50)"
"2020-05-19 12:00:00.002","2020-05-19 13:00:00.002","color(55,120,55,50)"
"2020-05-19 16:15:00.002","2020-05-19 17:00:00.002","color(55,120,55,50)"
"2020-05-20 08:30:00.002","2020-05-20 09:15:00.002","color(55,120,55,50)"
"2020-05-20 12:00:00.002","2020-05-20 13:00:00.002","color(55,120,55,50)"
"2020-05-20 16:15:00.002","2020-05-20 17:00:00.002","color(55,120,55,50)"
```

Vision - Equipment Schedule Scripting

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

Component Functions

.getDateAt(event)

- Description
  - Returns a date time representing a point in time at the mouse event position.
- Parameters
  - Event Object - A mouse event object.
- Return
  - Date - A datetime, representing a point in time on the chart where the mouse event occurred.

Extension Functions

onBackgroundDragged

- Description
  - Called when the user drags a segment on the schedule background.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - int **itemID** - The ID of the equipment item of the row where the user dragged.
  - Date **startDate** - The datetime corresponding to where the user started dragging.

- Date **endDate** - The datetime corresponding to where the user ended dragging.
- Event Object **event** - The mouse event.

- Return
  - None

onEventClicked

- Description
  - Called when the user clicks on a scheduled event. Use event.clickCount to detect double clicks.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - int **itemID** - The ID of the equipment item of the event that was clicked on.
  - int **eventId** - The ID of the event that was clicked on.
  - Event Object **event** - The mouse event.
- Return
  - None

onEventDropped

- Description
  - Called when the user drags and drops a scheduled event. It is up to this script to actually alter the underlying data to reflect the schedule change.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - int **eventId** - The ID of the scheduled event that was moved.
  - int **oldItemId** - The ID of the item this event was originally correlated against.
  - int **newItemId** - The ID of the item whose schedule the event was dropped on.

    - Date **oldStartDate** - The original starting datetime of the event.
    - Date **newStartDate** - The new starting datetime of the event.
    - Date **newEndDate** - The new ending datetime of the event.
- Return
    - None

onEventPopupTrigger

- Description
    - Called when the user right-clicks on a scheduled event. This would be the appropriate time to create and display a popup menu.
- Parameters
    - Component **self** - A reference to the component that is invoking this function.
    - int **itemId** - The ID of the equipment item of the event that was right-clicked on.
    - int **eventId** - The ID of the event that was right-clicked on.
    - Event Object **event** - The mouse event that caused the popup trigger.
- Return
    - None

onEventResized

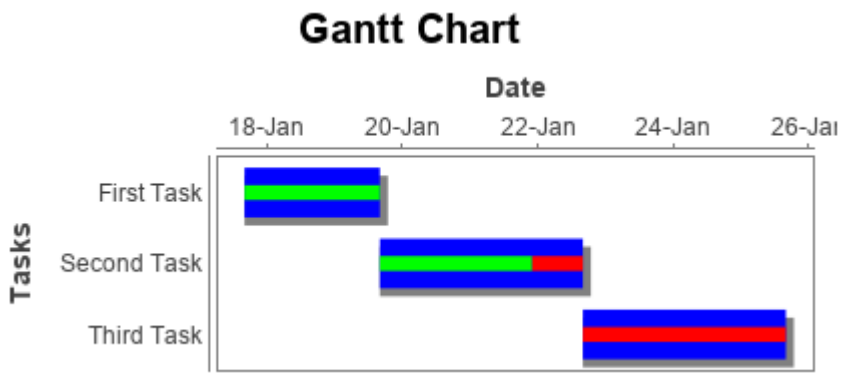
- Description
    - Called when the user drags the edge of an event to resize its time span. It is up to this script to actually alter the underlying data to reflect the schedule change.
- Parameters
    - Component **self** - A reference to the component that is invoking this function.
    - int **eventId** - The ID of the scheduled event that was resized.
    - int **itemId** - The ID of the item this event is correlated against.

    - Date **oldStartDate** - The original starting datetime of the event.
    - Date **oldEndDate** - The original ending datetime of the event.
    - Date **newStartDate** - The new starting datetime of the event.
    - Date **newEndDate** - The new ending datetime of the event.
- Return
    - None

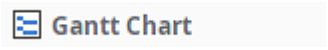
onPopupTrigger

- Description
    - Called when the user right-clicks outside of an event. This would be the appropriate time to create and display a popup menu.
- Parameters
    - Component **self** - A reference to the component that is invoking this function.
    - int **itemId** - The item ID of the equipment line that was clicked on (if any).
    - Event Object **event** - The mouse event that caused the popup trigger.
- Return
    - None

Vision - Gantt Chart



Component Palette Icon:



Description

A Gantt chart is used for task scheduling. It shows a list of named tasks, each of which have a start date, an end date, and a percentage complete. This allows an easy way to visualize tasks, workflows, and scheduling.

The Gantt chart is configured by populating its Data property. Each row of the dataset represents a task. There should be four columns: the task label, the start date, the end date, and the percentage (0-100) complete.

> **NOTE**
>
> You can bring up a context menu for this component when right-clicking on it either in the Designer's Preview Mode or in a Vision Client. See the **Charting - Right Click Menu** page for more details.

Properties

Name	Description	Property Type	Scripting	Category
Axis Font	The font for axis labels.	Font	.axisLabelFont	Appearance
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Chart Title	An optional title that will appear at the top of the chart.	String	.title	Appearance
Complete Color	The color to draw the amount completed in. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.completeColor	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait,	int	.cursorCode	Common

Name	Description	Property Type	Scripting	Category
	Hand, Move, SW Resize, or SE Resize.			
Data	The data driving the chart.	Dataset	.data	Data
Date Axis Title	A date label to display on the axis title.	String	.dateAxisTitle	Appearance
Incomplete Color	The color to draw the amount remaining to do in. See **Color Selector**.	Color	.incompleteColor	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Plot Background	The background color for the plot. See **Color Selector**.	Color	.plotBackground	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Task Axis Title	A task label to display on the Axis Title.	String	.taskAxisTitle	Appearance

Name	Description	Property Type	Scripting	Category
Task Color	The main color to draw tasks. See **Color Selector**.	Color	.taskColor	Appearance
Tick Font	The font for tick labels.	Font	.axisTickLabelFont	Appearance
Title Font	The font for the optional chart title.	Font	.titleFont	Appearance
Tooltips?	Show tooltips on tasks?	boolean	.tooltips	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

See the **Vision - Gantt Chart Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

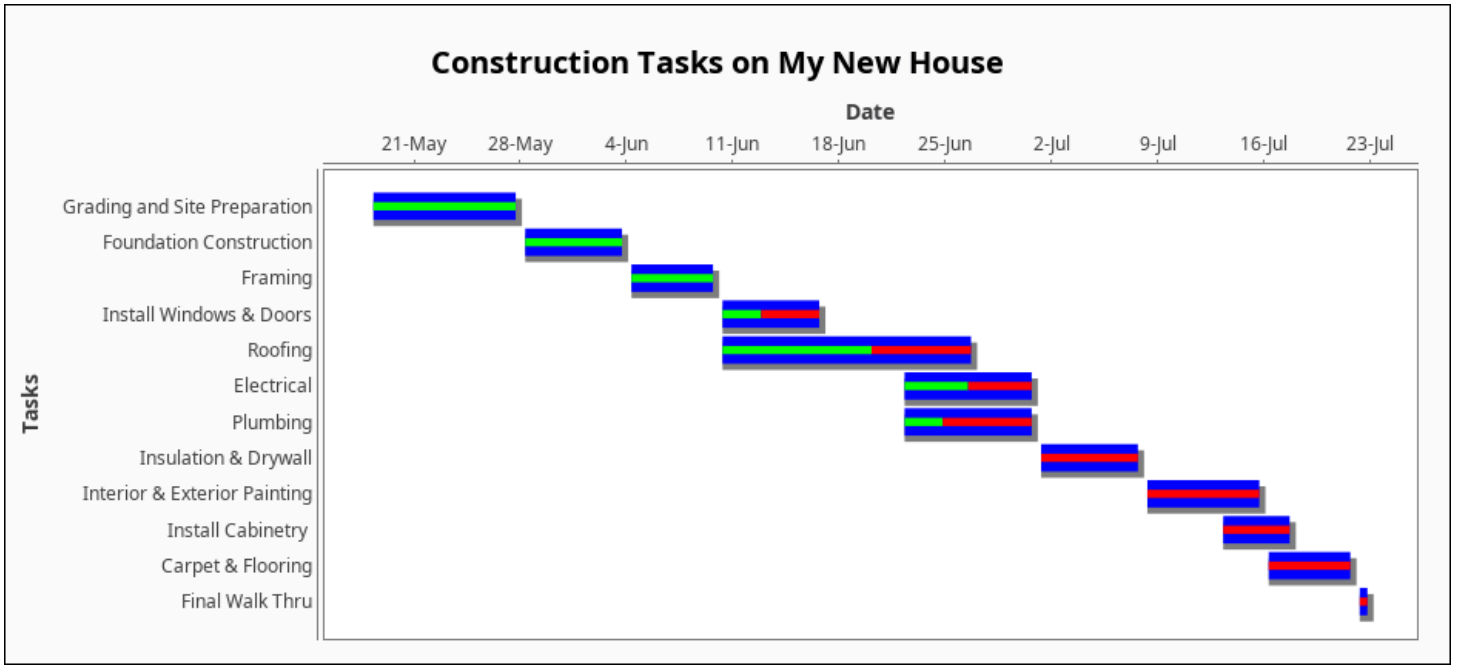
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the **Component Events** page

Customizers

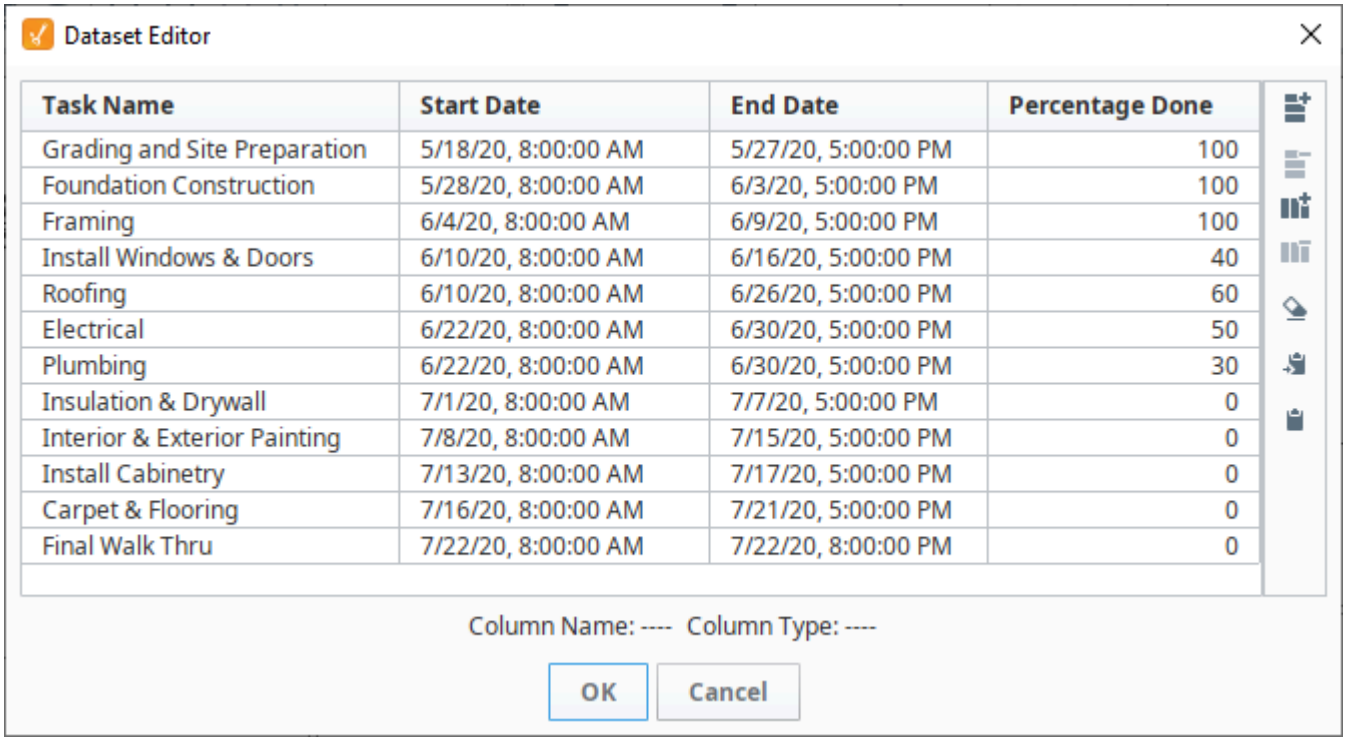
- **Vision Component Customizers**
- **Style Customizer**

Examples

This example shows the tasks associated with a construction project on a new house. It is configured by populating the Data Property. Each row of the dataset includes the start date, end date and a percentage complete for each task. It is a good tool for task scheduling and a easy way to visualize tasks, workflow and scheduling.



Gantt Chart - Dataset Editor



Gantt Chart - Raw Data

```
"#NAMES"
"Task Name","Start Date","End Date","Percentage Done"
"#TYPES"
"str","date","date","I"
"#ROWS","12"
"Grading and Site Preparation","2020-05-18 08:00:00.000","2020-05-27
17:00:00.000","100"
"Foundation Construction","2020-05-28 08:00:00.000","2020-06-03
17:00:00.000","100"
"Framing","2020-06-04 08:00:00.000","2020-06-09 17:00:00.000","100"
"Install Windows & Doors","2020-06-10 08:00:00.000","2020-06-16 17:00:00.000","40"
"Roofing","2020-06-10 08:00:00.000","2020-06-26 17:00:00.000","60"
"Electrical","2020-06-22 08:00:00.000","2020-06-30 17:00:00.000","50"
"Plumbing","2020-06-22 08:00:00.000","2020-06-30 17:00:00.000","30"
"Insulation & Drywall","2020-07-01 08:00:00.000","2020-07-07 17:00:00.000","0"
"Interior & Exterior Painting","2020-07-08 08:00:00.000","2020-07-15
17:00:00.000","0"
"Install Cabinetry ","2020-07-13 08:00:00.000","2020-07-17 17:00:00.000","0"
"Carpet & Flooring","2020-07-16 08:00:00.000","2020-07-21 17:00:00.000","0"
"Final Walk Thru","2020-07-22 08:00:00.000","2020-07-22 20:00:00.000","0"
```

Vision - Gantt Chart Scripting Functions

This page details the various component and extension functions for **Vision's Gantt Chart component**.

Component Functions

This component does not have component functions associated with it.

Extension Functions

configureChart

- Description
  - Provides an opportunity to perform further chart configuration via scripting.
- Parameters
  - Component **self**- A reference to the component that is invoking this function.
  - JFreeChart **chart**- A JFreeChart object. Refer to the **JFreeChart documentation** for API details.
- Return
  - None

Vision - Containers Palette

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

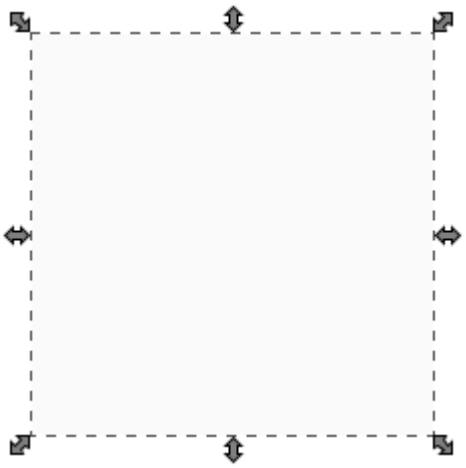
Container

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

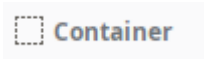
Template Repeater

Template Canvas

Vision - Container



Component Palette Icon:



Description

A Container can contain other components, including other containers. Uses for containers include:

- **Organization** - Containers can be used to group components together. These components can then 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 provide the opportunity to create a complex component that is made up of multiple other components. 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. Create a date range control that generates a SQL WHERE clause that can be used to control Charts and Tables. Create a label/button control that can be used to display datapoints, and pop up a parameterized window that displays meta-data (engineering units, physical location, notes, etc.) about that datapoint. Creating re-usable controls with Containers containing multiple components is the key to rapid application development.
- **Layout** - Containers are a great way to improve window aesthetics through borders and layout options.

> **INFO**

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

Properties

Name	Description	Property Type	Scripting	Category
Background Color	The background color of the component. 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), and Field Border. **Note**: The border is		.border	Common

Name	Description	Property Type	Scripting	Category
	unaffected by rotation.			
Combine Repaints	Set this to true for containers with many sub-components that need to redraw frequently (flashing, rotating, animating).	boolean	.combineRepaints	Behavior
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Font	Font of text on this component.	Font	.font	Appearance
Mouseover Text	The text that is displayed in the tooltip	String	.toolTipText	Common

Name	Description	Property Type	Scripting	Category
	which pops up on mouseover of this component.			
Name	The name of this component.	String	.name	Common
Opaque	If false, backgrounds are not drawn. If true, backgrounds are drawn.	boolean	.opaque	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Texture	Background texture image for this container.	String	.texturePath	Appearance
Tile Optimized	If true, this container's children should never	boolean	.optimizedDrawingEnabled	Behavior

Name	Description	Property Type	Scripting	Category
	overlap, and you'll get better painting performance.			
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

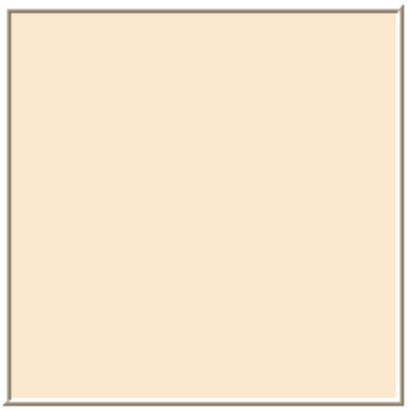
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the **Component Events** page.

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Examples

Customized Container with Border

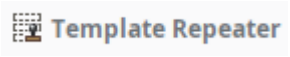


Property Name	Value
Border	Bevel (Double)
Background Color	255,232,204

Vision - Template Repeater



Component Palette Icon:



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 Parameter Name." For example, if the template has a custom property of "index" and the template repeater's Index Parameter Name is also "index," then the template will be repeated X many time with the value of X being inserted into the template's custom property called "index."
- Dataset - The template will be repeated once for each row in the "Template Parameters" dataset. The template's custom properties with the same names as the dataset's column names will assume the values of each row of the dataset.

> **INFO**
>
> An Example of configuring the Template Repeater can be found on the 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), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Flow Alignment	Alignment for "Flow" layout style. Options are: Left / Top Right / Bottom Center	int	.flowAlignment	Appearance
Flow Direction	When the layout style is flow, this property controls if the components in the	int	.flowDirection	Appearance

Name	Description	Property Type	Scripting	Category
	container flow horizontally or vertically.			
Horizontal Gap	The gap size to use for horizontal gaps.	int	.horizontalGap	Appearance
Index Parameter Name	A name of an integer parameter on the template that will be set to an index number.	String	.indexParamName	Behavior
Layout Style	Controls how the repeated template instances are laid out inside the repeater. Options are listed below: • **Vertical** - Template instances are listed vertically, top to bottom. The height of each instance will match the height of the template instance's definition. Each instance will resize horizontally to match the width of the Template Repeater. If there isn't	int	.layoutStyle	Appearance

Name	Description	Property Type	Scripting	Category
	enough space to render all instances in the Template Repeater, a scroll bar will appear on the repeater. • **Horizontal** - Templates are listed horizontally, left to right, The width of each instance will match the width of the template instance's definition. Each instance will resize vertically to match the height of the Template Repeater. If there isn't enough space to render all instances in the Template Repeater, a scroll bar will appear on the repeater. • **Flow** - Templates are placed using the			

Name	Description	Property Type	Scripting	Category
	rules specified on the Flow Direction and Flow Alignment properties. The width and height of each instance will match the dimensions on the template instance's definition. If there isn't enough space to render all instances in the Template Repeater, a scroll bar will appear on the repeater. • **BestFit** - Template positioned are determined automatically by the component. All instances will be resized so they are viewable on screen. In cases where a large number of instances are configured within the repeater, this			

Name	Description	Property Type	Scripting	Category
	mode can negatively impact readability by resizing the instances to the point where text on each instance becomes difficult to read.			
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	int	.repeatCount	Behavior

Name	Description	Property Type	Scripting	Category
	behavior is set to "Count."			
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

Scripting

See the **Vision - Template Repeater Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

- **Vision Component Customizers**

Examples

Code Snippet: getLoadedTemplates()

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

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

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

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

Vision - Template Repeater Scripting Functions

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

Component Functions

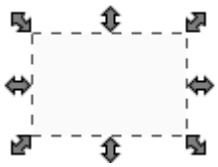
.getLoadedTemplates()

- Description
  - Returns a list of templates loaded into the Template Repeater. Properties on the components within each instance can be references by calling getComponent().
- Parameters
  - None
- Return
  - List of Templates

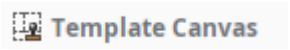
Extension Functions

This component does not have extension functions associated with it.

Vision - Template Canvas



Component Palette Icon:



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

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

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

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

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

Properties

Name	Description	Property Type	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), and Field Border. **Note**: The border is unaffected by rotation.	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	int	.scrollBehavior	Behavior

Name	Description	Property Type	Scripting	Category
	canvas will scroll in.			
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

Scripting

See the **Vision - Template Canvas Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

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

> **TEMPLATES PROPERTY**
>
> The "Templates" property, in the Property Editor, stores all the data that is entered into the customizer. New template instances can be created directly on the "Templates" property as well.

> To edit or view the dataset, click the Dataset Viewer next to the "Templates" property.

Template Canvas Customizer - Property Descriptions

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

More information on the Template Canvas Customizer can be found on the **Component Customizers** page.

Data Types and the Parameters Field

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

Data Type	Expected Format	Format Examples
Color	Colors may be entered in as either a name, or an RBG string	red 0,0,255
Date	Date objects may be entered as either a UNIX timestamp in milliseconds, or in the following notation. In all cases, quotation marks should not be added. `yyyy-MM-dd HH:mm:ss.SSS` `yyyy-MM-dd` `MM/dd/yyyy` `MM/dd/yyyy HH:mm:ss` `hh:mm:ss a` `hh:mm a` `MM/dd/yyyy hh:mm:ss a` `yyyy-MM-dd HH:mm:ss.SSS` `yyyy-MM-dd HH:mm:ss` `EEE MMM dd HH:mm:ss z yyyy`	1591374627000 2020-03-28 06:38:00:000

Examples

Code Snippet

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

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

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

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

Code Snippet - Seach by Name

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

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

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

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

#Iterate through each template instance
for template in tempInstance:

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

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

Code Snippet - Read User Input Example

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

#The code was originally design to work with the Read User Input example above,
#but can be easily modified to work with different templates.

#Reference the template canvas component, and call the getAllTemplates() method.
#This will return a list of every instance in the canvas
templateList = event.source.parent.getComponent('Template
Canvas').getAllTemplates()

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

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

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

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

Vision - Template Canvas Scripting Functions

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

Component Functions

.getAllTemplates()

- Description
  - Returns a list of the templates that comprise the template canvas.
- Parameters
  - Nothing
- Return
  - List - A list of VisionTemplate definitions. Each instance in the canvas will return it's definition's name. The names of each instance can be accessed with getInstanceName(). Individual components in each instance can accessed with getComponent().

.getTemplate(name)

- Description
  - Obtains the designated template object from the template canvas.
- Parameters
  - String **name**- The name of the template as defined by the "name" column of the dataset populating the template canvas.
- Return
  - VisionTemplate - Returns the template instance. Properties on the instance can be access by calling .propertyName

Extension Functions

initializeTemplate

- Description
  - This will be called once per template that is loaded. This is a good chance to do any custom initialization or setting parameters on the template.
- Parameters
  - Component **self** - A reference to the component invoking this function.
  - Vision Template **template** - The template. The name of the template in the dataset will be available as template.instanceName
- Return
  - None

Vision - Display Palette

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

Label

The Label component can display text, images, or both. Its text can be HTML formatted (like most components) a…

Numeric Label

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

Multi-State Indicator

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

LED Display

The LED display is a stylized numeric or alphanumeric label.

Moving Analog Indicator

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

Image

The Image component is used to display images. While other components (such as the Label) are capable of disp…

Progress Bar

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

Cylindrical Tank

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

Level Indicator

A component filled with liquid that rises and falls as the Value property changes. Can be placed behind a symbol …

Linear Scale

The Linear Scale component displays a series of tick marks and labels representing a linear range, as well as indi…

Barcode

The barcode component displays some text as a barcode.

Meter

Compass

Thermometer

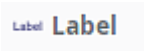
IP Camera Viewer

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

Vision - Label



Component Palette Icon:



The Label component can display text, images, or both. Its text can be HTML formatted (like most components) and can be bound to dynamic properties.

Properties

Property	Description	Property Type	Scripting	Category
Background Color	The background color of the label, if opaque is set to "true". Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.background	Appearance
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field 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,	int	.cursorCode	Common

Property	Description	Property Type	Scripting	Category
	Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.			
Disabled Image Path	The relative path of the image to be displayed when this component is not enabled.	String	.disabledPath	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Fill Background	If true, the label's background color will be drawn. If false, it will have a transparent background.	boolean	.fillBackground	Appearance
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The color of the Label's text.	Color	.foreground	Appearance
Horizontal Alignment	Determines the alignment of the label's contents along the X axis.	int	.horizontalAlignment	Layout
Horizontal Text Position	Determines the horizontal position of the	int	.horizontalTextPosition	Layout

Property	Description	Property Type	Scripting	Category
	label's text, relative to its image.			
Icon-Text Spacing	The space (in pixels) between the icon (if any) and the text (if any).	int	.iconTextGap	Appearance
Image Path	The relative path of the image.	String	.path	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Deprecated
Rotation	The angle of rotation in degrees.	int	.rotation	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Text	Text of this Label.	String	.text	Data

Property	Description	Property Type	Scripting	Category
Vertical Alignment	Determines the alignment of the label's contents along the Y axis.	int	.verticalAlignment	Layout
Vertical Text Position	Determines the vertical position of the label's text, relative to its image.	int	.verticalTextPosition	Layout
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

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

Scripting

Scripting Functions

This component does not have scripting functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

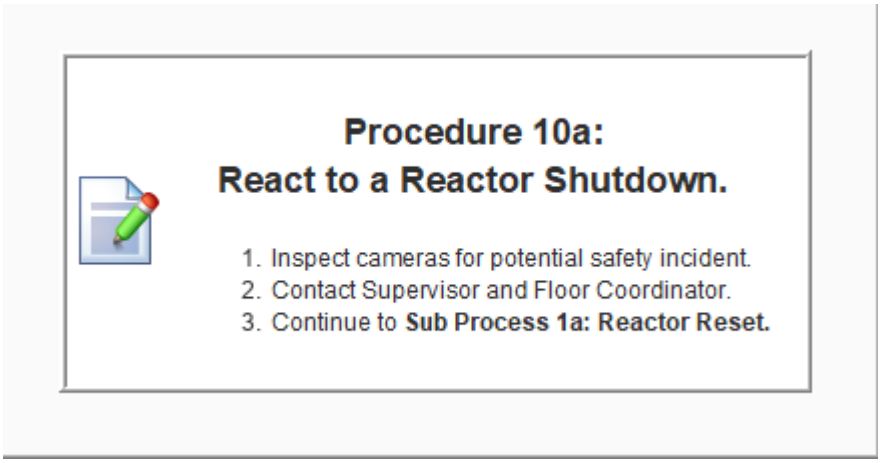
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the **Component Events** page

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Example

Stylized Label Inside a Popup Window



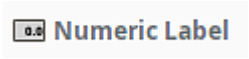
Properties

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

Vision - Numeric Label



Component Palette Icon:



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

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

Properties

Property	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), and Field 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,	int	.cursorCode	Common

Property	Description	Property Type	Scripting	Category
	Wait, Hand, Move, SW Resize, or SE Resize.			
Disabled Image Path	The relative path of the image to be displayed when this component is not enabled.	String	.disabledPath	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Fill Background	If true, the label's background color will be drawn. If false, it will have a transparent background.	boolean	.fillBackground	Appearance
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.foreground	Appearance

Property	Description	Property Type	Scripting	Category
Horizontal Alignment	Determines the alignment of the label's contents along the X axis.	int	.horizontalAlignment	Layout
Horizontal Text Position	Determines the horizontal position of the label's text, relative to its image.	int	.horizontalTextPosition	Layout
Icon-Text Spacing	The space (in pixels) between the icon (if any) and the text (if any).	int	.iconTextGap	Appearance
Image Path	The relative path of the image.	String	.path	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Number Format Pattern	The number formatting string used to format the value.	String	.pattern	Appearance

Property	Description	Property Type	Scripting	Category
Prefix	A string that will be placed before the number.	String	.prefix	Data
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Rotation	The angle of rotation in degrees.	int	.rotation	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Suffix	A string that will be placed after the number, and before the units.	String	.suffix	Data
Units	The engineering units to display after the number.	String	.units	Data
Value	The numeric value of this label.	double	.value	Data
Vertical Alignment	Determines the alignment of the label's contents along the Y axis.	int	.verticalAlignment	Layout
Vertical Text Position	Determines the vertical position of the label's text,	int	.verticalTextPosition	Layout

Property	Description	Property Type	Scripting	Category
	relative to its image.			
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

Customizers

- **Vision Component Customizers**

- Style Customizer

Examples

Numeric Label with Red Background and Percent Sign

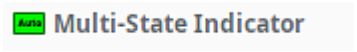


Property Name	Value
Units	%
Background Color	255,0,0

Vision - Multi-State Indicator

Off

Component Palette Icon:



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

Properties

Property	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), and Field 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,	int	.cursorCode	Common

Property	Description	Property Type	Scripting	Category
	Wait, Hand, Move, SW Resize, or SE Resize.			
Disabled Image Path	The relative path of the image to be displayed when this component is not enabled.	String	.disabledPath	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.foreground	Appearance
Horizontal Alignment	Determines the alignment of the label's contents along the X axis.	int	.horizontalAlignment	Layout

Property	Description	Property Type	Scripting	Category
Horizontal Text Position	Determines the horizontal position of the label's text, relative to its image.	int	.horizontalTextPosition	Layout
Icon-Text Spacing	The space (in pixels) between the icon (if any) and the text (if any).	int	.iconTextGap	Appearance
Image Path	The relative path of the image.	String	.path	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
State	The current state of the component.	int	.state	Data
Styles	Contains the component's	Dataset	.styles	Appearance

Property	Description	Property Type	Scripting	Category
	styles.			
Text	Text of this Label.	String	.text	Data
Vertical Alignment	Determines the alignment of the label's contents along the Y axis.	int	.verticalAlignment	Layout
Vertical Text Position	Determines the vertical position of the label's text, relative to its image.	int	.verticalTextPosition	Layout
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

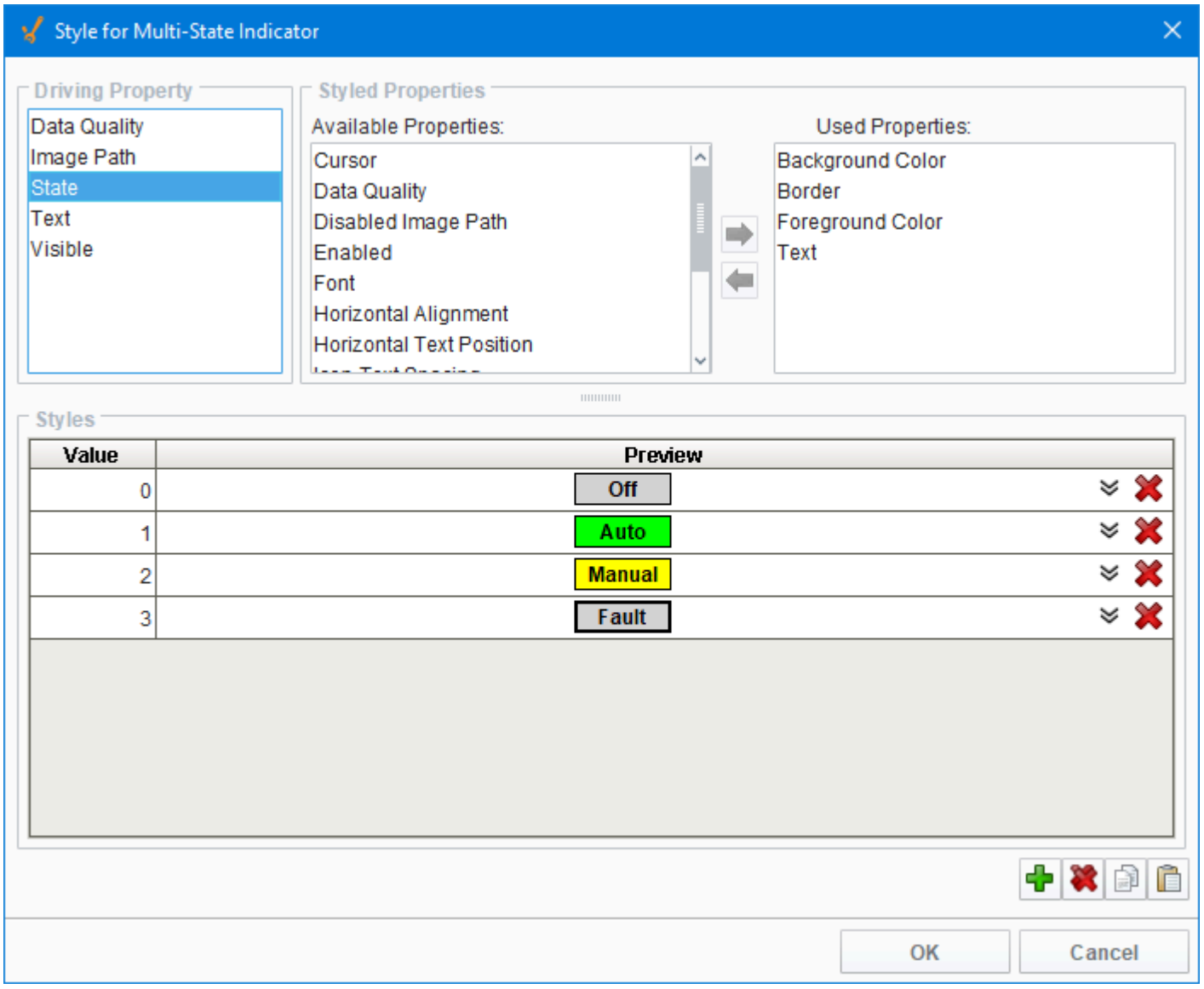
This component does not have extension functions associated with it.

Event Handlers

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

Customizers

The Style Customizer for the Multi-State Indicator includes one additional Driving Property: State. The State is represented by an integer, but the values and number of different states are customizable.



Additional Customizers

- **Vision Component Customizers**

- Style Customizer

Examples

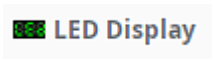


Property Name	Value
Styles	As defined by the style customizer.

Vision - LED Display



Component Palette Icon:



The LED display is a stylized numeric or alphanumeric label. It has three different visual styles: 7-segment, 14-segment, and 5x7 matrix. By default this component is in numeric mode. To display text, switch the component to alphanumeric mode.

Properties

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

Property	Description	Property Type	Scripting	Category
Horizontal Alignment	Determines the alignment of the display's contents along the X axis.	int	.horizontalAlignment	Layout
LED Lit	The color of lit LED segments. See **Color Selector**.	Color	.glyphForeground	Appearance
LED Unlit	The color of unlit LED segments. See Color Selector .	Color	.glyphBackground	Appearance
Letter Gap	The percentage of the height to be used as an inter-character spacing.	float	.gap	Layout
Margin	The margin for the interior of the display.	Insets	.margin	Layout
Mode	The mode of the display.	int	.mode	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Number Format Pattern	The number formatting string	String	.numberFormat	Behavior

Property	Description	Property Type	Scripting	Category
	used to format the value.			
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Style	The visual style of the display.	int	.style	Appearance
Styles	**Contains the component's styles.**	Dataset	.styles	Appearance
Text	The text value of the display, used when **Mode** is **Alphanumeric**.	String	.text	Data
Value	The numeric value of the display, used when **Mode** is **Numeric**.	double	.value	Data
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Examples

Custom LED Component



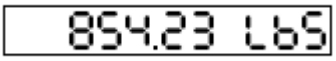
Property Name	Value
Mode	Alphanumeric
Text	ERR-28
Background Color	0,0,0
LED Lit	255,0,0
LED Unlit	0,0,0

Custom LED Component



Property Name	Value
Mode	Alphanumeric
Text	Hello World
Horizontal Alignment	Center

Custom LED Component



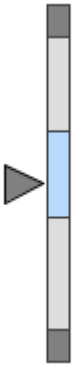
Property Name	Value
Border	Line Border
Mode	Alphanumeric
Text	852.23 lbs
Style	7 Segment
Background Color	255,255,255
LED Lit	0,0,0
LED Unlit	255,255,255

Custom LED Component

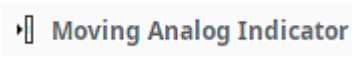


Property Name	Value
Style	5x7 Matrix
Background Color	255,255,255
Horizontal Alignment	Right

Vision - Moving Analog Indicator



Component Palette Icon:



The Moving Analog Indicator displays an analog value as an arrow pointing at a bar with segments showing the desired operating range, low and high alarm ranges, and interlock ranges.

To switch the Moving Analog Indicator between a horizontal vs vertical orientation, change the size so that it is either wide or tall. Setup of this component involves setting the ranges and binding the Process Value property to a Tag's value.

Properties

Property	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), and Field 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

Property	Description	Property Type	Scripting	Category
Desired High	The upper value of the desired operating range.	Double	.desiredHi	Data
Desired Low	The lower value of the desired operating range.	Double	.desiredLo	Data
Desired Range Color	The color of the desired range. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.desiredRangeColor	Appearance
High Alarm	The value above which is a high alarm.	Double	.hiAlarm	Data
High High Alarm	The value above which is a high-high alarm.	Double	.hihiAlarm	Data
High Interlock	The value above which an interlock	Double	.hiInterlock	Data

Property	Description	Property Type	Scripting	Category
	will be activated.			
Inactive Alarm Color	The color of inactive alarm range. See **Color Selector**.	Color	.inactiveAlarmColor	Appearance
Interlock Color	The color of the interlock range. See **Color Selector**.	Color	.interlockColor	Appearance
Level 1 Alarm Color	The color of an active level 1 alarm (Hi-Hi or Lo-Lo). See **Color Selector**.	Color	.level1AlarmColor	Appearance
Level 2 Alarm Color	The color of an active level 2 alarm (Hi or Lo). See **Color Selector**.	Color	.level2AlarmColor	Appearance
Low Alarm	The value below which is a low alarm.	Double	.loAlarm	Data
Low Interlock	The value below which an interlock will be activated.	Double	.loInterlock	Data

Property	Description	Property Type	Scripting	Category
Low Low Alarm	The value below which is a low-low alarm.	Double	.loloAlarm	Data
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Process Value	The current value of the process.	Double	.processValue	Data
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Range Fill	The background color of the range strip. See **Color Selector**.	Color	.rangeFill	Appearance
Range High	The overall high value for	double	.rangeHi	Data

Property	Description	Property Type	Scripting	Category
	the display.			
Range Low	The overall low value for the display.	double	.rangeLo	Data
Range Stroke	The stroke color for the range strip. See **Color Selector**.	Color	.rangeStroke	Appearance
Reverse Indicator	Put the indicator triangle on the other side of the track.	boolean	.reverseIndicatorLocation	Appearance
Setpoint Fill	The fill color of the setpoint indicator. See **Color Selector**.	Color	.setpointFill	Appearance
Setpoint Stroke	The stroke color of the setpoint indicator. See **Color Selector**.	Color	.setpointStroke	Appearance
Setpoint Value	The current value of the setpoint.	Double	.setpointValue	Data
Show Value	Show the current value above or	boolean	.showValue	Appearance

Property	Description	Property Type	Scripting	Category
	beneath the value indicator.			
Stroke Width	The stroke width for lines drawn.	float	.strokeWidth	Appearance
Styles	Contains the component's styles	Dataset	.styles	Appearance
Value Color	The color of the value label. See **Color Selector**.	Color	.color	Appearance
Value Font	The font for the value label.	Font	.font	Appearance
Value Format	The string format for the value, if it is shown.	String	.valueFormat	Appearance
Value Indicator Fill	The fill color of the value indicator. See **Color Selector**.	Color	.valueFill	Appearance
Value Indicator Stroke	The stroke color of the value indicator. See **Color Selector**.	Color	.valueStroke	Appearance

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

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

Customizers

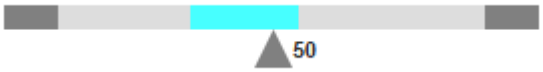
- **Vision Component Customizers**
- **Style Customizer**

Examples

Default Moving Analog Indicator Expanded Horizontally

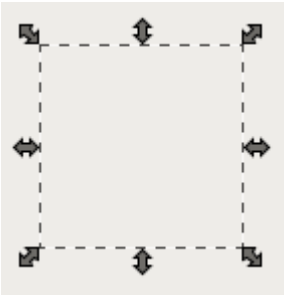


Stylized Moving Analog Indicator

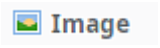


Property Name	Value
Show Value	True
Reverse Indicator	True
Stroke Width	0.0

Vision - Image



Component Palette Icon:



The Image component is used to display images. While other components (such as the **Label**) are capable of displaying images, the Image component has four additional features:

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

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

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

External Images

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

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

Properties

Property	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), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Color Swap Filter	Swap a specific color to another. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector .	boolean	.useColorSwap	Image Manipulation
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common

Property	Description	Property Type	Scripting	Category
Disabled Image Path	The relative path of the image to be displayed when this component is not enabled.	String	.disabledPath	Data
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Flip Horizontal	Flip (mirror) the image horizontally.	boolean	.flipHorizontal	Image Manipulation
Flip Vertical	Flip (mirror) the image vertically.	boolean	.flipVertical	Image Manipulation
Image Path	The relative path of the image.	String	.path	Data
Load In Background	Controls whether or not the image loading takes place on the UI thread or a background thread.	boolean	.loadInBackground	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag	QualityCode	.quality	Data

Property	Description	Property Type	Scripting	Category
	bindings on this component.			
Rotation	The angle of rotation in degrees.	int	.rotation	Image Manipulation
Stretch Height	If stretch mode is "Parameters", this will be the stretched height of the image If stretch mode is "% Bounds", this will be the percentage of the component's height.	int	.stretchHeight	Image Manipulation
Stretch Mode	Sets the stretch mode for this image.	int	.stretchMode	Image Manipulation
Stretch Width	If stretch mode is "Parameters", this will be the stretched width of the image If stretch mode is "% Bounds", this will be the percentage of the component's width.	int	.stretchWidth	Image Manipulation
Styles	Contains the component's styles.	Dataset	.styles	Appearance

Property	Description	Property Type	Scripting	Category
Swap From	If the Color Swap Filter is on, this color will be changed to the Swap To color.	Color	.swapFromColor	Image Manipulation
Swap Threshold	Threshold (0-255) for the swap from color matching. 0 is no tolerance, 255 is max tolerance.	int	.swapThreshold	Image Manipulation
Swap To	If the Color Swap Filter is on, the Swap From color will be changed to this color. See Color Selector .	Color	.swapToColor	Image Manipulation
Tint Color	If the Tint Filter is on, this is the color of the tint. See Color Selector .	Color	.tintColor	Image Manipulation
Tint Filter	Tint the entire image a color (works best with greyscale images).	boolean	.useTint	Image Manipulation
Use Cache	If false, this image will bypass the client image cache and load the image directly from the source.	boolean	.useCache	Behavior

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

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

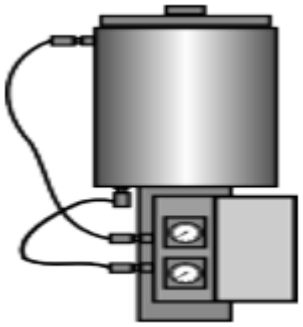
Event Handlers

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

Customizers

- **Vision Component Customizers**
- **Style Customizer**

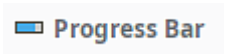
Examples



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

Vision - Progress Bar

Component Palette Icon:



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

Properties

Property	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), and Field 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

Property	Description	Property Type	Scripting	Category
Direction	Determines the direction of progress for this progress bar.	int	.direction	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See **Color Selector**.	Color	.foreground	Appearance
Horizontal?	If true, the progress bar will display horizontally, else it will display vertically. Manually resize the progress bar to display vertically.	boolean	.horizontal	Appearance
Indeterminate?	When true, the progress bar displays animation indicating that something is happening, but it will take an indeterminate amount of time	boolean	.indeterminate	Behavior

Property	Description	Property Type	Scripting	Category
Maximum	The maximum value that this progress bar will reach.	int	.maximum	Data
Minimum	The minimum value that this progress bar will reach.	int	.minimum	Data
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Show Percentage?	If true, the progress bar will display its percentage.	boolean	.stringPainted	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Text Color	The color of the text on the progress bar.	Color	.textColor	Appearance
Value	The current state of the Progress Bar.	int	.value	Data

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

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

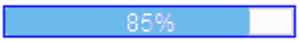
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the **Component Events** page

Customizers

- **Vision Component Customizers**
- **Style Customizer**

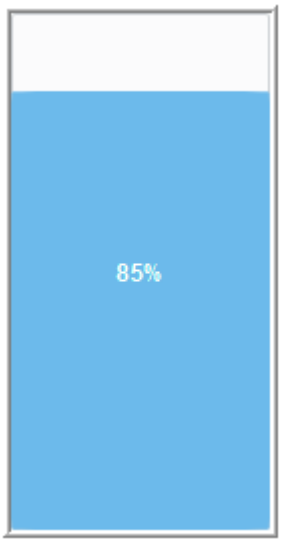
Examples

Horizontal Blue Progress Bar



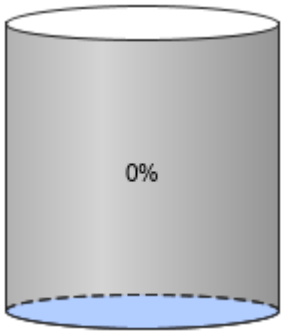
Property Name	Value
Border	Line Border
Value	85
Foreground Color	0,0,255
Horizontal?	True
Show Percentage?	True

Wide Vertical Blue Progress Bar



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

Vision - Cylindrical Tank



Component Palette Icon:



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

Properties

Property	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), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Capacity	Total capacity of tank.	double	.capacity	Data
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Font	Font of text on this component.	Font	.font	Appearance
Font Color	The color of the value and/or percentage	Color	.fontColor	Appearance

Property	Description	Property Type	Scripting	Category
	labels. See **Color Selector**.			
Foreground Color	The foreground color of the component. See **Color Selector**.	Color	.foreground	Appearance
Liquid Color	Color of the filled tank section. See **Color Selector**.	Color	.liquidColor	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Percent Format	Format string used for the percentage.	String	.percentFormat	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Rotation	The angle of rotation in degrees.	int	.rotation	Appearance
Show Percentage	Shows percentage of tank filled when enabled.	boolean	.showPercent	Appearance
Show Value	Shows numeric value, capacity, and units when enabled.	boolean	.showValue	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance

Property	Description	Property Type	Scripting	Category
Tank Color	Color of the non-filled tank section. See **Color Selector**.	Color	.tankColor	Appearance
Units	Units of measure for tank contents.	String	.units	Appearance
Value	Numeric value of tank's level.	double	.value	Data
Value Format	Format string used for the value.	String	.valueFormat	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

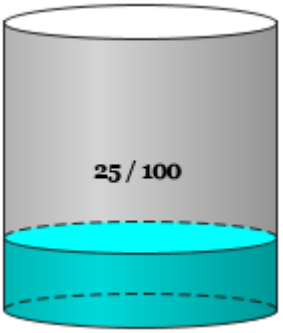
Event Handlers

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

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Examples



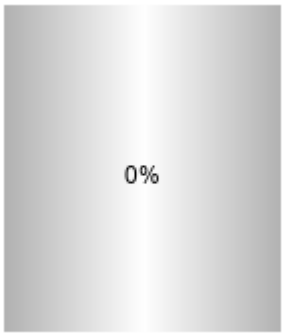
Property Name	Value
Value	25
Font	Georgia, Bold 12
Liquid Color	0,217,217
Show Value	True
Show Percentage	False

Setting value through Scripting

```
# You can set the component's value through scripting
event.source.parent.getComponent('Cylindrical Tank').value = 5.4

# Alternatively, you can use the .setValue method to set the component's value
event.source.parent.getComponent('Cylindrical Tank').setValue(5.4)
```

Vision - Level Indicator



Component Palette Icon:



A component filled with liquid that rises and falls as the Value property changes. Can be placed behind a symbol factor object that has a cutout in it.

Properties

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

Property	Description	Property Type	Scripting	Category
Font Color	The foreground color of the component. See **Color Selector**.	Color	.fontColor	Appearance
Gradient	Indicates whether the level will be drawn as a 3D gradient.	boolean	.gradient	Appearance
Liquid Waves	Indicate whether liquid waves are drawn.	boolean	.waves	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Orientation	Determines which direction the level "grows" for an increase in value.	int	.orientation	Appearance
Percent Format	Format string used for the percentage.	String	.percentFormat	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Show Percentage	Indicates whether the percentage of tank filled is displayed.	boolean	.showPercent	Appearance

Property	Description	Property Type	Scripting	Category
Show Value	Indicates whether the numeric value, capacity, and units are displayed.	boolean	.showValue	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Units	Units of measure for tank contents.	String	.units	Appearance
Value	Numeric value of tank's level.	double	.value	Data
Value Format	Format string used for the value.	String	.valueFormat	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Wave Height	The height of each wave.	int	.waveHeight	Appearance
Wave Length	The length of each wave.	int	.waveLength	Appearance

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

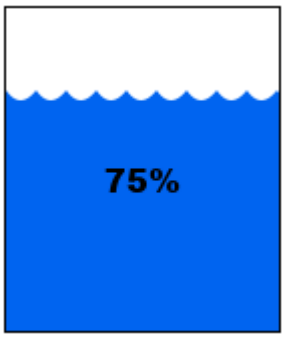
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the **Component Events** page

Customizers

- **Vision Component Customizers**
- **Style Customizer**

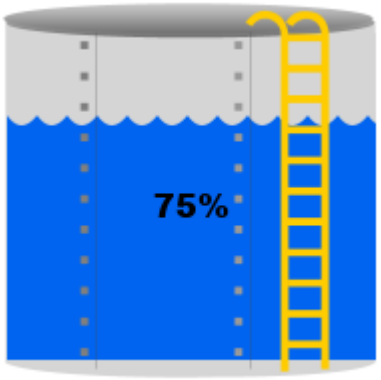
Examples

Level Indicator



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

Level Indicator



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

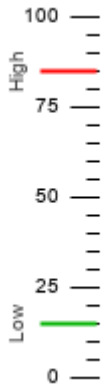
Property Name	Value
Border	Line Border
Value	75
Units	Gallons
Show Value	True
Gradient	False
Filled Color	0,100,240
Background Color	250,250,251
Font	Arial Black, Plain, 16
Wave Height	10
Wave Length	15

Setting value through Scripting

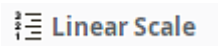
```
# You can set the component's value through scripting
event.source.parent.getComponent('Level Indicator').value = 5.4

# Alternatively, you can use the .setValue method to set the component's value
event.source.parent.getComponent('Level Indicator').setValue(5.4)
```

Vision - Linear Scale



Component Palette Icon:



The Linear Scale component displays a series of tick marks and labels representing a linear range, as well as indicators that represent a value or range of values positioned on the linear scale.

There is no tall/wide property for this component. This is based on the width/height of the component. A tall Linear Scale has tick marks on the left or right, and a wide component has tick marks on the top or bottom.

Properties

Property	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), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component. The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Fine Tick Color	The line color for fine ticks. Can be chosen from	Color	.fineTickColor	Appearance

Property	Description	Property Type	Scripting	Category
	color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector .			
Fine Tick Length	The line length for fine ticks, in pixels.	double	.fineTickLength	Appearance
Fine Tick Span	The span length for fine ticks. Should be a factor of the major and minor tick spans. Use zero to disable fine ticks.	double	.fineTickSpan	Data
Fine Tick Thickness	The line thickness for fine ticks, in pixels.	float	.fineTickStroke	Appearance
Indicators	This dataset stores the indicators (if any) for the scale.	Dataset	.indicators	Data
Label Angle	Changes the angle that the labels are drawn.	int	.labelAngle	Appearance

Property	Description	Property Type	Scripting	Category
Label Color	The color used for drawing tick labels. See Color Selector .	Color	.majorTickLabelColor	Appearance
Label Font	The font used for drawing tick labels. See **Color Selector**.	Font	.majorTickFont	Appearance
Label Format	The label format string. Examples: "%.1f" will render numbers like "15.0", "%.0f" will render numbers like "15". Using the empty string "" will disable the labels.	String	.majorTickLabelFormat	Appearance
Major Tick Color	The line color for major ticks. See **Color Selector**.	Color	.majorTickColor	Appearance
Major Tick Length	The line length for major ticks, in pixels.	double	.majorTickLength	Appearance
Major Tick Span	The span length for major ticks. Should be a multiple of the minor and fine tick spans.	double	.majorTickSpan	Data

Property	Description	Property Type	Scripting	Category
Major Tick Thickness	The line thickness for major ticks, in pixels.	float	.majorTickStroke	Appearance
Margin	The margin to leave blank as a percentage of the total height or width of the scale.	double	.margin	Appearance
Max Value	The upper bound of the scale.	double	.maxValue	Data
Min Value	The lower bound of the scale.	double	.minValue	Data
Minor Tick Color	The line color for minor ticks. See **Color Selector**.	Color	.minorTickColor	Appearance
Minor Tick Length	The line length for minor ticks, in pixels.	double	.minorTickLength	Appearance
Minor Tick Span	The span length for minor ticks. Should be a factor of the major tick span and a multiple of the fine tick spans. Use zero to disable minor ticks.	double	.minorTickSpan	Data

Property	Description	Property Type	Scripting	Category
Minor Tick Thickness	The line thickness for minor ticks, in pixels.	float	.minorTickStroke	Appearance
Mirror	Mirror the scale so it paints against the opposite edge.	boolean	.mirror	Appearance
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
Reverse Range	Reverse the scale so that values go from high to low instead of low to high.	boolean	.reverseRange	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

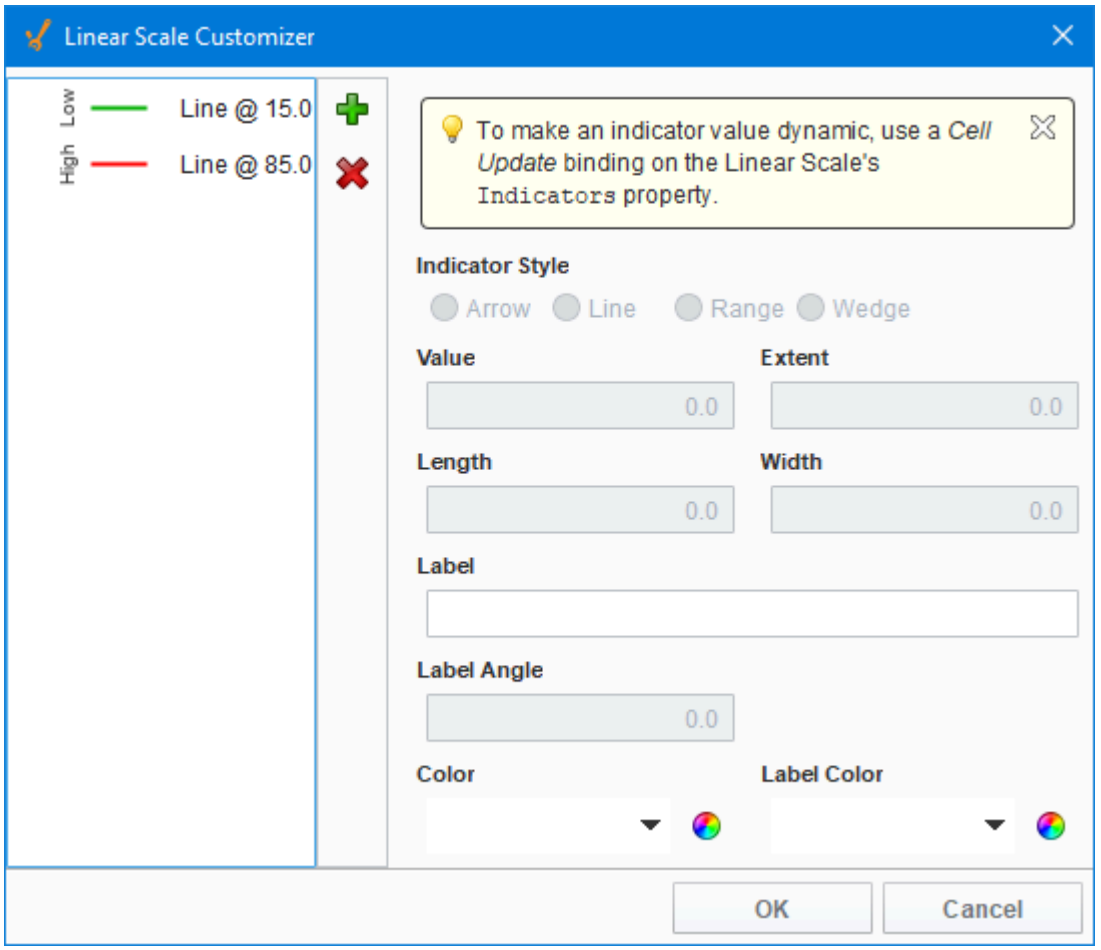
Event Handlers

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

Customizers

The Linear Scale Customizer allows you to configure the indicators that visually represent how your data is displayed on the scale. You can choose from several indicator styles: Arrow, Line, Range, and Wedge. Not all Linear Scale Customizer properties are available with all indicator styles. The property will be grayed out if it is not available for the selected indicator. Use the preview window to validate the style you want to use for your data.

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


Linear Scale Customizer
Low
High
Line @ 15.0
Line @ 85.0
To make an indicator value dynamic, use a Cell Update binding on the Linear Scale's Indicators property.
Indicator Style
Arrow
Line
Range
Wedge
Value
0.0
Extent
0.0
Length
0.0
Width
0.0
Label
Label Angle
0.0
Color
Label Color
OK
Cancel

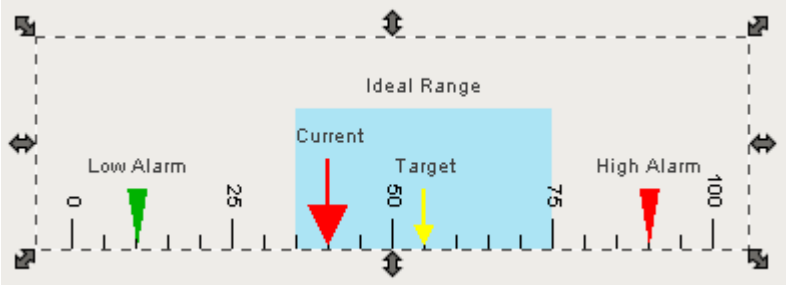
Linear Scale Customizer - Property Descriptions

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

- **Vision Component Customizers**

Examples

Horizontal Scale with Blue Indicators



Properties

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

Vision - Barcode



Component Palette Icon:



The barcode component displays some text as a barcode. The supported formats are:

- Code 128
- Code 39
- Extended Code 39
- Codabar
- Interleaved Code 25
- MSI
- EAN-13
- EAN-8
- Aztec
- Data Matrix
- PDF-417
- QR Code
- UPC-A

If you need to create a GSI-128 barcode, you can use the Code128 symbology. Use the following special characters to represent the four GSI function codes:

Function Code	Symbol
ESCAPE_FNC_1	ñ
ESCAPE_FNC_2	ò
ESCAPE_FNC_3	ó
ESCAPE_FNC_4	ô

Properties

Property	Description	Property Type	Scripting	Category
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.background	Appearance
Barcode Background	The background color of the actual barcode. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.barcodeBackground	Appearance
Barcode Format	The barcode format to display.	int	.barcodeType	Data
Barcode Height	The height of the barcode.	int	.barcodeHeight	Appearance
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel	Border	.border	Common

Property	Description	Property Type	Scripting	Category
	(Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.			
Check Digit	Include Check Digit?	boolean	.checkDigit	Data
Code	The code string that is converted into a barcode to display.	String	.code	Data
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.foreground	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common

Property	Description	Property Type	Scripting	Category
Narrowest Bar Width	The width (in pixels) of the narrowest bar.	int	.narrowestBarWidth	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Rotation	The angle of rotation in degrees.	int	.angleDegrees	Appearance
QRCode Error Correction Level	If you're creating a QR code, the QR code error correction level to use.	int	.qrEcLevel	Data
QRCode Version	If you're creating a QR code, the QR code version to use.	int	.qrCodeVersion	Data
Show Text?	If true, the code is displayed in human-readable text beneath the barcode.	boolean	.showText	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

Customizers

This component does not have any custom properties

Examples

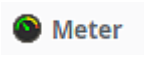


Property Name	Value
Code	123456789
Barcode Format	Extended Code 39 (narrow)
Show Text?	True

Vision - Meter



Component Palette Icon:



A meter display shows a value on a needle-gauge. The gauge's range can be broken up into five intervals. The intervals can have their own edge and background colors.

Properties

Property	Description	Property Type	Scripting	Category
Arc Width	The width of the colored interval arcs.	float	.arcWidth	Appearance
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field 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
Dial Background	The background color of the dial face. Can be chosen from color wheel, chosen from color palette,	Color	.dialBackground	Appearance

Property	Description	Property Type	Scripting	Category
	or entered as RGB or HSL value. See Color Selector .			
Dial Shape	The shape of the dial. This property determines how the dial face looks in the area not covered by the meter angle extent.	int	.dialType	Appearance
Interval 1 Background	The color to fill the wedge of this interval. See **Color Selector**.	Color	.interval1Background	Intervals
Interval 1 High	The upper bound of this interval.	double	.interval1High	Intervals
Interval 1 Low	The lower bound of this interval.	double	.interval1Low	Intervals
Interval 1 Outline	The color to paint the arc of this interval. See **Color Selector**.	Color	.interval1Outline	Intervals
Interval 2 Background	The color to fill the wedge of this interval. See **Color Selector**.	Color	.interval2Background	Intervals
Interval 2 High	The upper bound of this interval.	double	.interval2High	Intervals

Property	Description	Property Type	Scripting	Category
Interval 2 Low	The lower bound of this interval.	double	.interval2Low	Intervals
Interval 2 Outline	The color to paint the arc of this interval. See **Color Selector**.	Color	.interval2Outline	Intervals
Interval 3 Background	The color to fill the wedge of this interval. See **Color Selector**.	Color	.interval3Background	Intervals
Interval 3 High	The upper bound of this interval.	double	.interval3High	Intervals
Interval 3 Low	The lower bound of this interval.	double	.interval3Low	Intervals
Interval 3 Outline	The color to paint the arc of this interval. See **Color Selector**.	Color	.interval3Outline	Intervals
Interval 4 Background	The color to fill the wedge of this interval. See **Color Selector**.	Color	.interval4Background	Intervals
Interval 4 High	The upper bound of this interval.	double	.interval4High	Intervals
Interval 4 Low	The lower bound of this interval.	double	.interval4Low	Intervals
Interval 4 Outline	The color to paint the arc of this	Color	.interval4Outline	Intervals

Property	Description	Property Type	Scripting	Category
	interval. See **Color Selector**.			
Interval 5 Background	The color to fill the wedge of this interval. See **Color Selector**.	Color	.interval5Background	Intervals
Interval 5 High	The upper bound of this interval.	double	.interval5High	Intervals
Interval 5 Low	The lower bound of this interval.	double	.interval5Low	Intervals
Interval 5 Outline	The color to paint the arc of this interval. See **Color Selector**.	Color	.interval5Outline	Intervals
Meter Angle	The angle in degrees of the centerpoint of the meter (90 is straight up).	int	.meterAngle	Appearance
Meter Angle Extent	The extent, in degrees, of the entire meter.	int	.meterAngleExtent	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common

Property	Description	Property Type	Scripting	Category
Needle Color	The color of the meter's needle. See **Color Selector**.	Color	.needleColor	Appearance
Needle Size	The size of the base of the needle.	float	.needleSize	Appearance
Needle Stroke Color	The color of the needle's stroke. See **Color Selector**.	Color	.needleStrokeColor	Appearance
Needle Stroke Size	The size of the needle's stroke.	float	.needleStrokeSize	Appearance
Overall High Bound	The high bound for the whole meter.	double	.overallHigh	Data
Overall Low Bound	The lower bound for the whole meter.	double	.overallLow	Data
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Reverse Range?	If true, the meter will consider right to left needle movement as positive.	boolean	.reverseRange	Data
Show Tick Labels?	If true, value will be shown on	boolean	.ticks	Appearance

Property	Description	Property Type	Scripting	Category
	interval-boundary ticks.			
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Tick Color	The color of tick marks.	Color	.tickColor	Appearance
Tick Format	The number format to use for the tick labels.	String	.tickLabelFormat	Appearance
Tick Label Color	The color of the tick labels. See **Color Selector**.	Color	.tickLabelColor	Appearance
Tick Label Font	The font to use for the tick labels.	Font	.labelFont	Appearance
Tick Size	The distance between ticks.	double	.tickSize	Appearance
Units	A string to describe the units for the current value label.	String	.units	Appearance
Value	The value to display in this meter. The needle and current value label will change to reflect this.	double	.value	Data
Value Color	The color of the meter's current	Color	.valueColor	Appearance

Property	Description	Property Type	Scripting	Category
	value label. See **Color Selector**.			
Value Format	The number format to use for the value label.	String	.valueLabelFormat	Appearance
Value Label Font	The font to use for the current value label.	Font	.valueFont	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

See the **Vision - Meter Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

- Vision Component Customizers
- Style Customizer

Examples

Updated Fonts



Property Name	Value
Dial Background	0,0,128
Value	35
Unit	m/s
Value Label Font	Calibri, Italic, 16
Tick Label Font	Calibri, Italic, 12

Chord Meter with Modified Value Intervals



Property Name	Value
Value	35
Reverse Range?	True
Units	'None'
Arc Width	10
Meter Angle Extent	220
Meter Angle	0
Dial Shape	Chord
Interval 1 Low	40
Interval 2 High	60
Interval 2 Low	0
Interval 3 High	80
Interval 3 Low	60
Interval 4 High	100
Interval 3 Low	81

Vision - Meter Scripting Functions

This page details the various component and extension functions for **Vision's Meter component**.

Component Functions

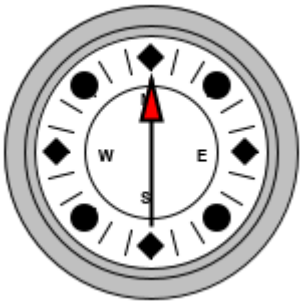
This component does not have component functions associated with it.

Extension Functions

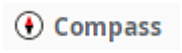
.configureChart(self, chart)

- Description
  - Provides an opportunity to perform further configuration via scripting.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - JFreeChart **chart** - A JFreeChart object. Refer to the **JFreeChart documentation** for API details.
- Return
  - None

Vision - Compass



Component Palette Icon:



The compass is a component that displays up to three needles at once on a cardinal direction compass. This can be useful for plotting anything that has a cardinal direction, such as the wind direction. Each needle can use one of nine different styles. Use the "Disabled" style to turn off any needle.

Properties

Property	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), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Center Color	The center color of the compass. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector .	Color	.centerColor	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Label Font	The font to use for the compass's	Font	.labelFont	Appearance

Property	Description	Property Type	Scripting	Category
	labels.			
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Rose Color	The background color of the rose. See Color Selector .	Color	.roseColor	Appearance
Rose Highlight	The highlight color of the rose. See **Color Selector**.	Color	.roseHighlightColor	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Value 1	Value 1 for the compass.	double	.value1	Data
Value 1 Color	The main color for Value 1's needle. See **Color Selector**.	Color	.value1Color	Appearance
Value 1 Needle	The needle type for this value.	int	.value1Needle	Data

Property	Description	Property Type	Scripting	Category
Value 1 Outline	The outline color for value 1's needle. See **Color Selector**.	Color	.value1OutlineColor	Appearance
Value 2	Value 2 for the compass.	double	.value2	Data
Value 2 Color	The main color for Value 2's needle. See **Color Selector**.	Color	.value2Color	Appearance
Value 2 Needle	The needle type for this value.	int	.value2Needle	Data
Value 2 Outline	The outline color for Value 2's needle. See **Color Selector**.	Color	.value2OutlineColor	Appearance
Value 3	Value 3 for the compass.	double	.value3	Data
Value 3 Color	The main color for Value 3's needle. See **Color Selector**.	Color	.value3Color	Appearance
Value 3 Needle	The needle type for this value.	int	.value3Needle	Data
Value 3 Outline	The outline color for Value 3's needle. See **Color Selector**.	Color	.value3OutlineColor	Appearance

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

Scripting

See the **Vision - Compass Scripting Functions** page for the full list of scripting functions available for this component.

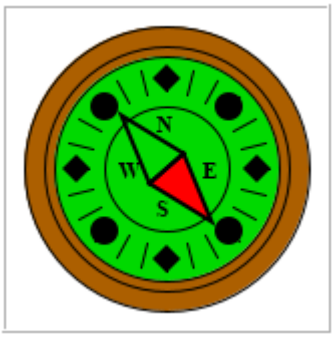
Event Handlers

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

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Examples



Property Name	Value
Center Color	0,217,0
Rose Color	172,95,0
Label Font	Times New Roman, Bold, 14
Value 1	140
Value 1 Color	255,0,0
Value 1 Needle	Pointer

Vision - Compass Scripting Functions

This page details the various component and extension functions for **Vision's Compass component**.

Component Functions

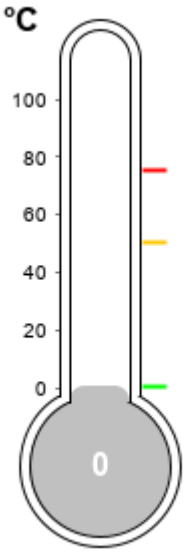
This component does not have component functions associated with it.

Extension Functions

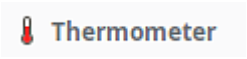
.configureChart(self, chart)

- Description
  - Provides an opportunity to perform further configuration via scripting.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - JFreeChart **chart** - A JFreeChart object. Refer to the **JFreeChart documentation** for API details.
- Return
  - None

Vision - Thermometer



Component Palette Icon:



This component displays a temperature value depicted as a level in a mercury thermometer. Three temperature intervals can optionally be defined with their own colors. The mercury will change color based on these intervals.

Properties

Property	Description	Property Type	Scripting	Category
Axis Label Color	The color of the meter's y-axis label. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.axisColor	Appearance
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Cursor	The mouse cursor to use when hovering over this component.	int	.cursorCode	Common

Property	Description	Property Type	Scripting	Category
	Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.			
Follow data in ranges	If true, the thermometer's Y axis will scale itself to zoom in on the current range.	boolean	.followDataInSubranges	Behavior
Interval 1 Color	The color of this interval. See **Color Selector**.	Color	.interval1Color	Intervals
Interval 1 High	The upper bound of this interval.	double	.interval1High	Intervals
Interval 1 Low	The lower bound of this interval.	double	.interval1Low	Intervals
Interval 2 Color	The color of this interval. See **Color Selector**.	Color	.interval2Color	Intervals
Interval 2 High	The upper bound of this interval.	double	.interval2High	Intervals

Property	Description	Property Type	Scripting	Category
Interval 2 Low	The lower bound of this interval.	double	.interval2Low	Intervals
Interval 3 Color	The color of this interval. See **Color Selector**.	Color	.interval3Color	Intervals
Interval 3 High	The upper bound of this interval.	double	.interval3High	Intervals
Interval 3 Low	The lower bound of this interval.	double	.interval3Low	Intervals
Mercury Color	The default color of the mercury. See **Color Selector**.	Color	.mercuryColor	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Overall High Bound	The high bound for the whole thermometer	double	.overallHigh	Data
Overall Low Bound	The lower bound for the	double	.overallLow	Data

Property	Description	Property Type	Scripting	Category
	whole thermometer			
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Styles	Contains the component's styles	Dataset	.styles	Appearance
Thermometer Color	The color of the outline of the thermometer. See **Color Selector**.	Color	.thermometerColor	Appearance
Thermometer Width	The width of the lines used to draw the thermometer.	int	.strokeWidth	Appearance
Units	A string to describe the units for the current value label.	int	.units	Appearance
Use Range Color	Controls whether or not the mercury color changes based on the range it is in.	boolean	.useSubrangePaint	Appearance
Value	The value to display in this	double	.value	Data

Property	Description	Property Type	Scripting	Category
	thermometer. The mercury level and value label will change to reflect this.			
Value Color	The color of the meter's current value label. See **Color Selector**.	Color	.valueColor	Appearance
Value Label Font	The font to use for the current value label.	Font	.valueFont	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

See the **Vision - Thermometer Scripting Functions** page for the full list of scripting functions available for this component.

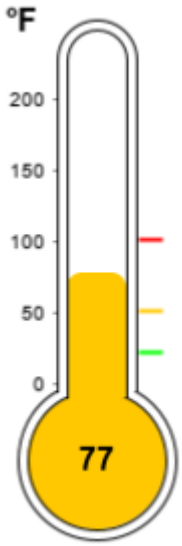
Event Handlers

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

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Examples



Property Name	Value
Units	Fahrenheit
Value	192
Interval 1 High	59
Interval 1 Low	20
Interval 2 High	100
Interval 2 Low	50
Interval 3 High	187
Interval 3 Low	100
Mercury Color	255, 200,0
Use Range Color	True

Vision - Thermometer Scripting Functions

This page details the various component and extension functions for **Vision's Thermometer component**.

Component Functions

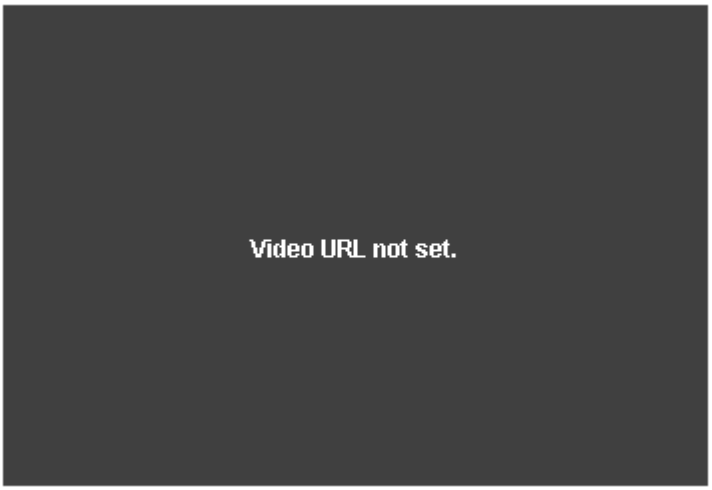
This component does not have component functions associated with it.

Extension Functions

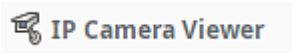
.configureChart(self, chart)

- Description
  - Provides an opportunity to perform further configuration via scripting.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - JFreeChart **chart** - A JFreeChart object. Refer to the **JFreeChart documentation** for API details.
- Return
  - None

Vision - IP Camera Viewer



Component Palette Icon:



The IP camera viewing component displays a video stream from a network camera directly in one of your windows. This can be a very powerful tool for allowing operators to view remote or inaccessible locations. Cameras can provide positive feedback about the state and position of machinery, weather, and other factors.

This component is capable of displaying two types of video:

- MJPEG (a.k.a. Motion JPEG) is a streaming video protocol that compresses video frames using standard JPEG compression. Compression rates are quite good, requiring low network bandwidth utilization. Framerates depend greatly on the dimensions of the video, but typically range from 1-20 frames per second.
- JPEG stills is not a true video protocol, but is rather the practice of continually refreshing an image that a camera is constantly overwriting. Its simplicity means that many cameras support it (usually along with another protocol). Frame rates are typically lower than MJPEG because a new connection must be opened for each frame.

Most network cameras on the market support one, if not both of these protocols. Even better, if you have an existing CCTV camera system, video server devices are available that CCTV camera inputs and provide MJPEG streams the network.

Finding the URL for your network camera's video stream is usually the only challenge in connecting this component. Most, if not all, network cameras have an internal web server, allowing viewers to use

web browsers to view their video stream. If you go to that webpage, and look at the HTML source of the page, you should be able to find the URL of the MJPEG or JPEG still stream.

> **HIGH RESOLUTION STREAMS**
>
> When viewing a feed from a High Resolution camera, the Camera Buffer Size property may need to be increased to contain all of the data from the stream.

Some examples:

Axis 2100 (MJPEG)

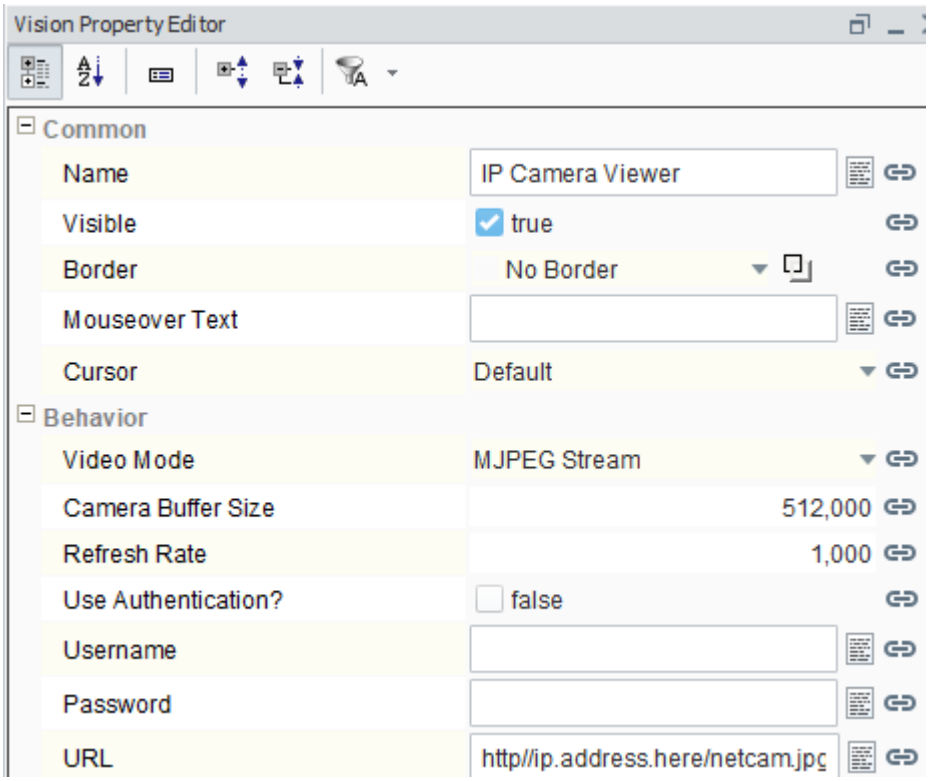
```
http://ip.address.here/axis-cgi/mjpg/video.cgi?resolution=640x480
```

Panasonic BL-C10A (MJPEG)

```
http://ip.address.here/nphMotionJpeg?Resolution=640x480&Quality=Standard
```

StarDot Netcam (JPEG stills)

```
http://ip.address.here/netcam.jpg
```



Properties

Property	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), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Camera Buffer Size	Set the size of the video buffer in bytes.	int	.cameraBufferSize	Behavior
Connection Retries	The number of times to attempt to connect to the stream.	int	.connectRetries	Behavior
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common

Property	Description	Property Type	Scripting	Category
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See **Color Selector**.	Color	.foreground	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Password	The password to authenticate with.	String	.password	Behavior
Refresh Rate	The rate (in ms) to poll the image if mode is 'JPEG Stills'.	int	.refreshRate	Behavior
Retry Delay	The delay (in ms) to wait between connection attempts.	int	.retryDelay	Behavior
Scale Mode	The scaling performance hint to use.	int	.scaleMode	Behavior
Scale Video	Scale the video to the size of the viewer component. Warning: CPU-intensive.	boolean	.scaleVideo	Behavior
Show Stats	If true, fps and Kbps statistical information	boolean	.showStats	Appearance

Property	Description	Property Type	Scripting	Category
	will be overlaid on the video.			
URL	The HTTP URL of the video stream to display.	String	.url	Behavior
Use Authentication?	If true, the URL connection will try to authenticate using the given username and password.	boolean	.useAuthentication	Behavior
User-Agent	If non-empty, the HTTP User-Agent to spoof.	String	.userAgent	Behavior
Username	The username to authenticate with.	String	.username	Behavior
Video Mode	Choose what type of video stream the URL points to.	int	.mode	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

Customizers

This component does not have any custom properties.

Examples



Property Name	Value
URL	http://trackfield.webcam.oregonstate.edu/mjpg/video.mjpg

Vision - Input Palette

The following components allow users to enter or select data.

Text Field

Numeric Text Field

Spinner

The spinner component represents a value that is part of a series of values, such as numbers and dates.

Formatted Text Field

This specialized text field is used for alphanumeric text input that must match some specific pattern or needs to b…

Password Field

A password field is like a text field that doesn't display the text that is being edited. You may alter the echo charac…

Text Area

Suitable for multi-line text display and editing. Will scroll vertically on demand. Will scroll horizontally if line wrap …

Dropdown List

The dropdown component is a great way to display a list of choices in a limited amount of space. The current sel…

Slider

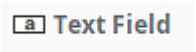
The slider component lets the user drag an indicator along a scale to choose a value in a range. The slider can be…

Language Selector

The Language Selector component gives an easy way to set the user's locale to control display of dates, times, n…

Vision - Text Field

Component Palette Icon:



Description

The Text Field component is used for input of any single-line text. This component will accept any alpha-numeric input. If you're looking for a numeric field, see the **Vision - Numeric Text Field**.

This field features a protected mode. When you enable the protectedMode property, the field is not editable even when it receives input focus. The user must double click on the field or press enter in order to edit the field. When they are done (press enter again or leave the field), the field becomes non-editable again.

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

Properties

Name	Description	Property Type	Scripting	Category
Background	The background color of the text box (when editable).	Color	.editableBackground	Appearance
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Field Border, and Line Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Commit On Focus Loss	If true, any pending edit will take effect when focus is lost. If false, the user must	boolean	.commitOnFocusLost	Behavior

Name	Description	Property Type	Scripting	Category
	press ENTER for an edit to take effect.			
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Defer Updates	When true, the 'text' property will not fire updates while typing, it will wait for Enter to be pressed.	boolean	.deferUpdates	Behavior
Editable?	If true, this is an input box, if false, this is display-only.	boolean	.editable	Behavior
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common

Name	Description	Property Type	Scripting	Category
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance
Horizontal Alignment	Determines the alignment of the label's contents along the X axis.	int	.horizontalAlignment	Layout
Maximum Characters	The text box will be limited to this number of characters. Use -1 for unlimited.	int	.maxChars	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up when the user mouses over of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Non-Editable Background	The background	Color	.nonEditableBackground	Appearance

Name	Description	Property Type	Scripting	Category
	color to use when this text box is non-editable.			
Protected Mode?	If true, users will need to double-click in the field in order to edit the text.	boolean	.protectedMode	Behavior
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Reject Updates During Edit	If true, this field will not accept updates from external sources (like DB bindings) while the user is editing the field.	boolean	.rejectUpdatesDuringEdit	Behavior
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Text	Text of this component.	String	.text	Data

Name	Description	Property Type	Scripting	Category
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
Touchscreen Keyboard Layout	Sets the touchscreen keyboard layout to use for this component.	String	.keyboardName	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

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

Scripting

Scripting Functions

See the **Vision - Text Field Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Examples

Code Snippet

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

oldValue = event.source.oldValue
```

Titled Panel



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

Vision - Text Field Scripting Functions

This page details the various component and extension functions available for **Vision's Text Field component**.

Component Functions

getSelectedText()

- Description
  - Returns the currently selected or highlighted text in the text field.
- Parameters
  - Nothing
- Return
  - String - Returns the currently selected or highlighted text in the text field.
- Scope
  - Client

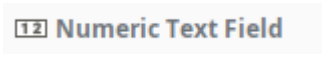
Extension Functions

This component does not have extension functions associated with it.

Vision - Numeric Text Field



Component Palette Icon



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

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

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

Properties

Property	Description	Property Type	Scripting	Category
Background	The background color of the text box (when editable).	Color	.editableBackground	Appearance
Border	The border surrounding this component. No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Field Border, and Line Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Commit On Focus Loss	If true, any pending edit will take effect when focus is lost. If false, the user must press Enter for	boolean	.commitOnFocusLost	Behavior

Property	Description	Property Type	Scripting	Category
	an edit to take effect.			
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Decimal Format	The formatting string used for displaying numbers.	String	.decimalFormat	Appearance
Defer Updates	When true, the value properties will not fire updates while typing, it will wait for Enter to be pressed.	boolean	.deferUpdates	Behavior
Editable?	If true, this is an input box, if false, this is display-only.	boolean	.editable	Behavior
Enabled	If disabled, a component	boolean	.componentEnabled	Common

Property	Description	Property Type	Scripting	Category
	cannot be used.			
Error on Out-of-Bounds	Show an error message if the user input is out-of-bounds?	boolean	.errorOnOutOfBounds	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance
Horizontal Alignment	Determines the alignment of the label's contents along the X axis.	int	.horizontalAlignment	Layout
Maximum	The maximum value (inclusive), if useBounds is true.	double	.maximum	Data
Minimum	The minimum value (inclusive), if useBounds is true.	double	.minimum	Data
Mouseover Text	The text that is displayed in the tooltip	String	.toolTipText	Common

Property	Description	Property Type	Scripting	Category
	which pops up on mouseover of this component.			
Name	The name of this component.	String	.name	Common
Non-Editable Background	The background color to use when this text box is non-editable	Color	.nonEditableBackground	Appearance
Number Type	What type of numbers should this field accept?	int	.mode	Data
Out Of Bounds Message	The error message to display if input is out-of-bounds.	String	.outOfBoundsMessage	Behavior
Protected Mode?	If true, users will need to double-click in the field in order to edit the value.	boolean	.protectedMode	Behavior
Quality	The data quality code for any Tag bindings on	QualityCode	.quality	Data

Property	Description	Property Type	Scripting	Category
	this component.			
Reject Updates During Edit	If true, this field will not accept updates from external sources (like DB bindings) while the user is editing the field.	boolean	.rejectUpdatesDuringEdit	Behavior
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Suffix	A string to display after the value.	String	.suffix	Appearance
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
Use Bounds?	Only allows user-entered values between a minimum and maximum. Unless you turn on "Error on out-of-	boolean	.useBounds	Behavior

Property	Description	Property Type	Scripting	Category
	bounds", user-entered values will be silently modified to be in-bounds.			
Value (Double)	The value as a double. Make sure you use the value property that corresponds to your Number Type setting.	double	.doubleValue	Data
Value (Float)	The value as a float. Make sure you use the value property that corresponds to your Number Type setting.	float	.floatValue	Data
Value (Integer)	The value as an integer. Make sure you use the value property that corresponds to your Number Type setting.	int	.intValue	Data
Value (Long)	The value as a long. Make sure you use the value property that	long	.longValue	Data

Property	Description	Property Type	Scripting	Category
	corresponds to your Number Type setting.			
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

See the **Vision - Numeric Text Field Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Examples

Code Snippet

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

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

2-digit Numeric Format



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

Vision - Numeric Text Field Scripting Functions

This page details the various component and extension functions available for **Vision's Numeric Text Field component**.

Component Functions

getSelectedText()

- Description

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

- Parameters

  Nothing

- Return

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

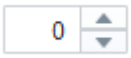
- Scope

  Client

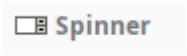
Extension Functions

This component does not have extension functions associated with it.

Vision - Spinner



Component Palette Icon:



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

Properties

Property	Description	Property Type	Scripting	Category
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Field Border, and Line Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Date Format	A date format pattern to use when the spinner is in date mode.	String	.dateFormat	Appearance
Date in Milliseconds	The date in milliseconds from epoch time. (Read only. Usable in bindings and scripting.)	long	.dateInMillis	Uncategorized

Property	Description	Property Type	Scripting	Category
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.foreground	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Number Format	A number format pattern to use when the spinner is in numeric mode.	String	.numberFormat	Appearance
Numeric Maximum	The maximum value this spinner	double	.maxValue	Data

Property	Description	Property Type	Scripting	Category
	will accept when in 'Integer' or 'Double' mode.			
Numeric Minimum	The minimum value this spinner will accept when in 'Integer' or 'Double' mode.	double	.minValue	Data
Numeric Step Size	The size to step up or down when in 'Integer' or 'Double' mode.	double	.stepSize	Behavior
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Spinner Mode	The mode controls which data type this spinner accepts.	int	.spinnerMode	Behavior
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior

Property	Description	Property Type	Scripting	Category
Value (Date)	The current value if mode is 'Date'.	Date	.dateValue	Data
Value (Double)	The current value if mode is 'Double'.	double	.doubleValue	Data
Value (Integer)	The current value if mode is 'Integer'.	int	.intValue	Data
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

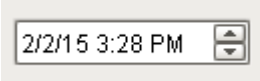
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the **Component Events** page

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Examples

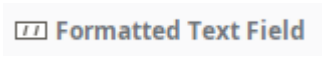
Date Spinner



Property Name	Value
Spinner Mode	Date

Vision - Formatted Text Field

Component Palette Icon:



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

Formatted Mask

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

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

Regular Expression

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

Properties

Property	Description	Property Type	Scripting	Category
Allows Invalid Text	Allows Invalid text to Commit.	boolean	.allowsInvalid	Behavior
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.background	Appearance
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Commit While Typing	Commits valid text while user is typing.	boolean	.commitsOnValidEdit	Behavior

Property	Description	Property Type	Scripting	Category
Committed Value	Committed text value.	String	.committedValue	Data
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Focus Lost Behavior	Controls how a transaction can be committed.	int	.focusLostBehavior	Behavior
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value.	Color	.foreground	Appearance

Property	Description	Property Type	Scripting	Category
	See **Color Selector**.			
Formatted Mask Pattern	Formatted Mask Validation Pattern.	String	.formattedMaskPattern	Behavior
Horizontal Alignment	Determines the alignment of the label's contents along the X axis.	int	.horizontalAlignment	Layout
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Overwrites Text	Overwrites text while typing.	boolean	.overwriteMode	Behavior
Reg Ex Pattern	Regular Expression Validation Pattern.	String	.validationPattern	Behavior
Styles	Contains the component's styles.	Dataset	.styles	Appearance

Property	Description	Property Type	Scripting	Category
Text	Contents of this Text Field.	String	.text	Data
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
Touchscreen Keyboard Layout	Sets the touchscreen keyboard layout to use for this component.	String	.keyboardName	Behavior
Validation Mode	Select regular expression or mask-driven field validation.	int	.validationMode	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Examples

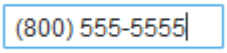
Formatted Mask

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

Regular Expression

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

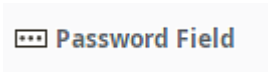
Phone Number Format



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

Vision - Password Field

Component Palette Icon:



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

Properties

Property	Description	Property Type	Scripting	Category
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Field Border, and Line 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,	int	.cursorCode	Common

Property	Description	Property Type	Scripting	Category
	Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.			
Echo Character	The character that is displayed instead of the real ones.	String	.echoCharacter	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common

Property	Description	Property Type	Scripting	Category
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Reject Updates During Edit	If true, this field will not accept updates from external sources (like DB bindings) while the user is editing the field. Default value is true.	boolean	.rejectUpdatesDuringEdit	Behavior
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Text	Text of this component	String	.text	Data
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
Touchscreen Keyboard Layout	Sets the touchscreen keyboard	String	.keyboardName	Behavior

Property	Description	Property Type	Scripting	Category
	layout to use for this component.			
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

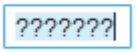
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the **Component Events** page

Customizers

- Vision Component Customizers
- Style Customizer

Example

Password Field with Question Marks as the Echo Character

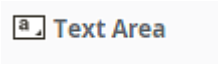


Property Name	Value
Echo Character	?

Vision - Text Area



Component Palette Icon:



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

Properties

Property	Description	Property Type	Scripting	Category
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Field Border, and Line Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Columns	The number of columns you expect to display (used as a hint for scrollbars).	int	.columns	Appearance

Property	Description	Property Type	Scripting	Category
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Defer Updates	When true, the 'text' property will not fire updates while typing. It will wait for the component to lose focus.	boolean	.deferUpdates	Behavior
Editable	Controls whether or not the user can edit the text within this text area. When the option is not selected, the text is not editable in the client and the background of the component will be grey.	boolean	.editable	Behavior

Property	Description	Property Type	Scripting	Category
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance
Line Wrap	Should this area wrap lines?	boolean	.lineWrap	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data

Property	Description	Property Type	Scripting	Category
Reject Updates During Edit	If true, this field will not accept updates from external sources (like DB bindings) while the user is editing the field.	boolean	.rejectUpdatesDuringEdit	Behavior
Rows	The number of rows you expect to display (used as a hint for scrollbars).	int	.rows	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Tab Size	This adjusts the default size of tab characters.	int	.tabSize	Appearance
Text	Text of this component.	String	.text	Data
Touchscreen Mode	Controls when this input component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior

Property	Description	Property Type	Scripting	Category
Touchscreen Keyboard Layout	Sets the touchscreen keyboard layout to use for this component.	String	.keyboardName	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

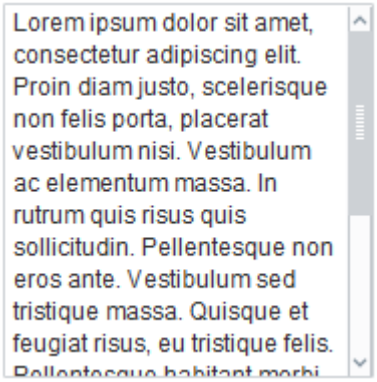
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the **Component Events** page

Customizers

- Vision Component Customizers
- Style Customizer

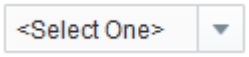
Examples

Word Wrap and No Scroll Bars until Needed

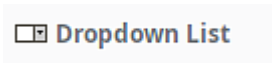


Property Name	Value
Line Wrap	True
Text	468 Characters
Rows	0
Columns	0

Vision - Dropdown List



Component Palette Icon:



The Dropdown component displays a list of choices in a limited amount of space. The list of choices is stored in a dataset, which can be manually typed in the Designer or populated dynamically from a property binding, often a SQL Query binding.

You may want to display choices to the user that are mapped to a different value internally, such as a numeric code. The columns in your dataset will determine which strings are visible to the end user (**Selected Label**) and which integers or strings are stored as the component's **Selected Value** or **Selected String Value**. There are three ways to configure a dropdown dataset:

Scenario 1: One column dataset with a set of string values

Column1
Apples
Oranges
Bananas

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

Scenario 2: Two column dataset with an integer and a string column

Column1	Column2
201	Apples
202	Oranges
203	Bananas

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

Scenario 3: Two column dataset with two string columns

Column1	Column2
APL	Apples
ORN	Oranges
BAN	Bananas

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

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

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

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

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

Properties

Property	Description	Property Type	Scripting	Category
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Field Border, and Line Border. **Note**: The border is unaffected by rotation.	Border	.border	Common

Property	Description	Property Type	Scripting	Category
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
Data	The dataset containing the list of choices in the dropdown. Either a one-column or two-column dataset. The first column is always the stored value, and the second column (if present) will be what is displayed to the user.	Dataset	.data	Data
Dropdown Display Mode	Changes the dropdown's display.	int	.mode	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common

Property	Description	Property Type	Scripting	Category
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance
Hide Table Columns?	A comma separated list of columns to hide from the dropdown table, for example, "0,2" (only used in table mode).	String	.hideTableColumns	Appearance
Horizontal Alignment	Determines the alignment of the contents along the X axis.	int	.horizontalAlignment	Layout
Max Row Count	The number of rows to display in the dropdown list before displaying a scrollbar.	int	.maximumRowCount	Appearance
Max Table Height	The maximum height allowed for the dropdown table (only used in table mode). If Max Table Height is negative, the	int	.maxTableHeight	Appearance

Property	Description	Property Type	Scripting	Category
	table popup will auto size to fill contents.			
Max Table Width	The maximum width allowed for the dropdown table (only used in table mode). If Max Table Width is negative, the table popup will auto size to fill contents.	int	.maxTableWidth	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
No Selection Label	The label to display when nothing is selected.	String	.noSelectionLabel	Behavior
No Selection String	The string value when nothing is selected.	String	.noSelectionString	Behavior
No Selection Value	The value when nothing is	int	.noSelectionValue	Behavior

Property	Description	Property Type	Scripting	Category
	selected.			
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Row Height	Determines the height of each item in the dropdown list. The default is -1 pixels, which causes the row height to be determined automatically by the current font.	int	.rowHeight	Appearance
Selected Index	The index of the selected item. (Read only. Usable in bindings and scripting.)	int	.selectedIndex	Uncategorized
Selected Label	The currently selected label.	String	.selectedLabel	Data
Selected String Value	The currently selected value, if the value column is a string.	String	.selectedStringValue	Data
Selected Value	The currently selected value.	Integer	.selectedValue	Data

Property	Description	Property Type	Scripting	Category
Selection Background	The background color of a selected cell in the dropdown list.	Color	.selectionBackground	Appearance
Selection Mode	The selection mode determines the behavior of the dropdown: whether its selected value must strictly be in the underlying set of choices, whether it is flexible, or if users can type into the component.	int	.selectionMode	Behavior
Show Table Header?	Selects whether or not the dropdown table header is displayed (only used in table mode).	boolean	.showTableHeader	Appearance
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Vertical Alignment	Determines the alignment of the	int	.verticalAlignment	Layout

Property	Description	Property Type	Scripting	Category
	contents along the Y axis.			
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

Customizers

- **Vision Component Customizers**

- Style Customizer

Examples

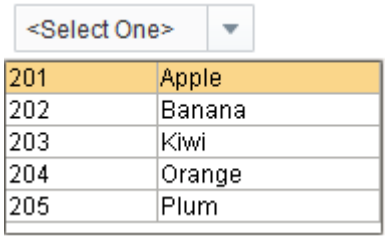
Display Multiple Columns in Dropdown

Code Snippet

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

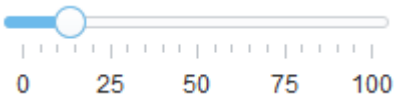
selRow = event.source.parent.getComponent('Dropdown').selectedIndex
data = event.source.parent.getComponent('Dropdown').data

code = data[selRow][0]
print code
```

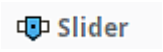


Property Name	Value
Dropdown Display Mode	Table
Show Table Header	False

Vision - Slider



Component Palette Icon:



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

Properties

Property	Description	Property Type	Scripting	Category
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Field Border, and Line 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
Defer Updates	Only publish updates to value when not actively being changed.	boolean	.deferred	Behavior

Property	Description	Property Type	Scripting	Category
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector .	Color	.foreground	Appearance
Horizontal Slider	If true, slider is horizontal, otherwise, it's vertical.	boolean	.horizontal	Appearance
Inverted?	Specify true to reverse the value range shown for the slider and false to put the value range in the normal order.	boolean	.inverted	Behavior
Major Tick Spacing	The distance, measured in values, between each major tick mark.	int	.majorTickSpacing	Appearance
Maximum Value	The value when the slider is all the way right or up.	int	.maximum	Data

Property	Description	Property Type	Scripting	Category
Minimum Value	The value when the slider is all the way left or down.	int	.minimum	Data
Minor Tick Spacing	The distance, measured in values, between each minor tick mark.	int	.minorTickSpacing	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Paint Labels?	If true, value labels will be shown.	boolean	.paintLabels	Appearance
Paint Ticks?	If true, value tick marks will be shown.	boolean	.paintTicks	Appearance
Paint Track?	If true, the track of the slider will be shown.	boolean	.paintTrack	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Snap To Ticks?	Only allows selection of values at the tick marks.	boolean	.snapToTicks	Behavior

Property	Description	Property Type	Scripting	Category
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Value	The current value of the slider.	int	.value	Data
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

Customizers

- Vision Component Customizers
- Style Customizer

Examples

Code Snippet

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

oldValue = event.source.oldValue
```

Vertical Slider with Border and Blue Text



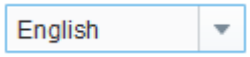
Property Type	Value
Maximum Value	250
Minor Tick Spacing	25
Foreground Color	0,0,255
Major Tick Spacing	50

Horizontal Slider without Tickmarks



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

Vision - Language Selector



Component Palette Icon:



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

Properties

Property	Description	Property Type	Scripting	Category
Background Color	The background color of the component.	Color	.background	Appearance
Border	The border surrounding this component. No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Field Border, and Line Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component.	Color	.foreground	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common

Property	Description	Property Type	Scripting	Category
Selected Locale	The display name of the currently selected locale. (Read only. Usable in bindings and scripting.)	String	.selectedLocale	Uncategorized
Selection Background	The background color of a selected cell in the dropdown list.	Color	.selectionBackground	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

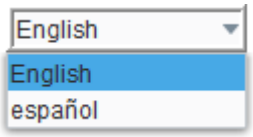
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the **Component Events** page

Customizers

This component does not have any customizers.

Example

Select Between Languages



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

Vision - Misc Palette

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

Paintable Canvas

The Paintable Canvas component can be custom painted using Jython scripting.

Line

The Line component displays a straight line. It can run north-south, east-west, or diagonally.

Pipe Segment

The Pipe Segment component displays a quasi-3D pipe.

Pipe Joint

The Pipe Joint displays a fancy joint component two join two pipe segments together.

Sound Player

The Sound Player component is an invisible component that facilitates audio playback in the client.

Timer

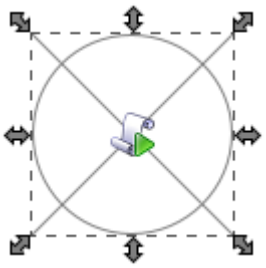
The Timer component is an invisible button that can be used to create repeated events in a window.

Signal Generator

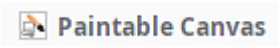
The Signal Generator is similar to the Timer component, but its value isn't simply a counter.

Web Browser

Vision - Paintable Canvas



Component Palette Icon:

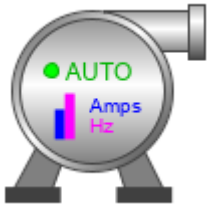


Description

The Paintable Canvas component can be custom "painted" using Jython scripting. By responding to the component's **repaint** Event Handler, 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.

> **NOTE**
>
> This component is an advanced component for those who are comfortable using scripting. It is not user-friendly, but 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 displayed because of the pre-loaded code in the repaint Script Editor. By editing the script, 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 accordingly in preview mode. Java2D is a vector drawing library, enabling you to create components that scale gracefully.



TIPS

- 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), and Field 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 Color	The foreground color of the component. See **Color**	Color	.foreground	Appearance

Name	Description	Property Type	Scripting	Category
	Selector.			
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

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

Customizers

- Vision Component Customizers
- Style Customizer

Examples

Drawing Custom Images

It is possible to draw geometric primitives and arbitrary shapes on the Paintable Canvas. This includes points, lines, curves, ellipses, arcs, rectangles, paths, and any combinations of the above to create a completely custom image. Below are script examples illustrating this functionality.

Script 1: Draw Geometric Primitives - Rectangle

```
from java.awt import Color
from java.awt.geom import Rectangle2D

# Specify the rectangle's size in pixels
width  = 100
height = 50

# Specify its position, in pixels, relative to the top left edge of the component
x = 20
y = 30

# Create the rectangle using the specs from steps 1 and 2.
rectangle = Rectangle2D.Float( x, y, width, height)

# Set the color and draw the rectangle
g = event.graphics
g.setColor(Color.GREEN)
g.draw(rectangle)
```

Script 2: Draw an Arbitrary Shape Based on the Coordinates Stored in a Dataset

```
# This example implies a dataset with each row storing the X and Y coordinates of
a single point.
# The columns in this case are called "xCoord" and "xCoord", respectively.

from java.awt import Color
from java.awt.geom import GeneralPath

# Get the shape coordinates from a dataset
coordinates = event.source.points

# Create an arbitrary shape
shape = GeneralPath()

# Create a starting point from the 0-th row of the dataset
shape.moveTo(coordinates.getValueAt(0, "xCoord"), coordinates.getValueAt(0,
"yCoord"))

# Iterate through the rows of coordinates and draw the lines
for rowNdx in range(1, coordinates.rowCount):
    shape.lineTo(coordinates.getValueAt(rowNdx, "xCoord"),
coordinates.getValueAt(rowNdx, "yCoord"))

# Close up the shape
shape.closePath()

# Set the color and draw the shape.
g = event.graphics
g.setColor(Color.GREEN)
g.draw(shape)
```

Additionally, you can specify the line color, stroke style, fill color and gradient for the shape. Refer to the following Oracle tutorials for more infomation:

- **Geometric primitives**
- **Arbitrary shapes**
- **Stroking and Filling Graphics**

Working with Existing Images

One of the most common use cases for the Paintable Canvas component is displaying images retrieved either from a database table using a SQL query or from a file server using the image URL.

Displaying an Image Stored in a Database Table

The following script shows how to retrieve an image stored as a BLOB in a SQL database and display it using the Paintable Canvas component. The query in this example relies on a specific SQL table structure. Make sure to adjust the query, including table and column names, as well as any filtering, to match your table structure and needs.

Displaying an Image Stored in a Database Table Script

```
from javax.imageio import ImageIO
from java.io import ByteArrayInputStream

# See if the image is cached; if not, load it from the database
bytes = event.source.getClientProperty("img-bytes-cached")

if not bytes:
    query = "SELECT image_blob FROM imagestore LIMIT 1"
    bytes = system.db.runScalarPrepQuery(query, [])
    event.source.putClientProperty("img-bytes-cached", bytes)

# Read the image from the bytes
image = ImageIO.read(ByteArrayInputStream(bytes))

# Draw the image
event.graphics.drawImage(image, 0, 0, event.source)
```

Displaying an Image from a URL

The script below shows how to retrieve and display an image from a URL. The URL can be hard-coded, or read from a custom property.

Displaying an Image from a URL

```
from java.net import URL
from javax.imageio import ImageIO

# Obtain the URL string.
imageURL = "your/URL/here"

# Create a Java URL object from the URL string
url = URL(imageURL)

# Read the image at the URL
image = ImageIO.read(url)

# Display the image, aligned to the top left corner
event.graphics.drawImage(image, 0, 0, event.source)
```

ADJUSTING IMAGE SIZE

When displaying existing images, make sure to consider their original size. If the image does not have the same dimensions as the size of the Paintable Canvas component, it will either be cropped or only partially fill the Paintable Canvas component depending on if it's larger or smaller than the Paintable Canvas.

However, you can scale the image using the `scale` method of the Graphics2D object. Refer to **Scaling, Zooming, and Rotating** for examples.

Drawing Text

If you want to display text on the Paintable canvas by itself or to label images, Java Graphics2D library provides the `drawString` method. The string given to the drawString method can be specified as a literal, obtained from a custom property, or put together programmatically.

```
from java.awt import Color
from java.awt.geom import GeneralPath

# Specify the string to display
string = "Some string to display."

# Specify X and Y coordinates of the lower left point for the displayed text
startingX = 50
startingY = 10

# Set the color and draw the string
g = event.graphics
g.setColor(Color.BLUE)
g.drawString(string,  startingX, startingY)
```

Scaling, Zooming, and Rotating

You can scale an image by simply defining the scale factor or calculating the width and height values to dynamically match the component size. Dynamically scaling an image is most helpful when the ratio of the image size to the component size is not known in advance.

Defining the Scale Factor

```
# Get a reference to the  java.awt.Graphics2D object
g = event.graphics

# Define the scale factor, and scale the image horizontally and vertically
scaleFactor = 2.0
g.scale(scaleFactor, scaleFactor)

# Draw the image, aligned ot the top left of the component
g.drawImage(image,0,0,event.source)
```

Dynamically Scale an Image

```
# Get the width and height of the BufferedImage object returned from ImageIO.read
call:
imageW = image.getWidth()
imageH =  image.getHeight()

# Get the width and height of the Paintable Canvas component
canvasW = event.width
canvasH = event.height

# Calculate the scale factors;
# Account for 1 pixel of the component's border dimensions
scaleX = (canvasW-1.0) / imageW
scaleY = (canvasH-1.0) / imageH

# Apply scaling and draw the image
g = event.graphics
g.scale(scaleX, scaleY)
g.drawImage(image,0,0,event.source)
```

The same dynamic technique as the scaling option above can be used to dynamically control the zoom factor via a custom property. The following example implies that the Paintable Canvas component has an Integer type custom property called `zoom` containing the zoom percentage (200 for zooming by 200%).

Controlling the Zoom Factor

```
from javax.imageio import ImageIO
from java.io import ByteArrayInputStream

# Get the image bytes from the database or cache
bytes = event.source.getClientProperty("img-bytes-cached")
if not bytes:
    bytes = system.db.runScalarPrepQuery(
        "SELECT image_blob FROM imagestore LIMIT 1",
        []
    )
    event.source.putClientProperty("img-bytes-cached", bytes)

# Read the image from the bytes
image = ImageIO.read(ByteArrayInputStream(bytes))

# Get the scale factor and convert the Integer percent value to a Float scale
factor
scaleFactor = event.source.zoom * 0.01

# Scale and draw the image
g = event.graphics
g.scale(scaleFactor, scaleFactor)
g.drawImage(image, 0, 0, event.source)
```

This last example shows how to rotate the image based on rotation center and angle values. The rotation center coordinates and the rotation angle value are obtained from the component's custom properties. With this, user controls like sliders, can be provided to dynamically control rotation.

Rotate an Image

```
from javax.imageio import ImageIO
from java.io import ByteArrayInputStream

# Get the image bytes from the database or cache
bytes = event.source.getClientProperty("img-bytes-cached")
if not bytes:
    bytes = system.db.runScalarPrepQuery(
        "SELECT image_blob FROM imagestore LIMIT 1",
        []
    )
    event.source.putClientProperty("img-bytes-cached", bytes)

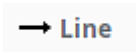
# Read the image from the bytes
image = ImageIO.read(ByteArrayInputStream(bytes))

# Get the center and angle of rotation from custom properties
rotationCenterX = event.source.rotationCenterX
rotationCenterY = event.source.rotationCenterY
rotationAngle   = event.source.rotationAngle

# Draw the image
g = event.graphics
g.rotate(rotationAngle, rotationCenterX, rotationCenterY)
g.drawImage(image, 0, 0, event.source)
```

Vision - Line

Component Palette Icon:



Description

The line component displays a straight line. It can run north-south, east-west, or diagonally. You can add arrows to either side. The line can be dashed using any pattern you want. You can even draw the line like a sinusoidal wave.

> **REPORTING LINE COMPONENT**
>
> If you are looking for the Line component used in Reporting, refer to 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

Name	Description	Property Type	Scripting	Category
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

Name	Description	Property Type	Scripting	Category
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

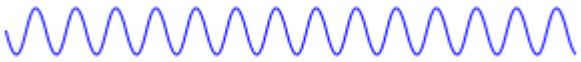
Event handlers allow you to run a script based off specific triggers. See the full list of available event handlers on the **Component Events** page

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Examples

Line with Sinusoidal Pattern



Property	Setting
Color	0,0,255
Line Style	Sinusoidal

Line with Arrow Endpoints

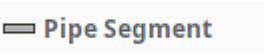


Property	Setting
Color	217,0,0
Line Style	Plain
Left Arrow	True
Left Arrow Size	25
Line Width	4
Right Arrow	True
Right Arrow Size	25

Vision - Pipe Segment



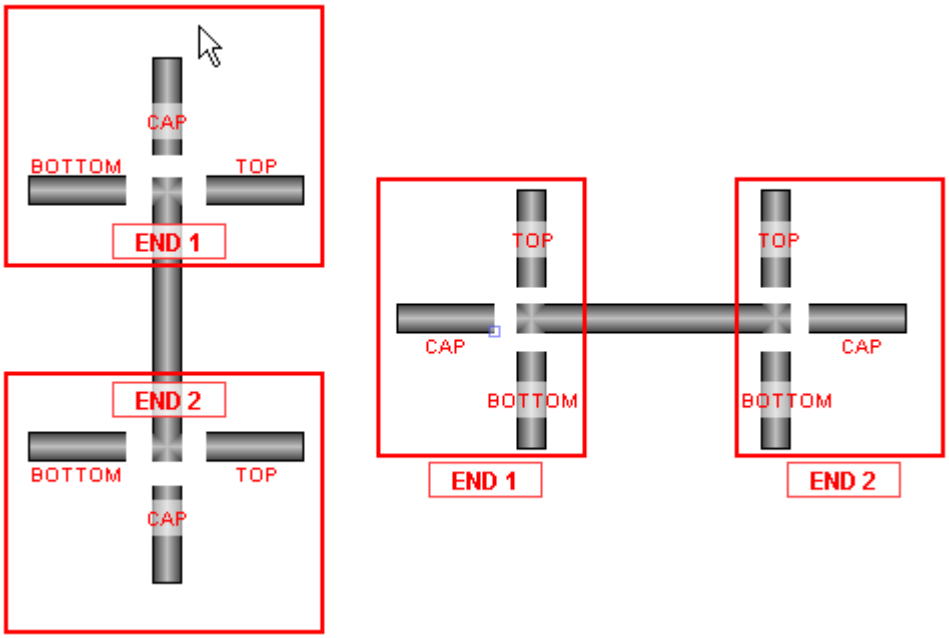
Component Palette Icon:



Description

The pipe segment component displays a quasi-3D pipe. In its basic form it looks very much like a rectangle with a round gradient. The difference comes in its advanced rendering of its edges and endcaps. You can configure each pipe segment's end to mate perfectly with another pipe segment butted up against it perpendicularly. The result looks like a pipe welded together in a 90° corner.

The control of the pipe's ends are done using 6 booleans - three per 'end'. End 1 is the top/left end, and End 2 is the bottom/right end. You turn off each boolean if there will be another pipe butted up against that side. The following diagram illustrates the naming conventions:



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), and Field 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	.secondaryColor	Appearance
End 1 Bottom?	Draw the border at end #1's bottom?	boolean	.end1Bottom	Appearance

Name	Description	Property Type	Scripting	Category
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

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

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Vision - Pipe Joint



Component Palette Icon:



Description

The pipe joint displays a joint component to join two pipe segments together. By turning off the cardinal directions, this will display a two-, three-, or four-pipe union.

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), and Field 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

Name	Description	Property Type	Scripting	Category
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

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

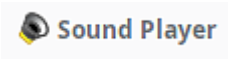
Customizers

- **Vision Component Customizers**
- **Style Customizer**

Vision - Sound Player



Component Palette Icon:



Description

The Sound Player component is an invisible component that facilitates audio playback in the client. Each Sound Player component has one sound clip associated with it, and will play that clip on demand. There is a built in triggering system, as well as facilities to loop the sound while the trigger is set. 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

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

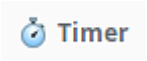
Customizers

- **Vision Component Customizers**

Vision - Timer



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 tis the Bound, at which point it repeats. It is often useful to bind other values to a timer's Value property.

For instance, if you set the timer's Bound property to 360, and bind an object's rotation to the Value property, the object will spin in a circle when the timer is running. How fast the timer counts is up to the Delay property, which is the time between counts in milliseconds.

Want to run a script every time the timer counts? First, make sure you don't actually want to write a project **Timer Script**, which will run on some interval whenever the application is running. In contrast, a script that works via a Timer component will only run while the window that contains the Timer is open, and the Timer is running. The way to do this is to attach an event script to the **actionPerformed** event.

Properties

Name	Description	Property Type	Scripting	Category
Bound	The value is always guaranteed to be less than this upper bound.	int	.max	Data
Delay (ms)	The delay in milliseconds between timer events.	int	.delay	Behavior
Initial Delay (ms)	The delay in milliseconds before the first event when running is set to true.	int	.initialDelay	Behavior
Name	The name of this component.	String	.name	Common
Running?	Determines whether or not the timer sends timer events.	boolean	.running	Behavior
Step by	The amount added to the value each time this timer fires for use as a counter. (should be positive)	int	.step	Data
Value	The current value of this timer, for use as a counter. At each iteration, this value will be set to ((value + step) MOD bound)	int	.value	Data

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

Event Handlers

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

Customizers

This component does not have any custom properties.

Example

Expression Binding Example

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

Vision - Signal Generator



Component Palette Icon:



Description

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

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

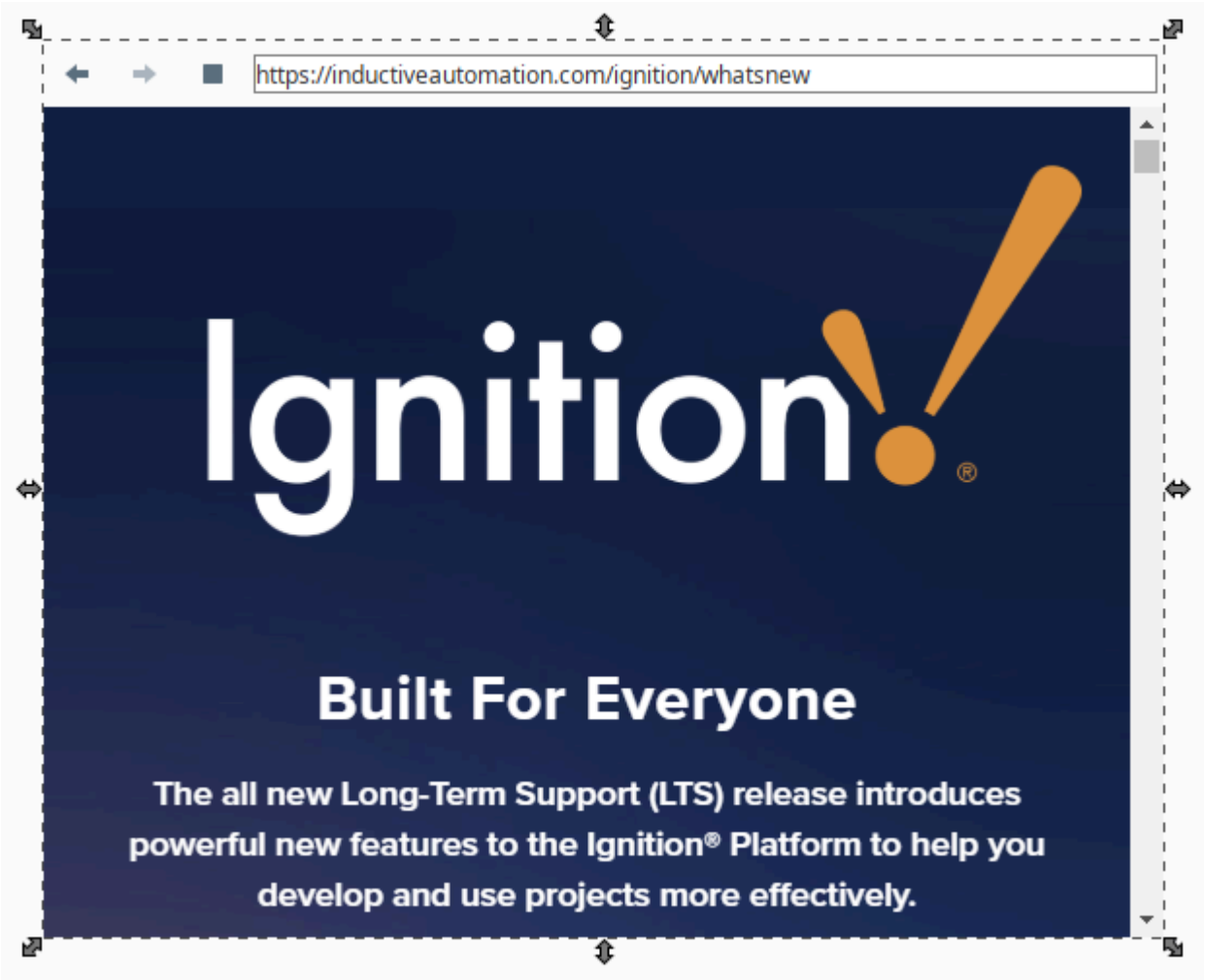
Event Handlers

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

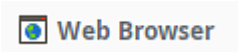
Customizers

This component does not have any custom properties.

Vision - Web Browser



Component Palette Icon



Description

The **Web Browser** component in Designer allows you to embed a full web browser inside of an Ignition Client from the internet or from your local intranet inside of your projects. The browser is based on the open source project Chromium, and supports modern web technology such as HTML5, CSS, and Javascript.

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

This component utilizes JxBrowser. As a result it will only run on 64-bit operating systems that are supported by JxBrowser (ARM OS's are not supported at this time): **https://jxbrowser-support.teamdev.com/docs/guides/introduction/requirements.html**

REQUIRED LINUX LIBRARIES

Missing Libraries

The Web Browser Component depends on an embedded version of the Chromium browser. Required dependencies on some Linux distributions, such `libgcrypt` and `libgconf-2-4`, are not bundled with the embedded browser and must be separately installed on the system. Running the following commands as applicable can resolve the issue:

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

```
sudo apt install libgcrypt20
```

Consult the Chromium installation instructions for your distribution for more detailed information.

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), and Field Border. **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	int	.httpProxyPort	Data

Name	Description	Property Type	Scripting	Category
	for HTTP connections. This setting is only used when **Use Proxies** is checked.			
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

Name	Description	Property Type	Scripting	Category
Mode	Data source for browser. Mode controls whether **Starting URL** or **Starting HTML** will be used.	int	.mode	Data
Name	The name of this component.	String	.name	Common
Popups Allowed	This flag is used to allow popups in the web page displayed.	boolean	.popupsAllowed	Behavior
Proxy Exceptions	A comma delimited list of rules for websites that will bypass the proxy servers. An example sting would be `*foo.com, <local>,127.0.1`. This setting is only used when **Use Proxies** is checked.	String	.proxyExceptions	Data
Proxy Password	The password to use for proxy authentication. This setting is only used when **Use Proxies** and **Use Proxy Authentication** are checked.	String	.proxyPassword	Data
Proxy Username	The username to use for proxy	String	.proxyUsername	Data

Name	Description	Property Type	Scripting	Category
	authentication. This setting is only used when **Use Proxies** and **Use Proxy Authentication** are checked.			
SOCKS Proxy Port	The port number for SOCKS proxies.	int	.socksProxyPort	
SOCKS Proxy Server	The host name to use for SOCKS proxies. Can be empty.	String	.socksProxyServer	
Show Navigation Buttons	Show the navigation buttons at the top of the frame.	boolean	.showNavigation	Behavior
Starting HTML	The initial HTML displayed when the Mode is set to HTML. Starting HTML is `<html><body></body></html>` by default, which gives a blank page.	String	.startingHtml	Data
Starting URL	The initial URL displayed when the Mode is set to URL. Starting URL is blank by default.	String	.startingUrl	Data
Touchscreen Mode	Controls when this input components responds if	int	.touchscreenMode	Behavior

Name	Description	Property Type	Scripting	Category
	touchscreen mode is enabled.			
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

See the **Vision - Web Browser Component Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

- **Vision Component Customizers**

Examples

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.

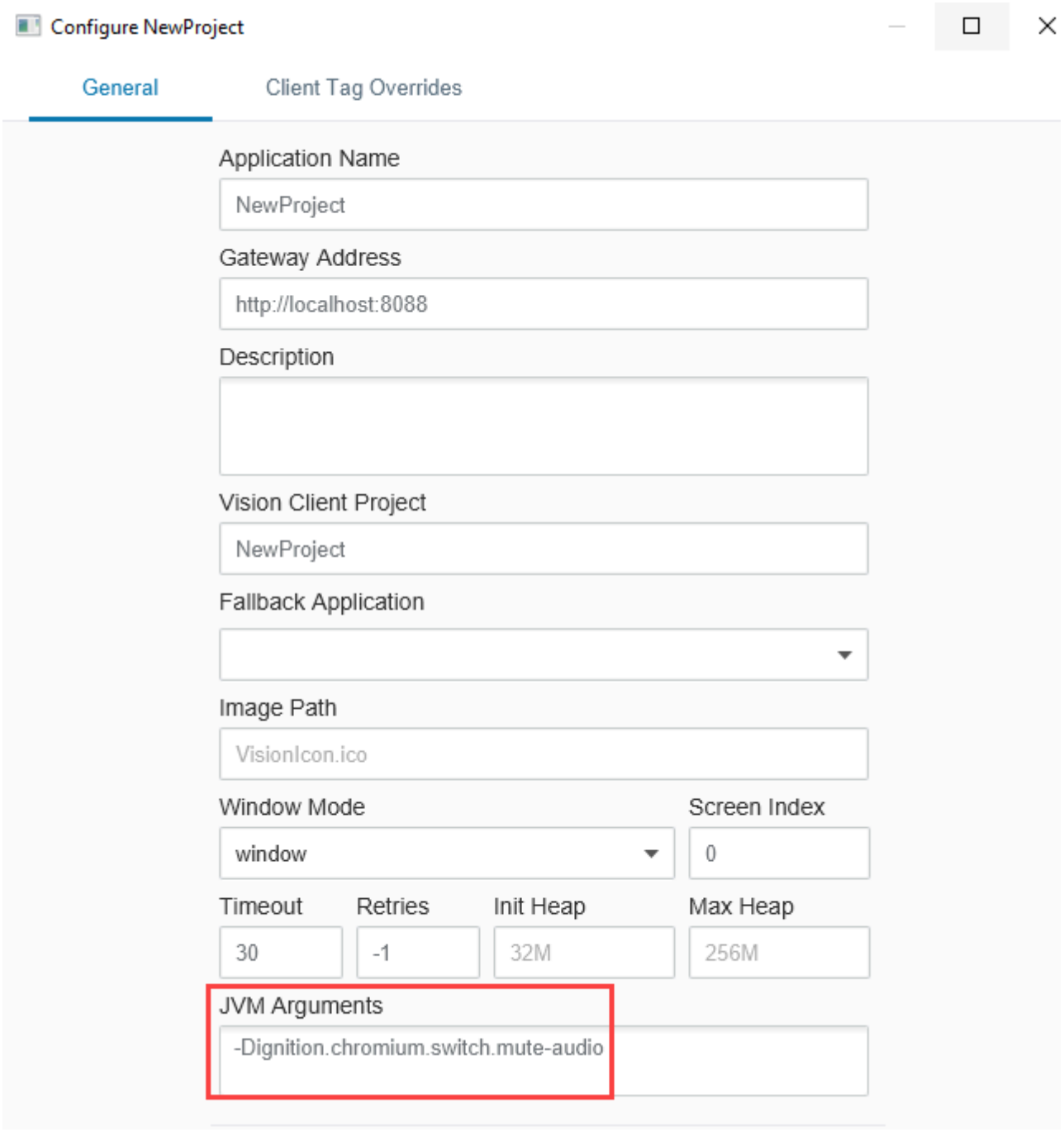
If you're going to make use of a switch, then you would do so on the Designer Launcher's/Vision Client Launcher application, under the JVM Arguments field. Below is an example on how to configure a switch for a client using the Vision Client Launcher. The same method applies for the Designer Launcher.

1. Open the Vision Client Launcher.

2. Create a new application or **manage the settings** on an existing application.

3. 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 - Web Browser Scripting Functions

This page details the various component and extension functions available for **Vision's Web Browser** component.

Component Functions

.getBrowser()

- Description
  - This function will return the underlying browser object. See **JxBrowser Guidelines** for more information.
- Parameters
  - None
- Return
  - Object - The Browser Object.

.executeJavaScript()

- Description
  - This function allows users to execute arbitrary JavaScript on the loaded page.
- Parameters
  - String **javaScript** - The code to execute on the page.
- Return
  - None

.getImage()

- Description
  - This function will return a byte array screenshot of the current browser window, in JPEG format.
- Parameters
  - None
- Return
  - ByteArray - The current browser window, rendered as a JPEG, in binary format.

.back()

- Description
  - This function navigates one page back in the browser history.
- Parameters
  - None
- Return
  - None

.forward()

- Description
  - This function navigates one page forward in the browser history.
- Parameters
  - None
- Return
  - None

.refresh()

- Description
  - This function refreshes the current page.
- Parameters
  - None
- Return
  - None

Extension Functions

initialize()

- Description
  - Called when the Web Browser component is initialized. Provides a chance to initialize the browser further. Enabling or disabling this function will cause the Web Browser component to re-initialize.
- Parameters
  - Component **self**: A reference to the component that is invoking this function.
  - JxBrowser **browser**: The underlying JxBrowser instance of the Browser class.
  - BrowserView **browserView**: The underlying rendering class that contains the Browser instance.
- Return
  - Nothing

Vision - Reporting Palette

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

Report Viewer

Row Selector

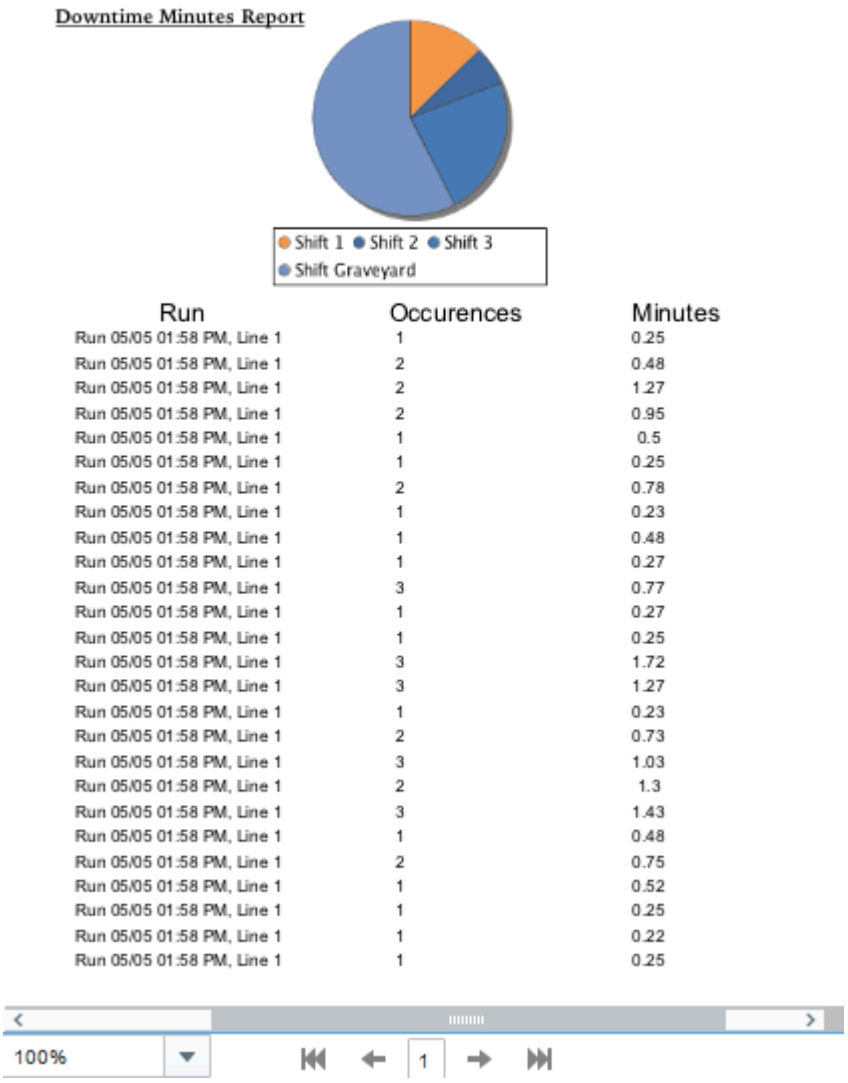
Column Selector

File Explorer

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

PDF Viewer

Vision - Report Viewer



Component Palette Icon:



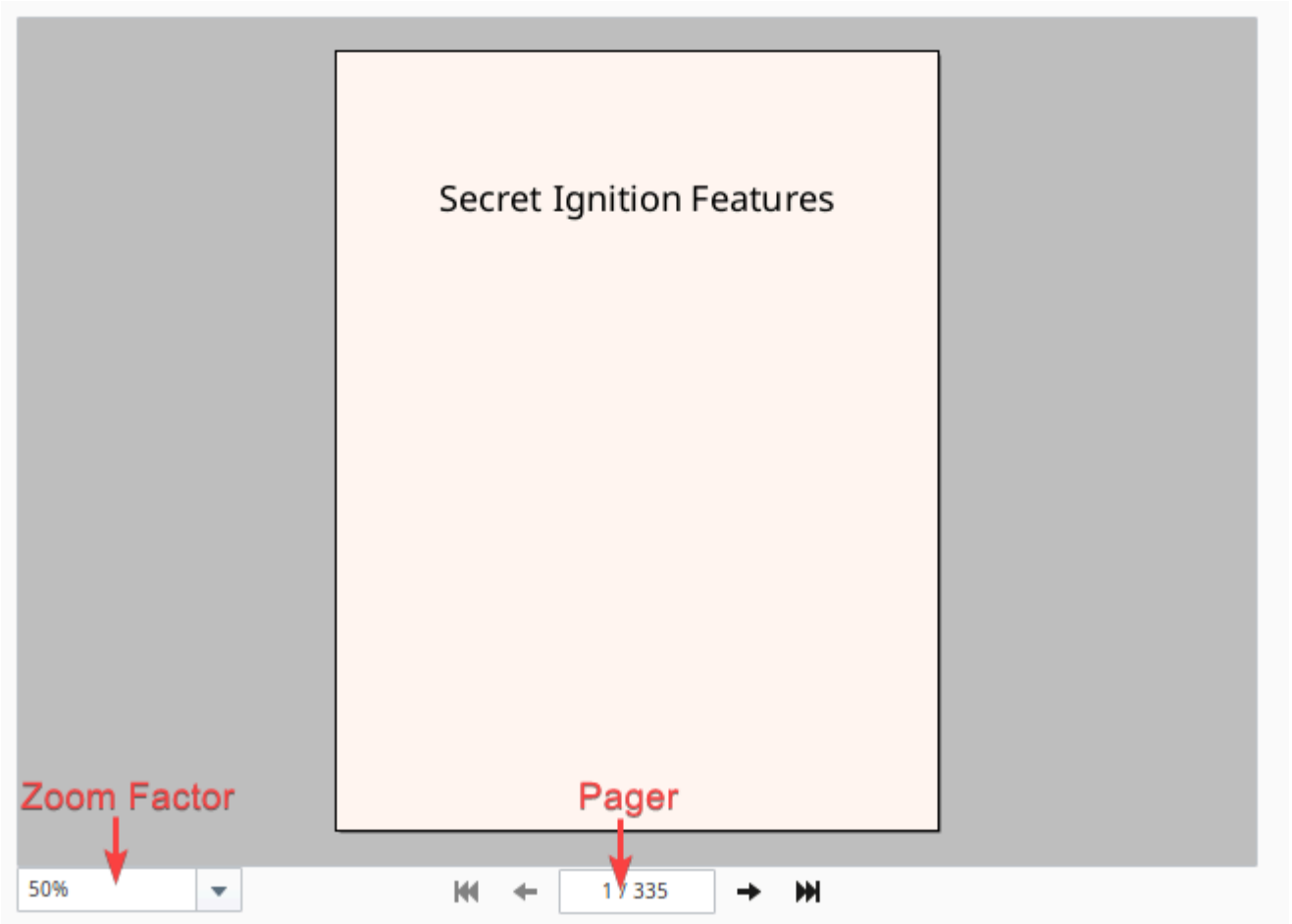
Description

The Report Viewer component provides a way to run and view Reports in Vision windows. Parameters added during Report creation are provided as Properties in the Viewer and can override any default values set in the Report Resource. Right clicking on the Report Viewer brings up a menu that allows you to easily print the report or save it in various formats.

To begin using the Report Viewer, the project must first have at least one report configured. Once a report exists in the project, then the **Report Path** property on the Report Viewer can be used to select the report, which will cause the component to render the selected report. If the report has any

parameters, those will be exposed under the **Report Parameters** category in the Property Editor, allowing you to configure bindings on them.

Interface Elements



Element	Description
Zoom Factor	Determines the zoom level of the rendered report.
Pager	Determines which page is currently shown in the report viewer.
Right Click	Right-clicking on the rendered report will present a popup menu, allowing the user to save a copy of the report, or print it out to an available printer.

Properties

Property	Description	Property 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), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Current Page	Current page in the report you would like to view.	Int	.currentPage	Data
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

Property	Description	Property Type	Scripting	Category
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
Suggested Filename	The filename that will come up by default when the user saves the report to disk.	String	.suggestedFilename	Behavior

Property	Description	Property Type	Scripting	Category
Visible	If disabled, the component will be hidden.	Boolean	.visible	Common
Zoom Factor	Zoom factor for the rendered report. This property directly controls the zoom factor interactive element displayed on the component.	Float	.zoomFactor	Data

Report Parameters Category

The Report Viewer component features a dynamic subset of properties under the **Report Parameters** category. This category is populated by **report parameters** that are defined on the reported specified by the **Report Path** property.

Scripting

See the **Vision - Report Viewer Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Examples

Utilizing reportLoading

```
```

```
# This example will check if the report has finished loading. If so, print the
report.

# Reference the report viewer
reportViewer = event.source.parent.getComponent('Report Viewer')

if reportViewer.reportLoading:
    system.vision.showWarning("The report is still loading. Please wait")
else:
    reportViewer.print()
```

Vision - Report Viewer Scripting Functions

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

Component Functions

.print(printerName, showDialog)

> **NOTE**
> The following print method will only work if a report has finished loading on the Report Viewer component.

- Description
  - Uses the named printer and determine if the print dialog window should appear or not.
- Parameters
  - String **printerName** - The name of the printer the report should be sent to. Will use the default printer if left blank. [optional]
  - Boolean **showDialog** - True if the dialog window should appear, False if the dialog window should be skipped. Will be true if left blank. [optional]
- Return
  - None

print() 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)
```

getBytesPDF()

- Description
  - Return the bytes of the generated report in the Report Viewer using PDF format.
- Parameters
  - None
- Return
  - Byte Array - The bytes of the report in PDF format.

getBytesPNG()

- Description
  - Return the bytes of the generated report in the Report Viewer using PNG format.
- Parameters
  - None
- Return
  - Byte Array - The bytes of the report in PNG format.

saveAsPDF(fileName)

- Description
  - Prompts the user to save a copy of the report as a PDF. Shows a file selection window with the extension set to PDF.
- Parameters
  - String **fileName** - A suggested filename to save the report as
- Return
  - None

saveAsPDF() Example

Code Snippet - 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")
```

saveAsXls(fileName)

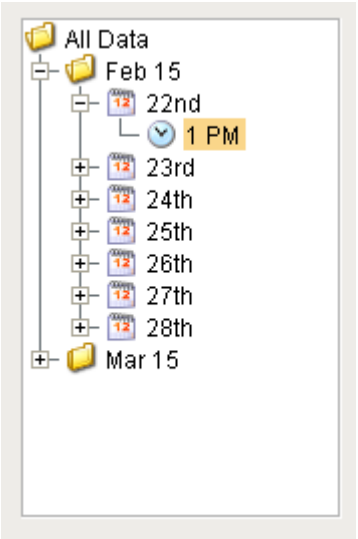
- Description
  - Prompts the user to save a copy of the report as an XLS file. Shows a file selection window with the extension set to XLS.
- Keyword Args
  - String **fileName** - A suggested filename to save the report as.
- Return
  - None

Extension Functions

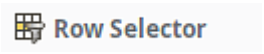
onReportGenerated

- Description
  - Called when the Report generation process has been completed.
- Keyword Args
  - Component **self** - A reference to the component invoking this method.
  - Byte Array **pdfBytes** - The PDF formatted bytes generated by the Report.
- Return
  - None

Vision - Row Selector



Component Palette Icon:



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.

Properties

Property	Description	Property Type	Scripting	Category
All Data Node Text	Text for the All Data node, if it is displayed.	String	.allDataNodeText	Appearance
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.background	Appearance
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field 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	Cursor	.cursor	Common

Property	Description	Property Type	Scripting	Category
	Resize, or SE Resize.			
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

Property	Description	Property Type	Scripting	Category
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

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

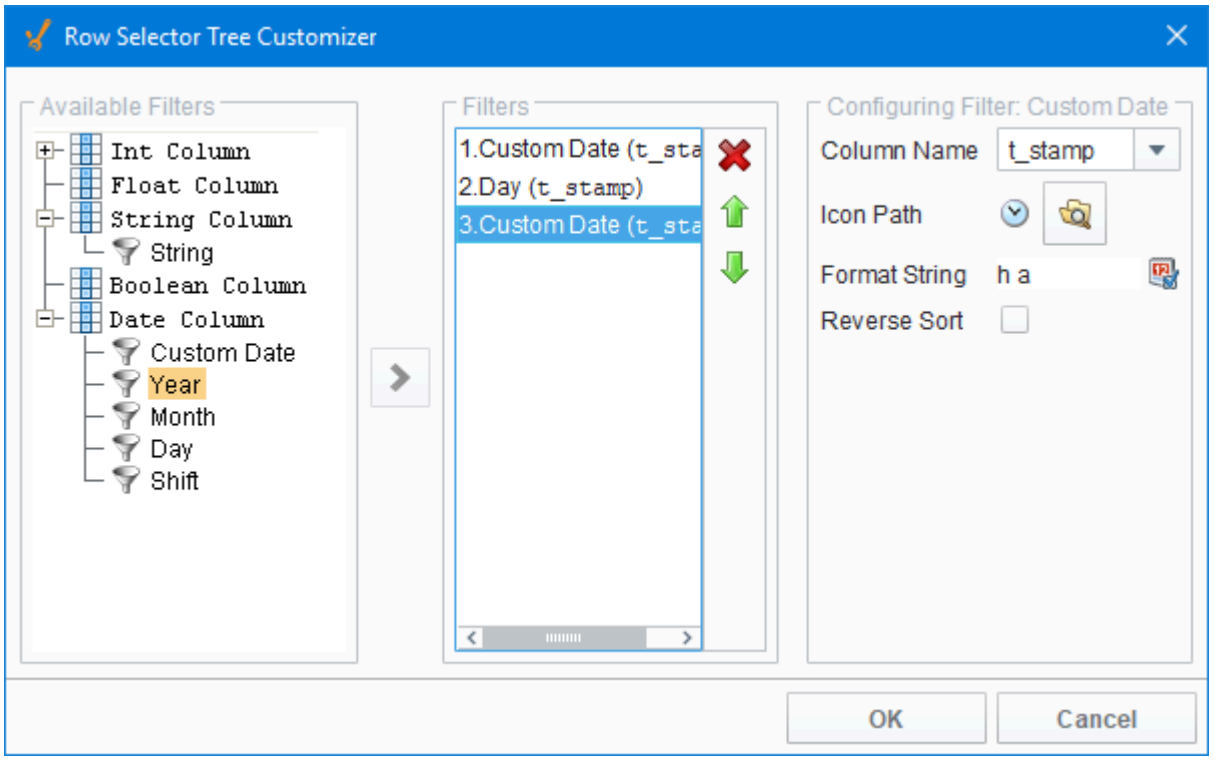
Customizers

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

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 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

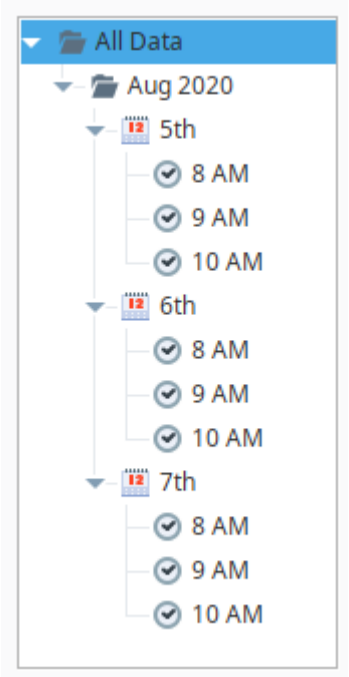
Examples

There are no examples associated with this component. Refer to the examples in the **Common Reporting Tasks**.

Using the Row Selector



The **Row Selector** component allows users to filter a dataset based on unique values of one or more columns. Each level in the sorting tree is based on these properties. The user will see a dynamically generated expandable tree that groups their data by any number of choices. As a user clicks down the tree, objects bound to the dataset will indicate the filtered data.



A common way to filter the data is by time. For example, you can feed the Row Selector an input dataset that represents a large time range, and have it break it up by Year, Month, and Day. Then you can power a report with the output dataset that lets the user dynamically create reports for any time range. When configuring the Row Selector for the first time, you'll notice some default Date filters in the Row Selector Tree Customizer to help you quickly configure and filter raw data by time. If you don't want to filter your data by time, then simply delete the default filters and create your own.

See the Appendix for more information on the **Row Selector**.

Example

A Row Selector has two important properties: **Data In** and **Data Out**. The Row Selector component filters the data in the **Data In** property and pushes the filtered result to the **Data Out** property. Let's

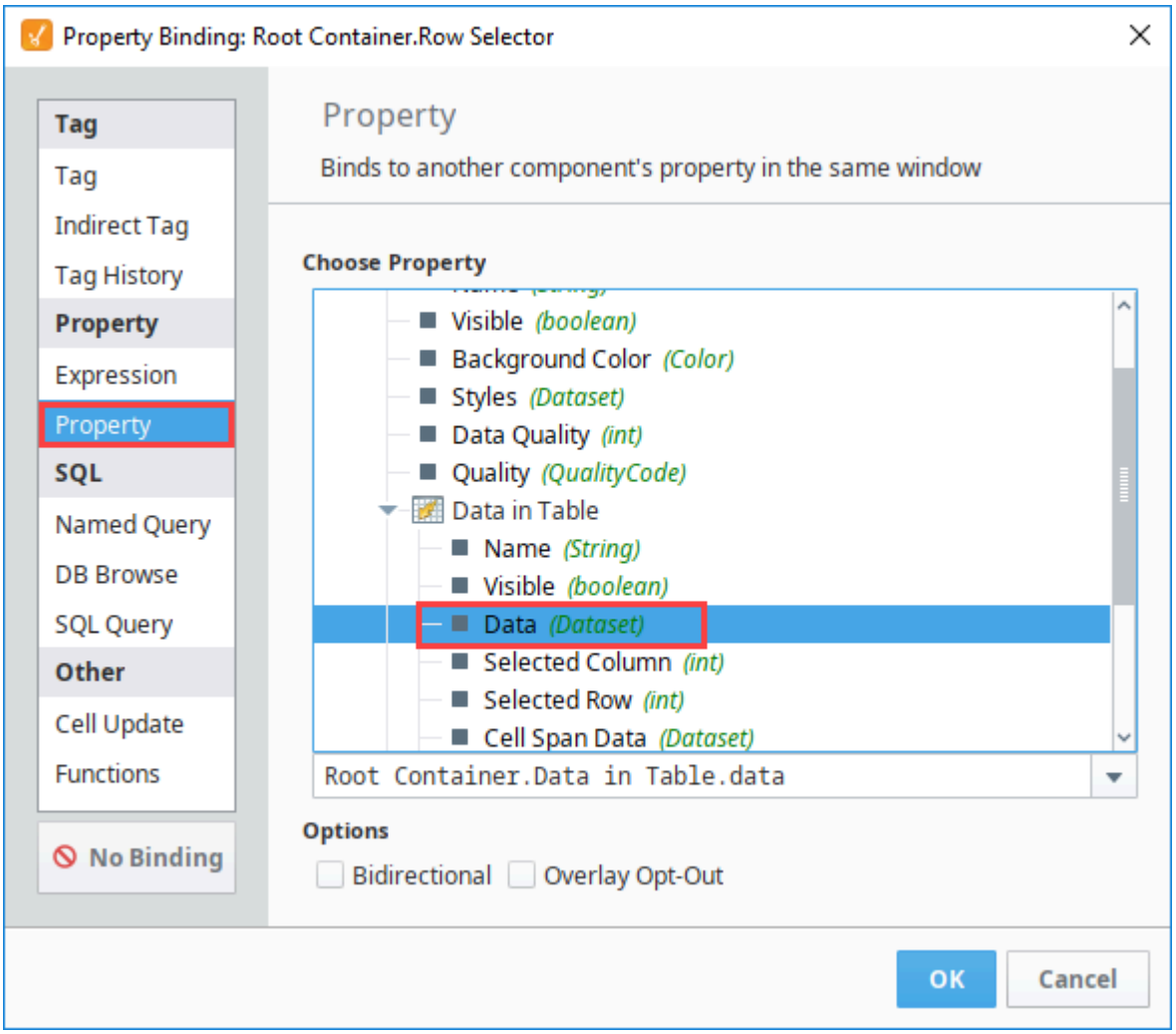
configure a Row Selector to filter on some raw data. (This example uses a Power Table component which gives you the option of using some sample test data, or you can create your own data).

1. Drag a Power Table component on to your window and change the Name of the Power Table to "**Data In Table**".

2. Scroll down the Property Editor and set the **TestData** property to '**true**'. This will populate the **Data** property with some test data. Alternatively, you could manually populate the **Data** property using either the Dataset Viewer or by creating a binding on the property.

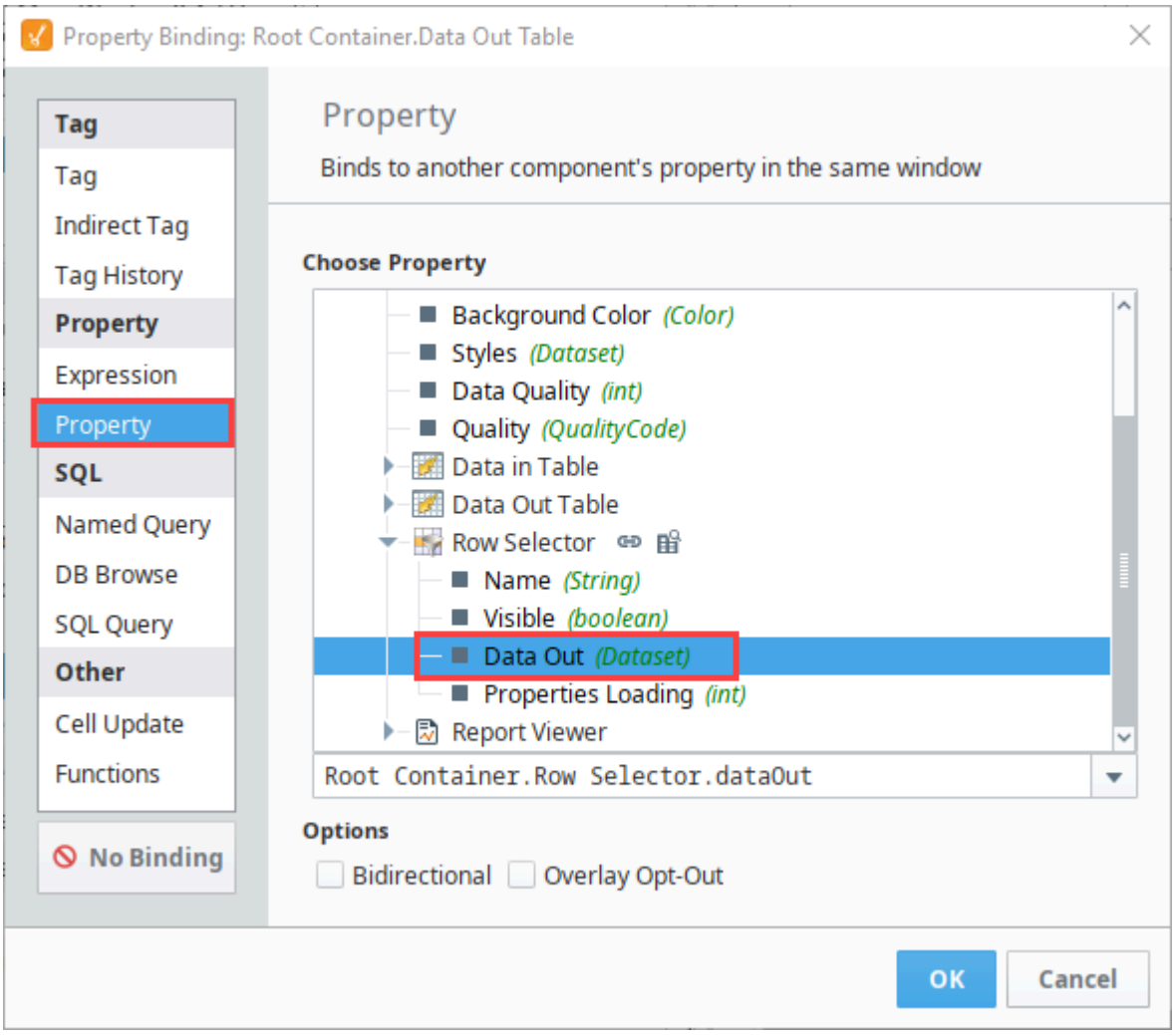
3. Drag a Row Selector component on to your window. With the Row Selector selected, click the **Binding** icon on the Row Selector's **Data In** property.

4. Select the **Property** Binding Type, and bind it to the **Data** property in the **Data In Table**, and click **OK**.



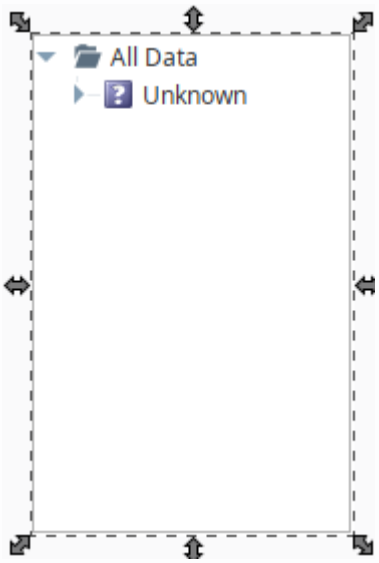
5. Drag another Power Table component on to your window, change the Name to "**Data Out Table**".

6. With the **Data Table Out** still selected, bind the **Data** property to the Row Selector's **Data Out** property and click **OK**.

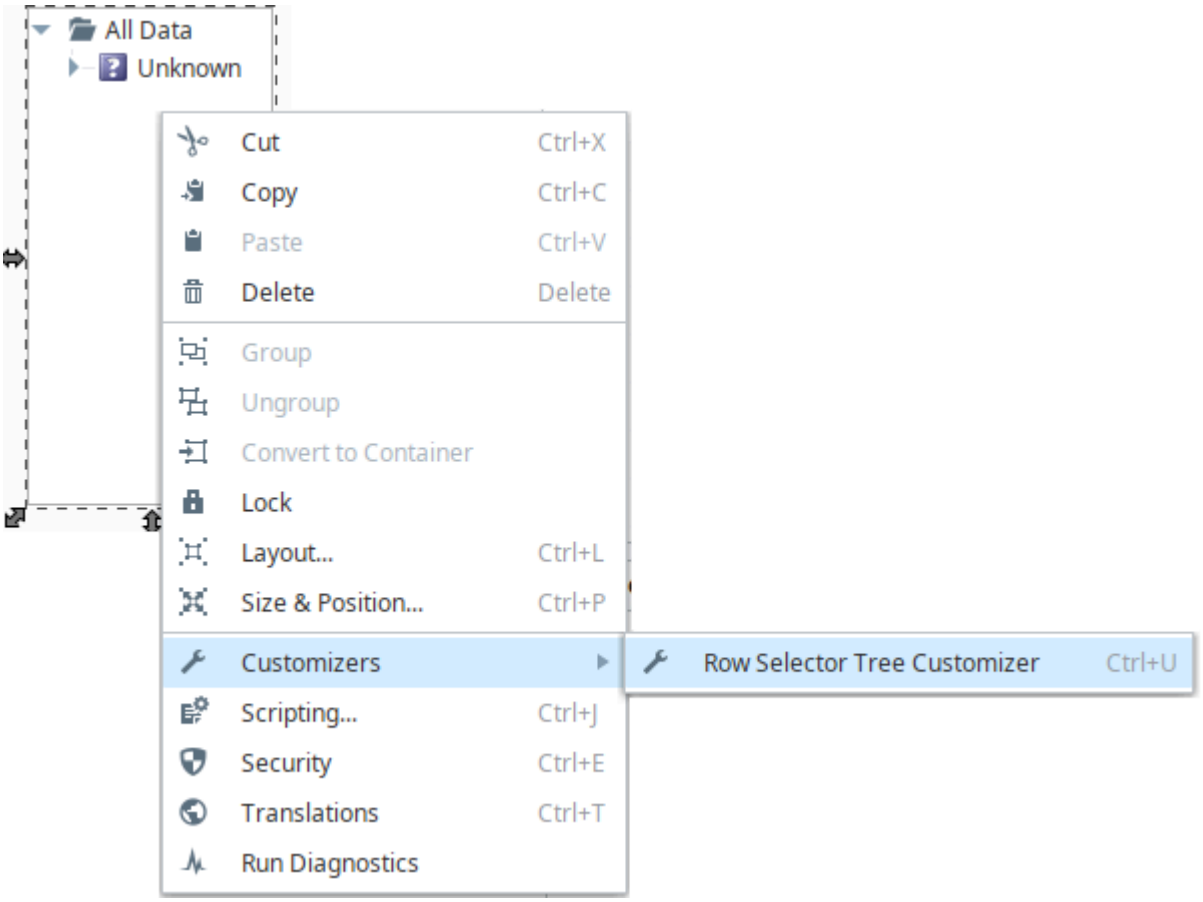


NOTE

Don't worry about the 'Unknown' nodes in the Row Selector component. The Row Selector has three default Date filters which haven't been configured for your data yet. You will need to reconfigure them, or delete them and configure your own unique set of filters. Once configured, the 'Unknown' nodes will disappear.



7. Now let's configure your filters. Right click the Row Selector component and scroll down to **Customizers > Row Selector Tree Customizer**.

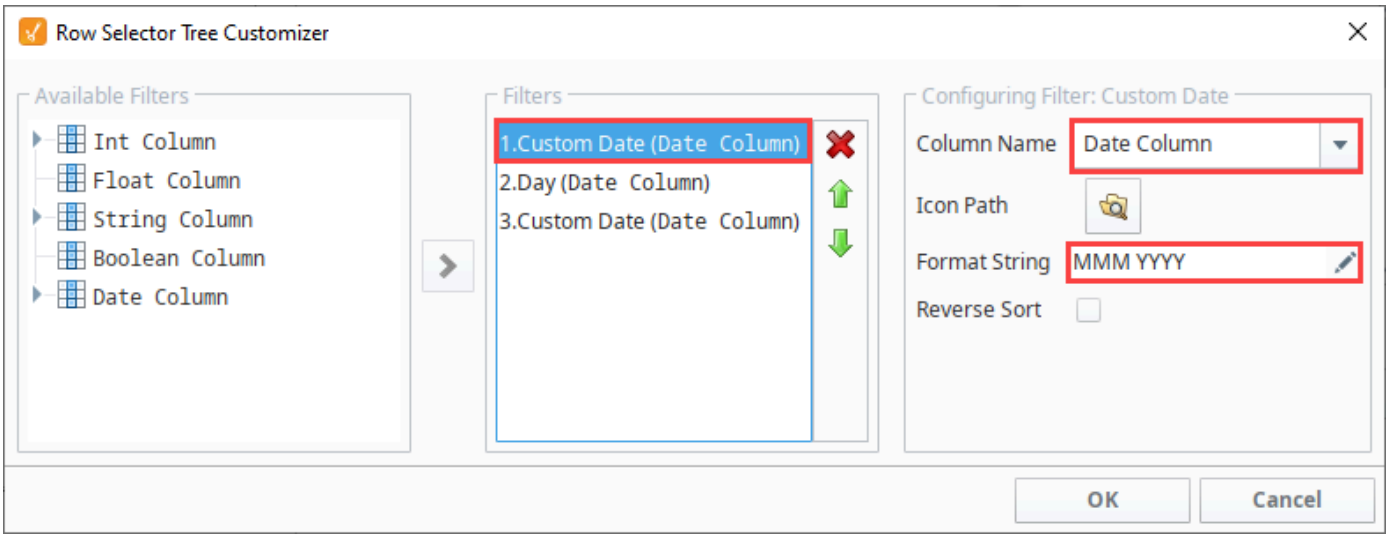


   The Row Selector Tree Customizer provides three default Date filters. Here you can customize the parameters of each Date filter or choose another filter type that is more appropriate for your dataset. Each column on the Data In Table will be listed in the Available Filters tree, and the types of filters available to each column depends on the datatype of the column.

8. To start customizing, select the first filter, then change the **Column Name** to Data Column. (This example uses the **Date** column to filter on the Month, Day, and Time combination.)

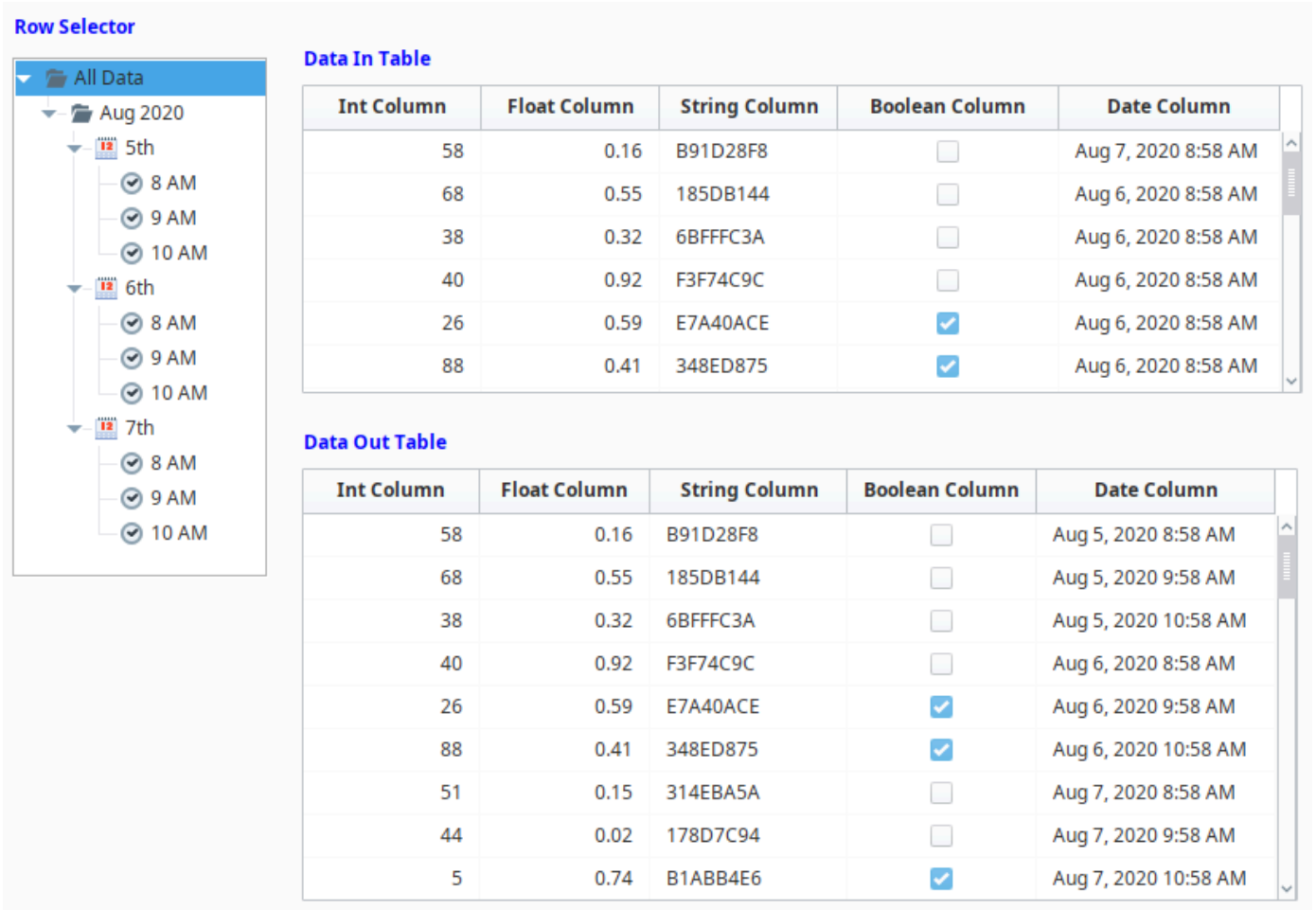
9. Change the **Format String** to MMMM yyyy.

10. Click **OK**.



11. Put your Designer into **Preview Mode**.

12. Select Month, Day, or Time to filter on. The filtered results are displayed in the Data Out Table. In this example, we filtered on June 20th, so the Data Out Table only contains records that match that date.



NOTE

> When designing your report window, it's not necessary to display the Data In Table only the Data Out Table. You also don't need a component to house the data: the Data In property on the Row Selector could simply retrieve the raw data with a binding.

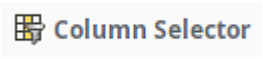
Here are a few more Row Selector examples:

- A Line Graph bound to a Row Selector - Set up grouping to be first by month and year, then day, then hour, like the example above. Clicking on a month and year will dynamically update the graph for that time period. Further clicking to a specific day or hour will re-filter the graph for that period.
- A **Report Viewer** bound to a Row Selector - Grouping by department (String) would allow selection by department, automatically regenerating the Report on selection.
- An "Alarm History" Table bound to a Row Selector - This could first be broken down severity level (Integer), then broken into "Alarm Acknowledged" / "Not Acknowledged" (Boolean based). Clicking "Severity 3" would filter the table to all Severity 3 alarms. Selecting "Unacknowledged" would then filter the table to Unacknowledged alarms of Severity 3.

Vision - Column Selector



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.

Properties

Property	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), and Field 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	Dataset	.Data_in	Custom Properties

Property	Description	Property Type	Scripting	Category
	configuration and there may be more of these input dataset 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

Property	Description	Property Type	Scripting	Category
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

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

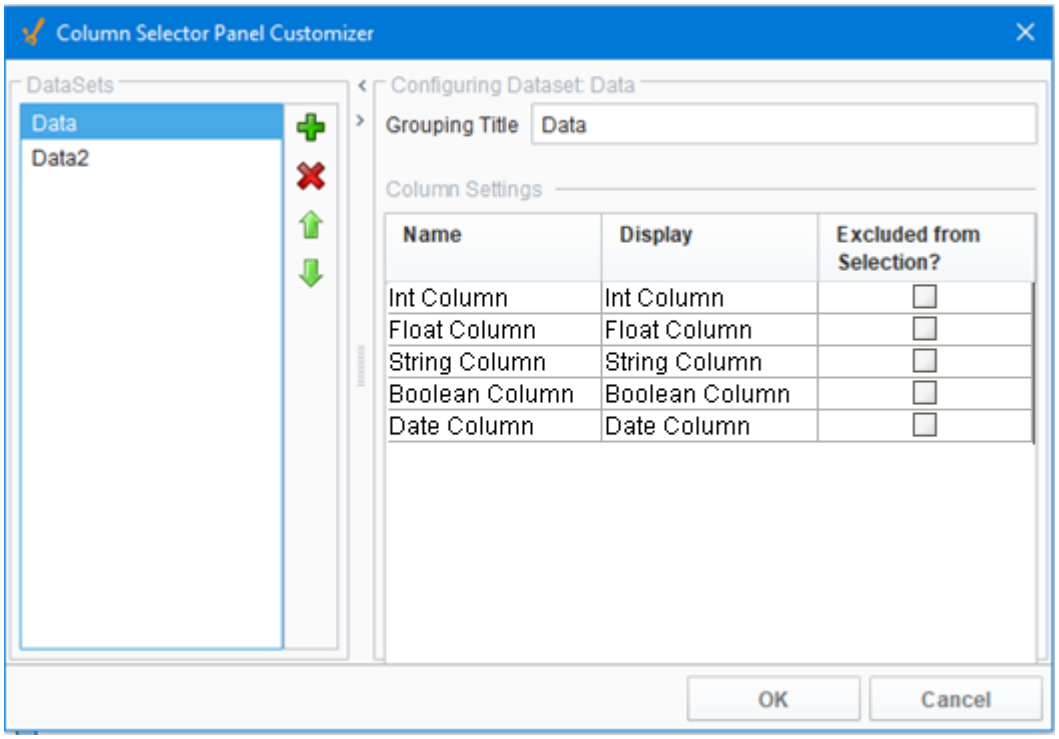
Event Handlers

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

Customizers

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

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.

Using the Column Selector



The Column Selector is similar to the Row Selector except that instead of filtering rows, it filters out entire columns from the output dataset. Each column from the input dataset is shown as a checkbox and allows users to show or hide variables in the datasets via the checkboxes, then output the resulting dataset.

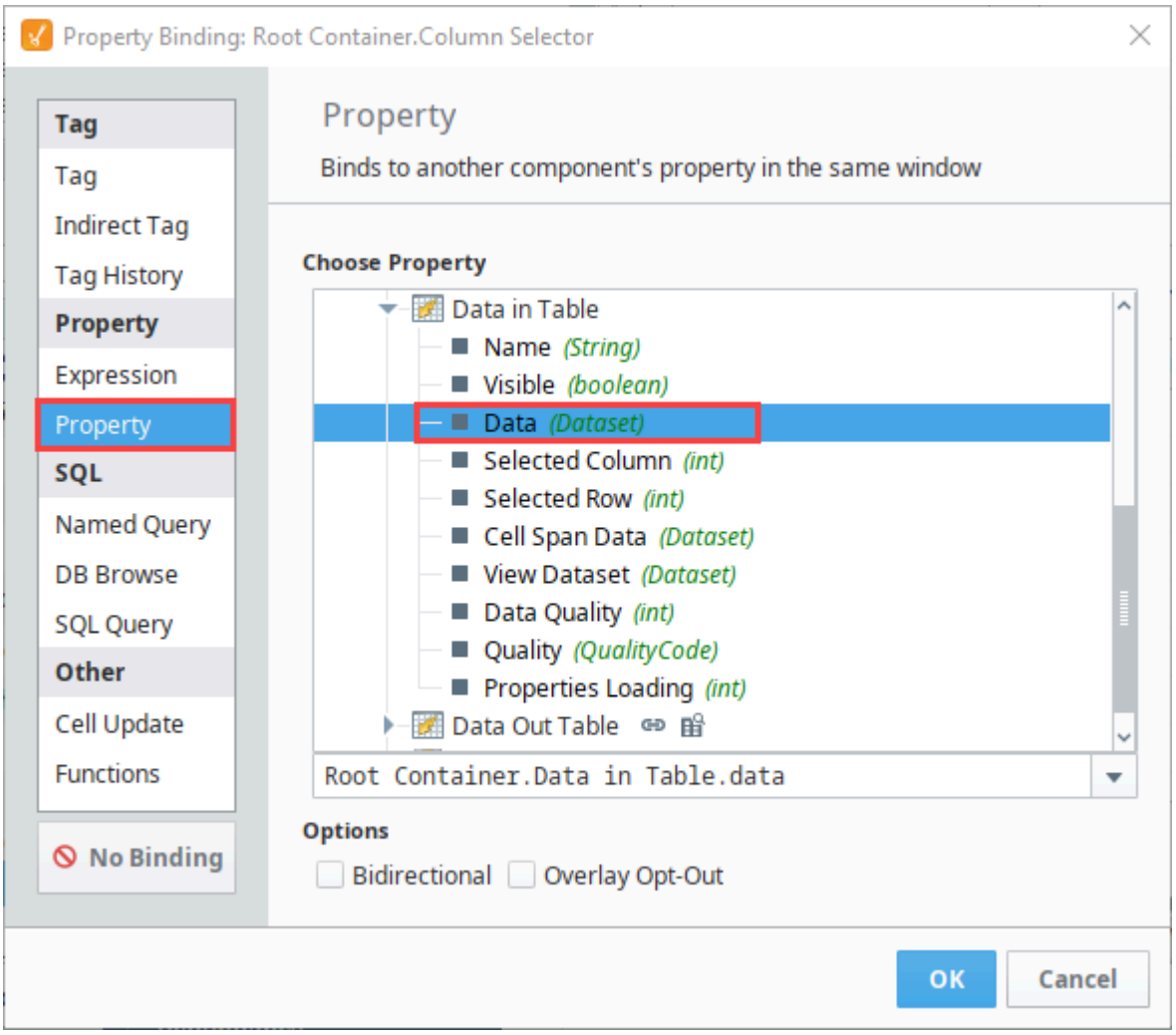
The Column Selector allows users to choose which columns in a dataset they wish to use. If an object is bound to the Column Selector it will update itself whenever a user checks or unchecks a column. This allows users to dynamically show or hide Table columns, "pens" on a graph, data in a **Report Viewer**, or any other component set up to use a dataset.

1. Drag a Power Table component on to your window and change the **Name** of the Power Table to "**Data In Table**".

2. Scroll down the Property Editor and set the **TestData** property to True. This will populate the **Data** property with some test data.

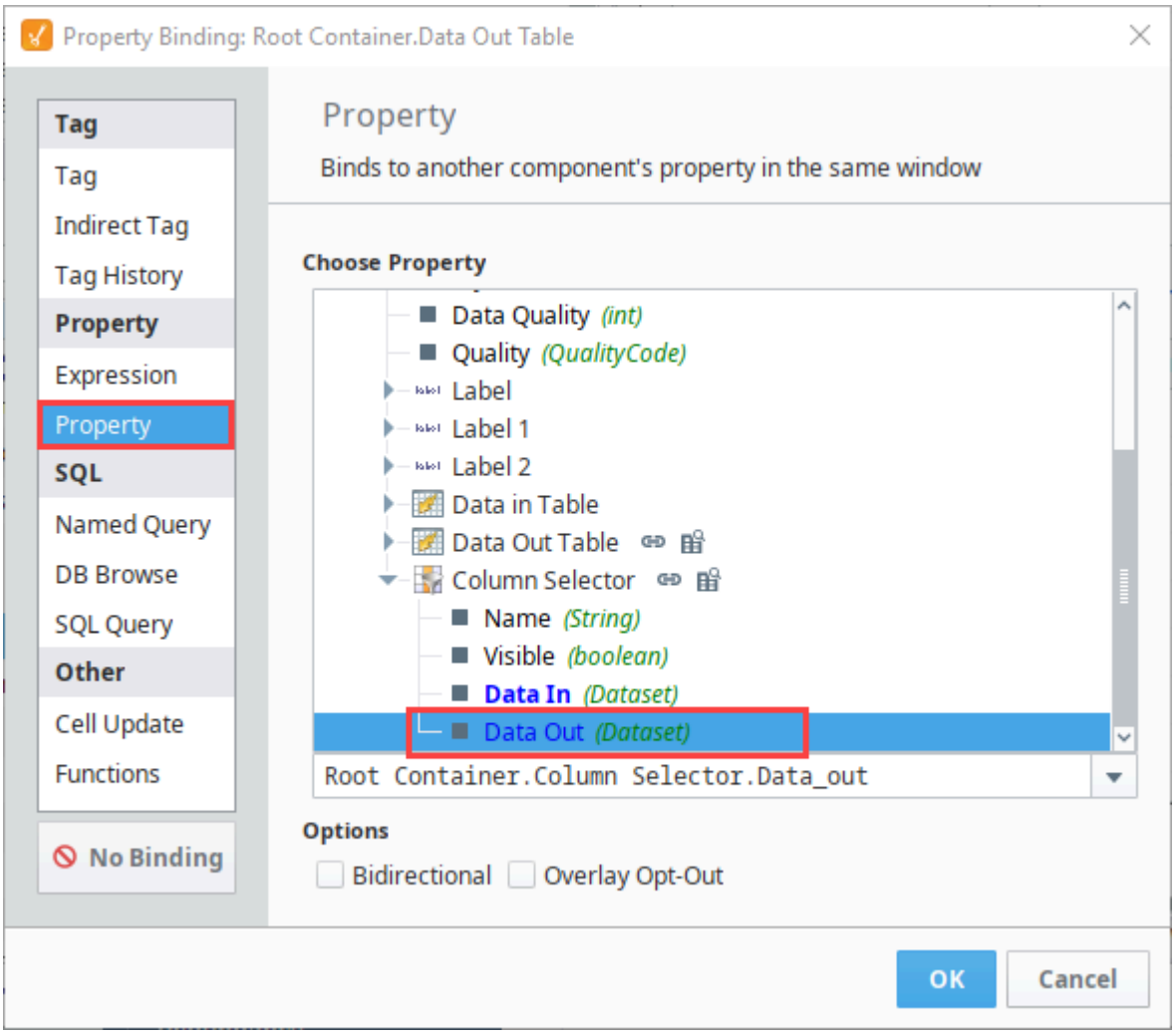
3. Drag a Column Selector component on to your window. With the Column Selector selected, click the **Binding** icon on the Column Selector's **Data In** property.

4. Select the **Property** Binding Type and bind it to the **Data** property in the **Data In Table** and click **OK**.



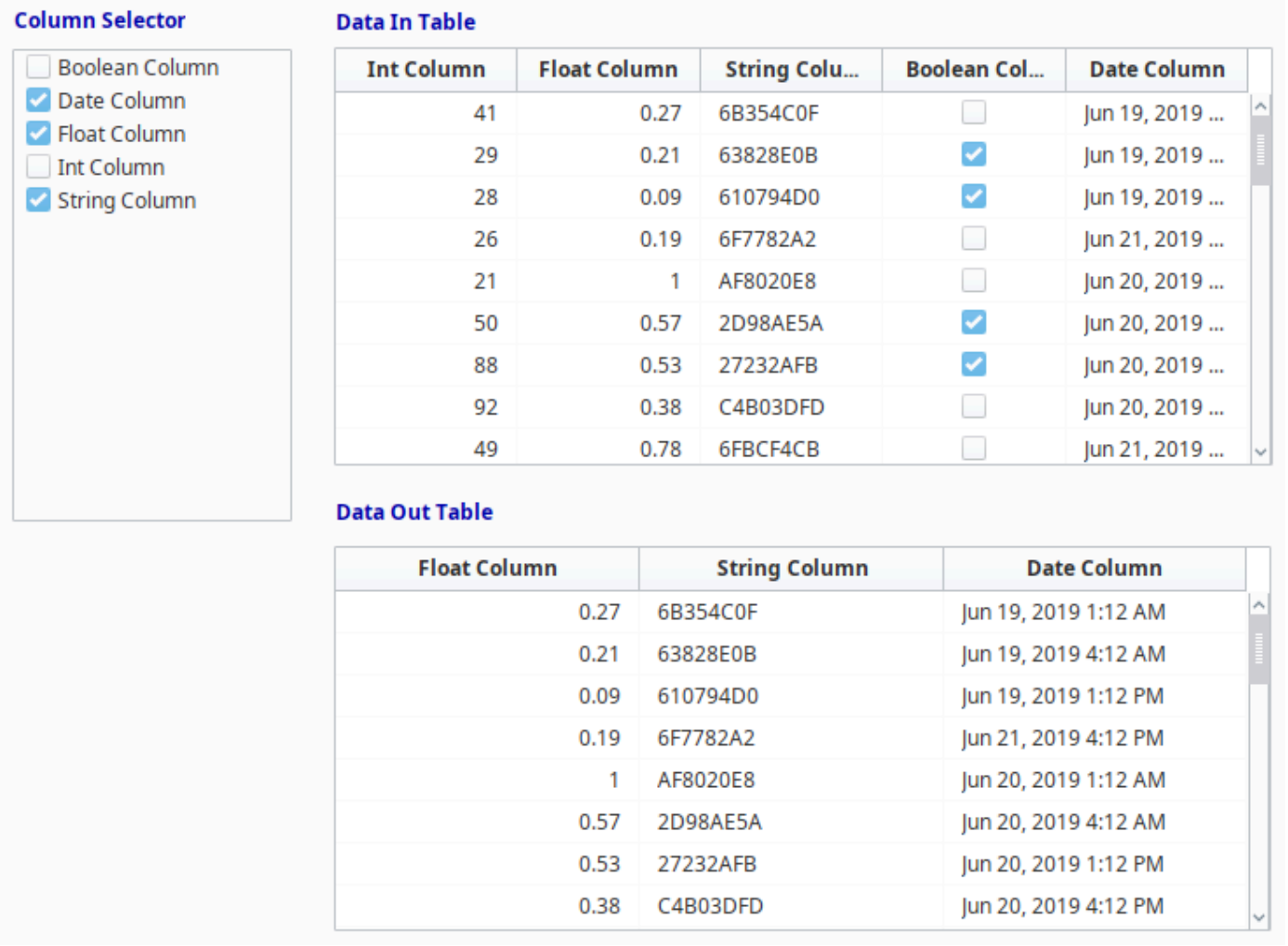
5. Drag another Power Table component on to your window, change the Name to "**Data Out Table**".

6. Bind the **Data** of the **Data Out Table** property to the Column Selector **Data Out** property and click **OK**.



7. Put your Designer into **Preview Mode**. In Preview Mode, you can can hide specific columns by unchecking the boxes in the Column Selector.

8. Use the check boxes to select columns you want displayed or hidden. In this example, we hid the **Boolean** and **Int** columns as shown in the Data Out Table.

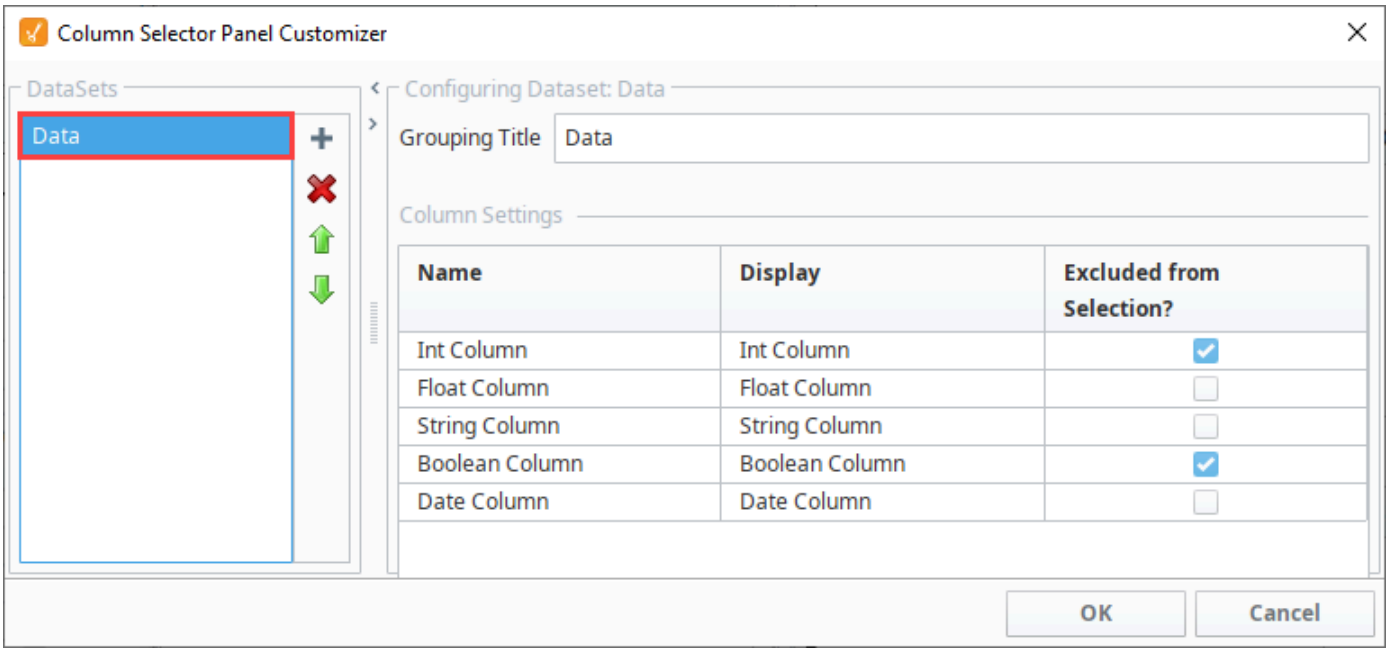


9. If you want to customize the column display for all users, put the Designer back into Design mode.

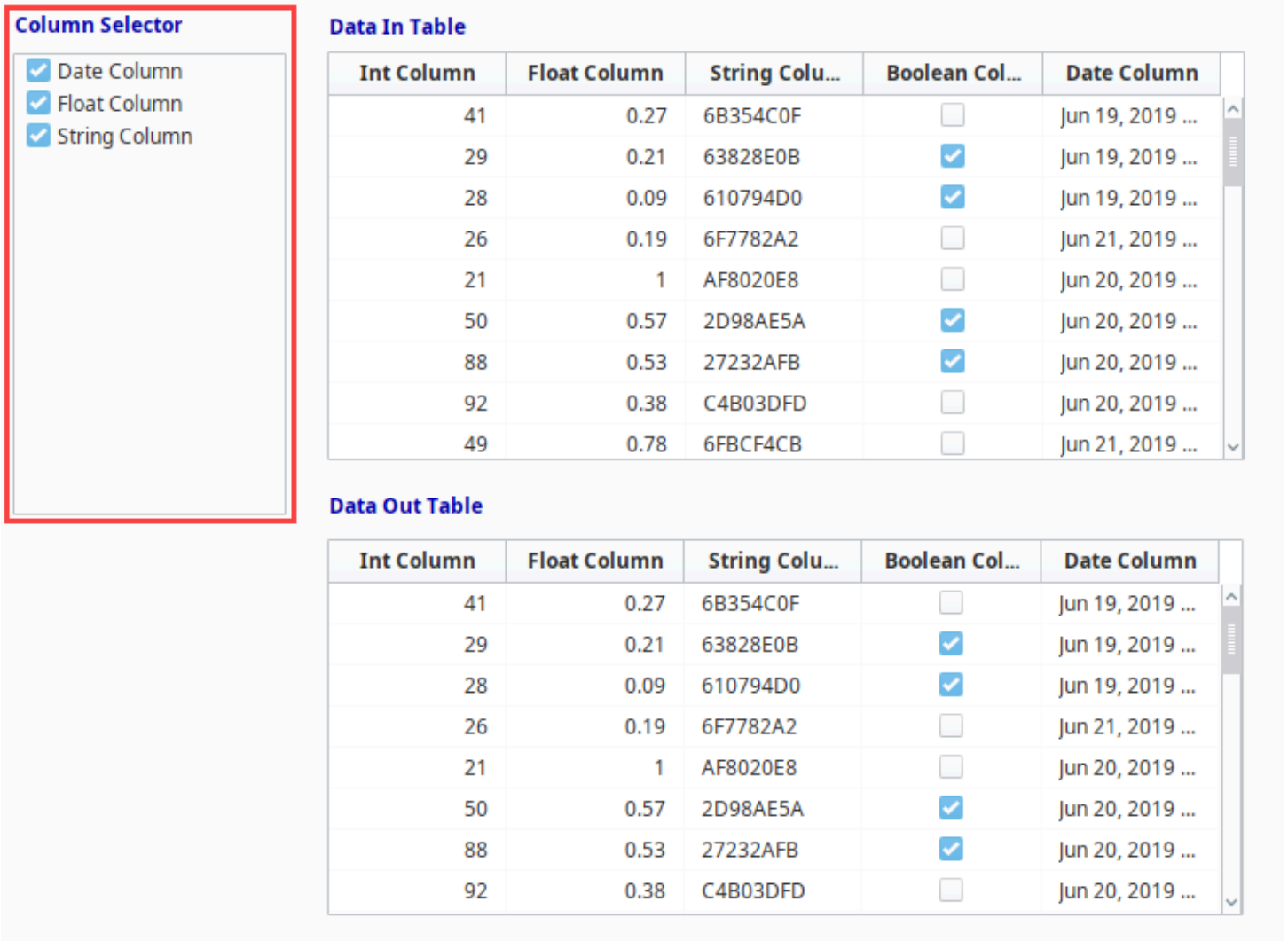
10. Right click on the Column Selector and choose **Customizers > Column Selector Customizer**.

11. Click the **Excluded from Selection** box next to any column that you don't want displayed. The "Excluded from Selection" option determines if the user is allowed to hide the column from the client via the Column Selector.

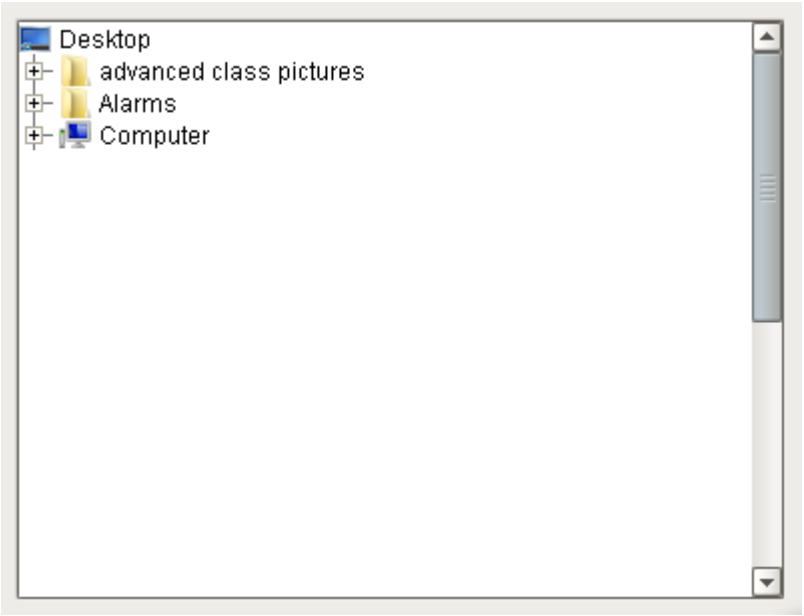
12. Click **OK**.



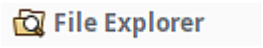
Now, the columns are excluded from the selection in the Customizer (Int and Boolean columns) and are not displayed in the Column Selector of the Client and prevented from being hidden by the user.



Vision - File Explorer



Component Palette Icon:



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 **PDF Viewer** page.

New in 8.1.14 Users can right-click on the File Explorer to refresh the component in a Vision Client.

Properties

Property	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), and Field 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
Enabled	If disabled, the component can't be used.	boolean	.componentEnabled	Common

Property	Description	Property Type	Scripting	Category
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 Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Root Directory	A directory to act as the root of the file explorer.	String	.rootDir	Behavior
Selected Path	The selected file or folder's path.	String	.selectedPath	Data
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

Component Functions

This component does not have component functions associated with it.

Extension Functions

This component does not have extension functions associated with it.

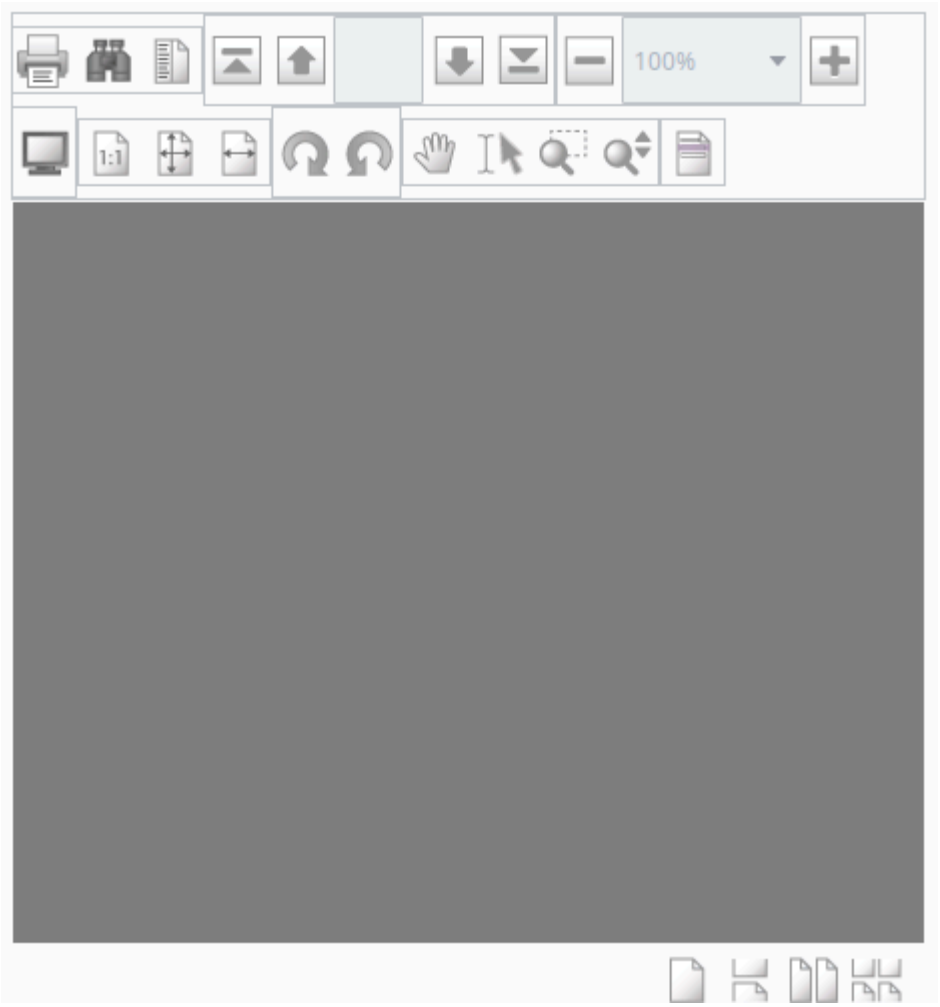
Event Handlers

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

Customizers

The File Explorer component does not have a customizer.

Vision - PDF Viewer



Component Palette Icon:



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**.

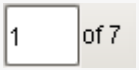
This component is typically used in conjunction with the File Explorer component, in order to create a PDF viewing window. Simply bind the Selected Path property in the PDF Viewer to the File Explorer's Selected Path property. See the **File Explorer's documentation** for further instructions on how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

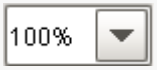
Properties

Property	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), and Field 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

Property	Description	Property Type	Scripting	Category
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
	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 Pane	Will show/hide the Utility panel. The Utility Panel contains the following tabs: • Search - Will search the document for a specific word or phrase. • Bookmarks - Will display all of the bookmarks for the loaded pdf and allow you to quickly jump to them. • Thumbnails - Will display a thumbnail view of all of the pages of the loaded pdf. Clicking on one will jump to it. • Annotations - Will create a multitude of annotations on the currently loaded pdf. After adding an annotation, it can be selected and then configured in the Utility Panel. Annotations include highlights, strike through, underlines, text notes, and actions like navigating to a url. • Layers - Will display the layers of the currently loaded pdf, if any.
	First Page	Will navigate back to the first page of the pdf.
	Previous Page	Will navigate back one page of the pdf.
	Current Page 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.

Toolbar Buttons	Name	Function
	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.

Toolbar Buttons	Name	Function
	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 Pane.
	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

See the **Vision - PDF Viewer Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Vision - PDF Viewer Scripting Functions

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

Component Functions

.loadPDFBytes(bytes, name)

- Description
  - This function will pass in the bytes of a PDF and load them into the PDF Viewer component. Please see **Storing Files in a Database** for more details
- Parameters
  - string **bytes** - The bytes of the PDF to be displayed on the component
  - string **name** - The name of the PDF
- Return
  - None

.print(showDialog)

- Description
  - This function will print the PDF.
- Parameters
  - boolean **showDialog** - If true, shows the user a print dialog. Default is true [optional]
- Return
  - None

.setZoomFactor(zoom)

- Description
  - This function will set the current zoom level of the PDF, adjusted to stay within the minimum / maximum zoom range. Will zoom in on center of page.
- Parameters
  - float **zoom** - Zoom factor to use. 1.0 is no zoom.
- Return
  - None

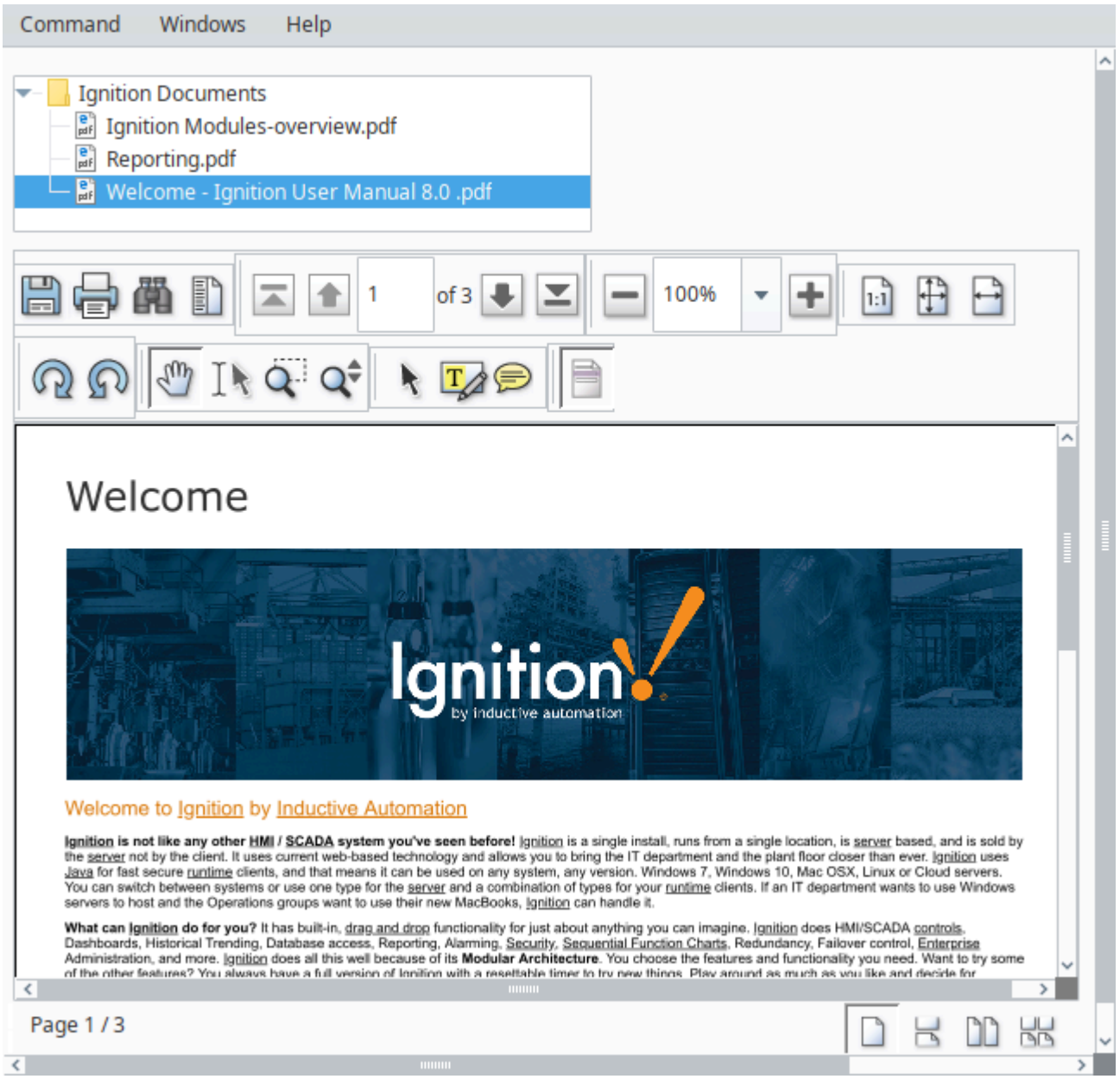
Extension Functions

This component does not have extension functions associated with it.

Using the PDF Viewer with the File Explorer Component



The **File Explorer** component displays a file system tree structure that allows users to navigate around various folders. It can be rooted to any folder including shared network folders, and can filter file types by their file extension like 'pdf.' The File Explorer is typically used in conjunction with the PDF Viewer component in order to create a PDF viewing window. This is very useful for viewing documents from within a project.

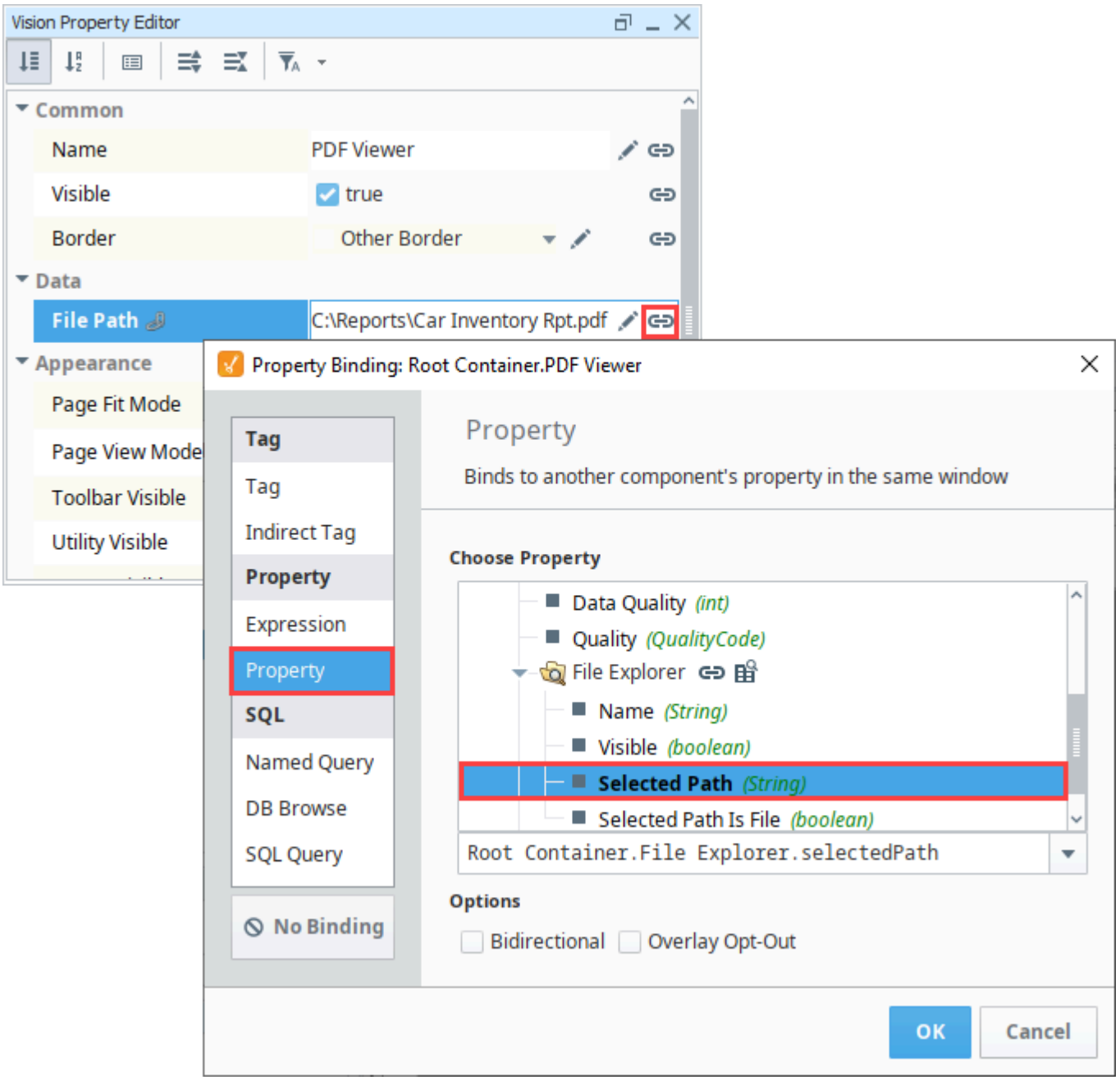


Let's set up the File Explorer and the PDF Viewer to create a window to view a PDF documents.

1. In the Designer, drag a **File Explorer** component and **PDF Viewer** component on to a window and place them side by side.

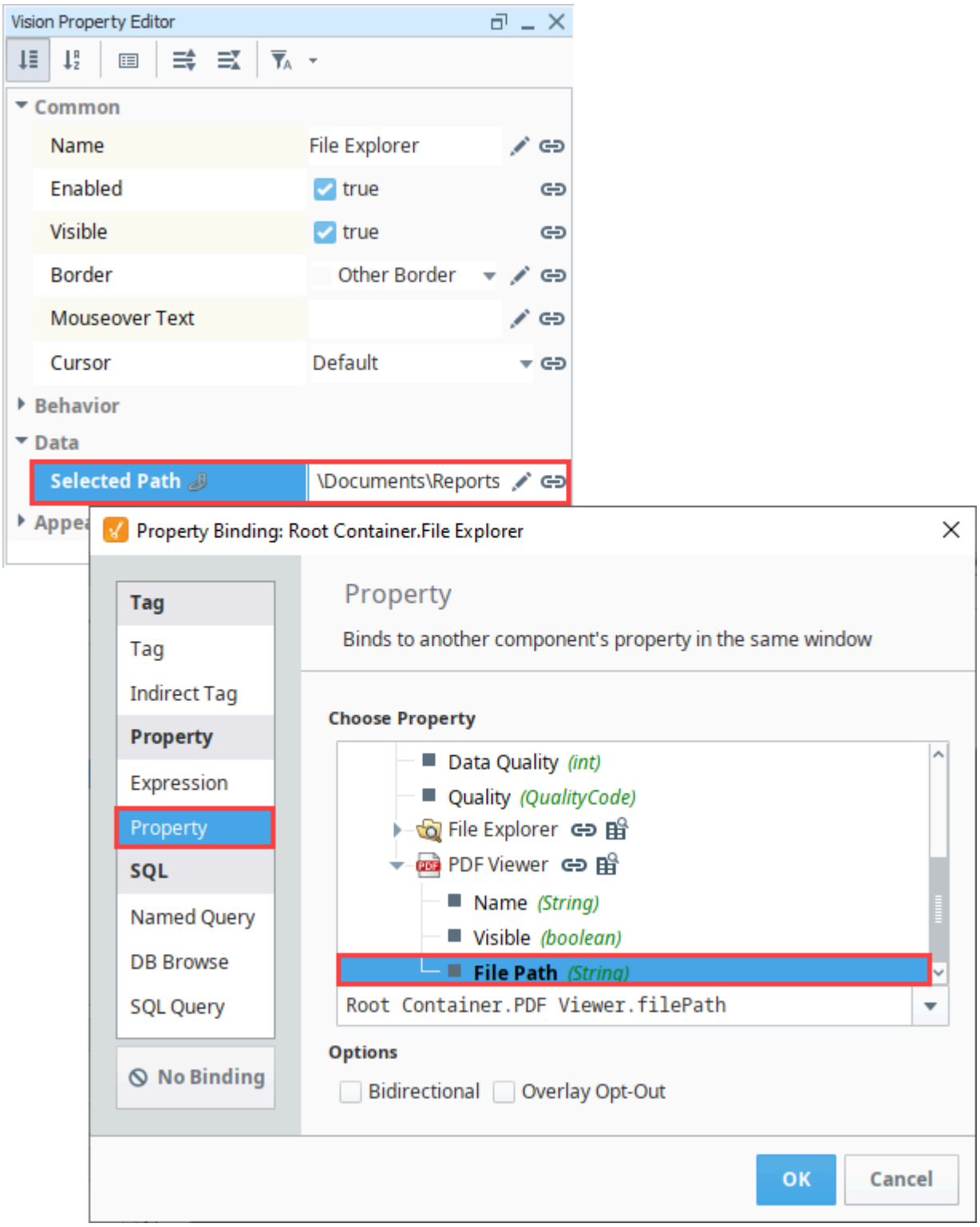
2. Select the **PDF Viewer** component and click the binding icon next to the **File Path** property.

3. Select the **Property** binding type, and drill down to the **File Explorer Selected Path** property. Click **OK**.

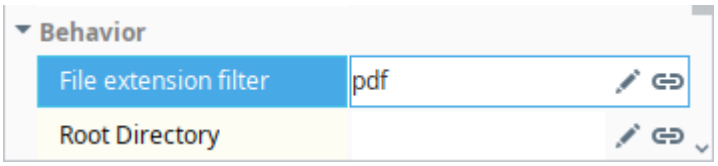


4. Next, select the **File Explorer** component. In the Vision Property Editor, set the **Selected Path** property to a folder path. You can type in a path or bind the root directory by clicking on the binding icon for the **Selected Path** as shown in the image below. The Root Directory restricts which directories are accessible on the component. This is typically used to restrict access to a particular folder so that the user doesn't have access to the entire file system. All folders nested within the Root Directory can be accessed on the component.

> **NOTE**
>
> If you set the File Explorer's Root Directory to a network folder, all clients will be able to access documents within all folders in that network folder.



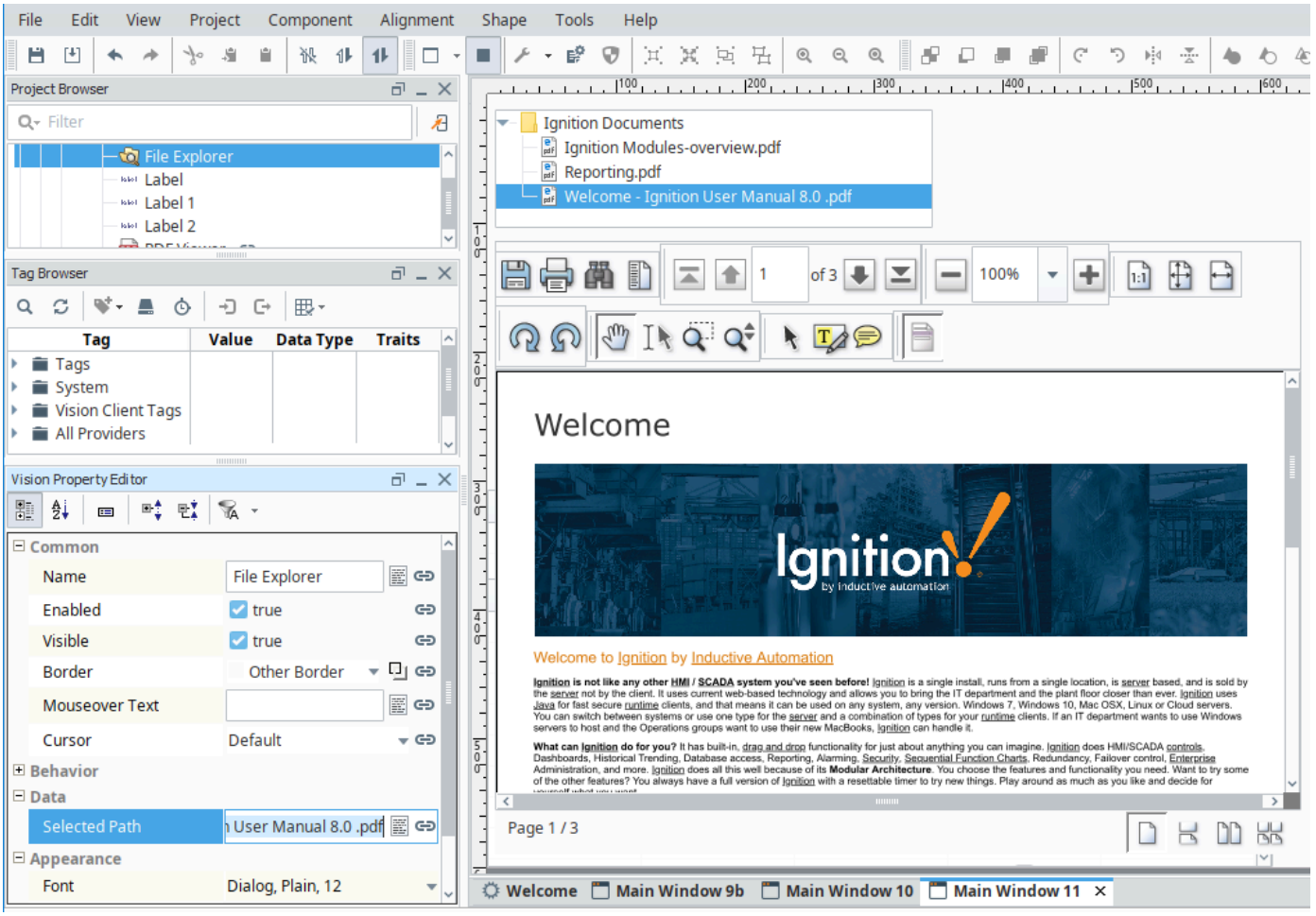
5. To filter for only PDF file types, enter '**pdf**' (without quotes) in the **File Extension Filter** property, otherwise, all file types will be displayed inside the Root Directory.



6. In **Preview Mode**, click on one of the reports in your Root Directory. The **PDF Viewer** works by passing the file path of the file you selected to the PDF Viewer. You'll also notice several PDF Viewer properties that drive the appearance of the PDF Viewer: Page Fit Mode, Hide or Show Toolbar, Utility Bar, Highlight, Select, Save, and Print.

NOTE

Clicking on the Save icon in the runtime saves a copy of the report to the **client** computer, not the Ignition Gateway.

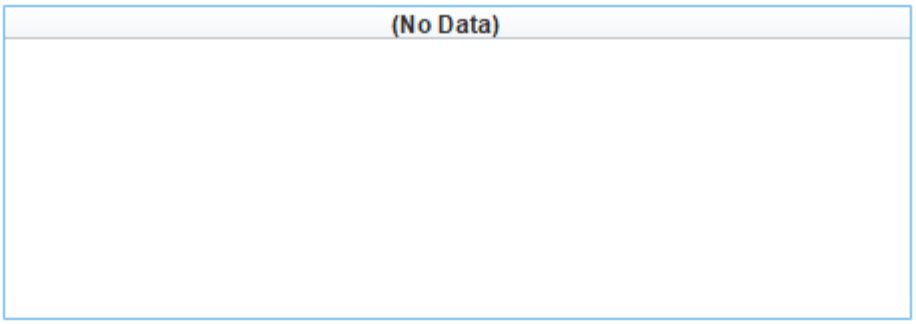


Vision - Tables Palette

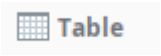
The following components give you various types of tables for displaying values.

- **Table**
- **Power Table**
- **List**
- **Tree View**
- **Comments Panel**
- **Tag Browse Tree**

Vision - Table



Component Palette Icon:



Description

The Table component allows you to display tabular data in a variety of ways. Important features include:

- **Column Sorting**: Allow users to sort the data by clicking on the column headers. Sorting has three modes: Ascending, Descending, and "Natural", which uses the default order of the data.
- **Mapped Row Coloring**: Map the background color of each row to a particular column.
- **Column Translation**: Allow the table component to handle all code mapping, such as mapping 0 to "Off" and 1 to "On".
- **Images**: Map values to images.
- **Progress Bar Indication**: Display numeric data as progress bars inside cells, providing fast visual reference for bounded amounts.
- **Number and Date formatting**: Format numbers and dates to your specification.
- **Column Hiding**: Hide columns from view.
- **Printing**: Print tables directly to multi-paged printouts.
- **Editing**: Columns can be made user-editable. Changes will be reflected in the underlying dataset, at which point they can be mapped back to a database.

Changing the Column Widths

To change a table's column's widths, simply switch into preview mode and use your mouse to resize the columns, and then switch back to design mode. To ensure that the changes to the column widths

appear in the client, right-click on the table to open the table customizer and click OK without clicking anywhere else in the customizer. Clicking anywhere else in the customizer before clicking OK will reset the table column widths.

Editable Table

By setting any column to editable in the Table's customizer, the user will be able to double-click in the cell and edit the data. You can the respond to the resulting cellEdited event with an event handler and persist the data.

Properties

Property	Description	Property Type	Scripting	Category
Auto-Resize Mode	Determines how the table resizes the columns.	int	.autoResizeMode	Behavior
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.	Color	.background	Appearance
Background Mode	This mode determines the color that this table's cell's backgrounds will be.	int	.backgroundColorMode	Appearance
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched	Border	.border	Common

Property	Description	Property Type	Scripting	Category
	(Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.			
Column Attributes Data	The dataset describing the column attributes.	Dataset	.columnAttributesData	Data
Column Selection Allowed	This flag is used in conjunction with the Row Selection Allowed property to determine whether whole-rows, whole-columns, or both (single-cells) are selectable.	boolean	.columnSelectionAllowed	Behavior
Cursor	The mouse cursor to use when hovering over this component. Options are: Default,	int	.cursorCode	Common

Property	Description	Property Type	Scripting	Category
	Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.			
Data	The data for this table.	Dataset	.data	Data
Edit Click Count	The number of clicks required to start editing a cell.	int	.clickCountToStart	Behavior
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See **Color Selector**.	Color	.foreground	Appearance
Grid Line Color	The color used to draw grid lines. See **Color Selector**.	Color	.gridColor	Appearance

Property	Description	Property Type	Scripting	Category
Header Font	Font of the table's header text.	Font	.headerFont	Appearance
Header Foreground Color	The foreground color of the table's header. See **Color Selector**.	Color	.headerForeground	Appearance
Header Visible	Whether or not the table header is visible.	boolean	.headerVisible	Appearance
Initially Selected Row	The index of the row that should be selected by default.	int	.initialRowSelection	Behavior
Mouseover Text	The text that is displayed in the tooltip that pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Odd Row Background	The color which odd rows will be colored if	Color	.oddBackground	Appearance

Property	Description	Property Type	Scripting	Category
	background mode is 'Alternating'. See **Color Selector**.			
Opaque	If false, backgrounds are not drawn. If true, backgrounds are drawn.	boolean	.opaque	Common
Properties Loading	The number of properties currently being loaded. (Read only. Usable in bindings and scripting.)	int	.propertiesLoading	Uncategorized
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Deprecated
Resizing Allowed	Whether or not the user is allowed to resize table headers or not.	boolean	.resizingAllowed	Behavior
Row Height	The height of each row, in pixels.	int	.rowHeight	Appearance

Property	Description	Property Type	Scripting	Category
Row Selection Allowed	This flag is used in conjunction with the Column Selection Allowed property to determine whether whole-rows, whole-columns, or both (single-cells) are selectable.	boolean	.rowSelectionAllowed	Behavior
Selected Column	The index of the first selected column, or -1 if none.	int	.selectedColumn	Data
Selected Row	The index of the first selected row, or -1 if none.	int	.selectedRow	Data
Selection Background	The background color of a selected cell. See **Color Selector**.	Color	.selectionBackground	Appearance
Selection Foreground	The foreground color of a selected cell.	Color	.selectionForeground	Appearance

Property	Description	Property Type	Scripting	Category
	See **Color Selector**.			
Selection Mode	This mode determines if only one row/cell/column can be selected at once, or single or multiple intervals.	int	.selectionMode	Behavior
Show Horizontal Grid Lines?	Shows horizontal grid lines.	boolean	.showHorizontalLines	Appearance
Show Vertical Grid Lines?	Shows vertical grid lines.	boolean	.showVerticalLines	Appearance
TestData	Toggle this property to fill in the table's data with random data.	boolean	.test	Misc
Touchscreen Mode	Controls when this table component responds if touchscreen mode is enabled.	int	.touchscreenMode	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Scripting

See the **Vision - Table Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

- **Table Customizer**
- **Component Customizers**

Examples

Binding to Selected Data

It is possible to bind other components to values in the selected row of the table. To do this, you will need to write an expression binding that protects against the case when nothing is selected or there are no rows. An expression like this would bind a label to the selected row's value in the "ProductCode" column:

Expression Binding

```
if({Root Container.MyTable.selectedRow} = -1,
    "n/a", // this is the fail case
    {Root Container.MyTable.data}[{Root
Container.MyTable.selectedRow},"ProductCode"] // this selects from the dataset
)
```

If you're binding to an integer, date, or other non-String value inside a dataset, you will need to cast the value to the correct type. This binding would cast the selected "Quantity" column to an integer:

Expression Binding

```
```

```
if({Root Container.MyTable.selectedRow} = -1,
    -1, // this is the fail case
    toInt({Root Container.MyTable.data}[{Root
Container.MyTable.selectedRow},"Quantity"]) // this selects from the dataset
)
```

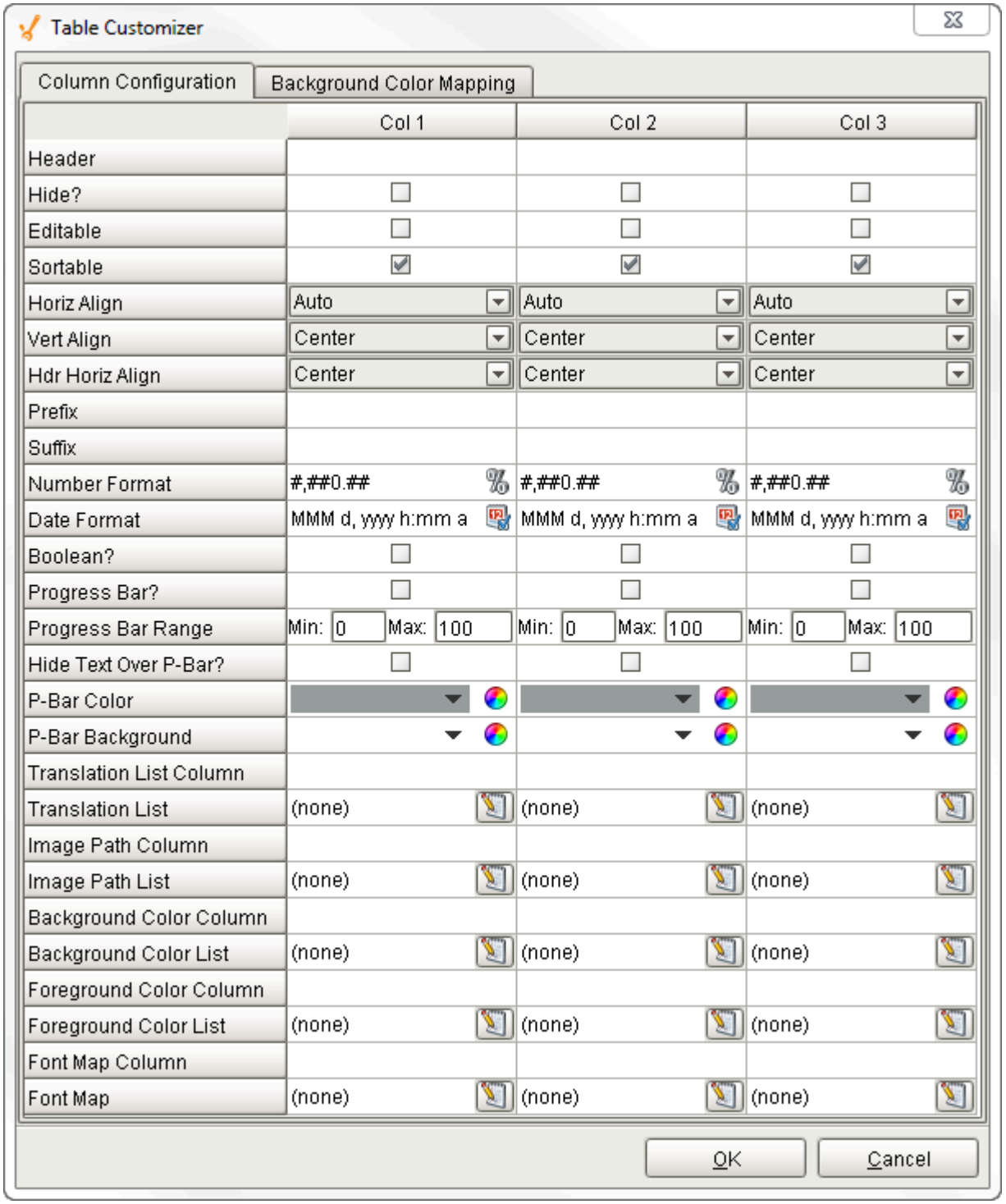
Adding a New Row

Code Snippet

```
#The following would add a row to the table.
#Note that this function takes a list
#And that the property types of the list are the same as the table.

name = "Motor 1"
state = 2
amps = 35
list = [name, state, amps]
table = event.source.parent.getComponent('Table')
table.addRow(list)
```

Vision - Table Customizer



Description

The **Table component** is one of the most flexible and easy to configure components in Ignition. It has its own Table Customizer that allows you to make changes to tabular data and display it in a variety ways. The customizer not only lets you customize each column in the table, but together with its data properties and use of scripting and extension functions, it lets you configure how each cell in the table looks and behaves.

Customizers

The Table Customizer allows you to configure how you want the table to look to users. When you open the Table Customizer, you'll notice two tabs: Column Configuration and Background Color Mapping. The Column Configuration tab contains a number of column configuration properties that can be used to customize each column in the dataset to look a certain way. You can assign a header name, hide a column, make the column editable and sortable, align the text within the column, add a prefix by putting a "$" in front of a value, or suffix by adding a "%" at the end of a value, select a number and date format, turn the column into a progress bar, translate a number into a string or image or even into a background or foreground color. It's even possible to change the background, foreground, and font for the text in each particular column or cell, and more.

In the Background Color Mapping tab, you can set the table's Background property to 'Mapped', and choose a column to govern the background color of each row. The column is specified in the Mapping Column dropdown selector. The column must be a numeric type. The number to color translation works with the contents of the mapping column rows to format the cells in accordance with the selected color.

> **TESTDATA PROPERTY**
>
> If you want to test how the Table Customizer works in the Table, drag a Table on to your workspace, go to the Test Data property in the Property Editor, and check the 'false' checkbox. It will automatically fill the table with some test data so you get test out the Table Customizer

- **Component Customizers**

Table Customizer Properties

Column Configuration Tab

Property	Description
Header	Provide a custom name to the column header.
Hide	Hides the column.
Editable	Allows the editing of the cell pertaining to the column.
Sortable	Allows the user to sort the table according to the selected column.
Horiz Align	Aligns the contents of the column.
Vert Align	Aligns the contents of the column.
Hdr Horiz Align	Aligns the contents of the column.
Prefix	A custom text that proceeds the contents of each cell.
Suffix	A custom text that follows the contents of each cell.
Number Format	A format of the cell if the contents of the cell are number types.
Boolean	Changes the contents of the cell to reflect a 'check box' look and feel.
Progress Bar	A graphical bar is represented in the cell instead of a number.
Progress Bar Range	Sets the min and max range of the progress bar.
Hide Text Over P-Bar	Makes the value and text that controls the progress bar visible or invisible.
P-Bar Color	The color of the progress bar.
P-Bar Background	The color of the cell that has a progress bar.

Property	Description
Translation List Column	This works in conjunction with the Translation List. The key is provided by a named column resulting in the cells being translated according to the list that contains the key pairs.
Translation List	Defines the key/Translation pairs and translates the contents of the cell accordingly.
Image Path Column	This works in conjunction with the Image Path List. The key is provided by a named column resulting in the cells being translated according to the list that contains the key pairs.
Image Path List	Defines the key/Translation pairs and translates the contents of the cell accordingly.
Background Color Column	This works in conjunction with the Background Color List. The key is provided by a named column resulting in the cells being translated according to the list that contains the key pairs.
Background Color List	Defines the key/Translation pairs and translates the contents of the cell accordingly.
Foreground Color Column	This works in conjunction with the Foreground Color List. The key is provided by a named column resulting in the cells being translated according to the list that contains the key pairs.
Foreground Color List	Defines the key/Translation pairs and translates the contents of the cell accordingly.
Font Map Column	This works in conjunction with the Foreground Color List. The key is provided by a named column resulting in the cells being translated according to the list that contains the key pairs.
Font Map	Defines the key/Translation pairs and translates the contents of the cell accordingly. An example of a font translation could look like this "Dialog, Bold, 12"

Color Mapping Tab

Property	Description
Mapping Column	Select a column to govern the background color of each row.
Number to Color Translation	A numeric value (typically an integer) that drives the background and foreground color of a row. For every number or value, you can choose a different color.
Fallback Color	Default color that can be set when a value does is not defined.

Examples

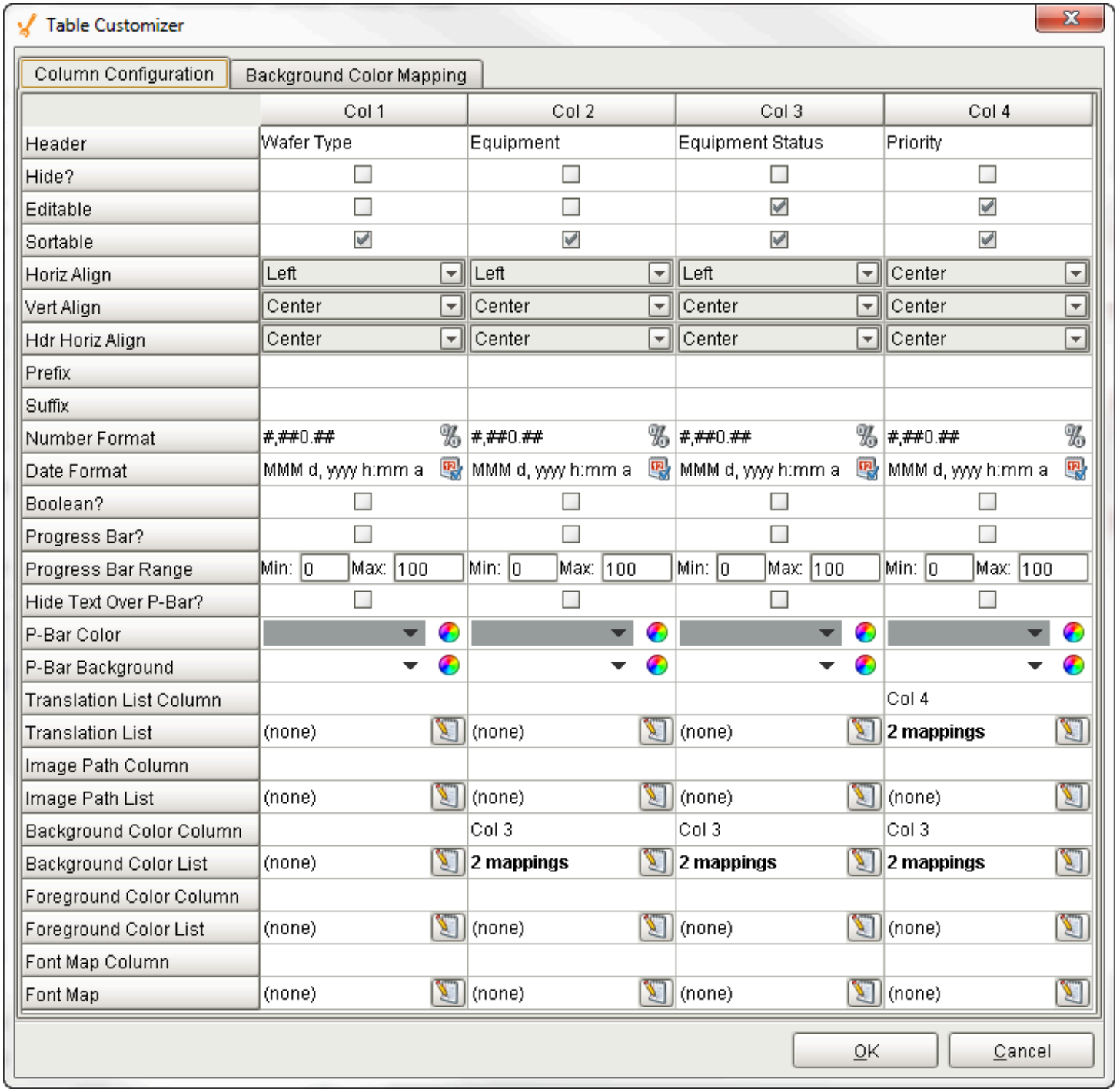
The table in this example uses several mappings:

- Col 4 changed a number into a string: translated a priority "1" to Critical, and priority "2" to High. It also change the background colors of the cells for both priorities.
- Col 3 changed the background colors for the equipment status's "Maintenance" and "Idle" to pale red.
- Col 2 change the background color of the equipment name to pale red for the equipment status's that were "Idle" and "Maintenance."

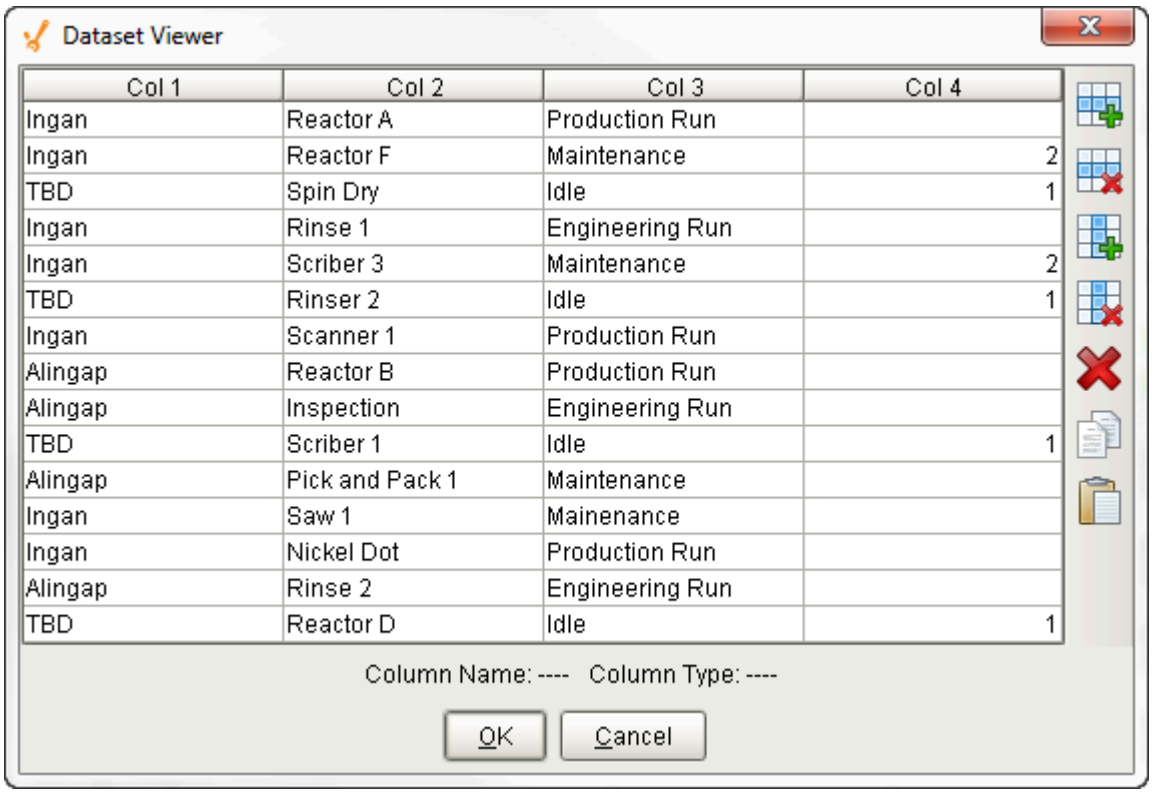
Table

Wafer Type	Equipment	Equipment Status	Priority
Ingan	Reactor A	Production Run	
Ingan	Reactor F	Maintenance	High
TBD	Spin Dry	Idle	Critical
Ingan	Rinse 1	Engineering Run	
Ingan	Scriber 3	Maintenance	High
TBD	Rinser 2	Idle	Critical
Ingan	Scanner 1	Production Run	
Alingap	Reactor B	Production Run	
Alingap	Inspection	Engineering Run	
TBD	Scriber 1	Idle	Critical
Alingap	Pick and Pack 1	Maintenance	
Ingan	Saw 1	Mainenance	
Ingan	Nickel Dot	Production Run	
Alingap	Rinse 2	Engineering Run	
TBD	Reactor D	Idle	Critical

Table Customizer



Data Property Dataset



Vision - Table Scripting Functions

This page details the various component and extension functions available for **Vision's Table component**.

Component Functions

.addRow(newRow)

- Description
  - Adds a new row to the end of the table's dataset
- Parameters
  - PySequence **newRow** - A sequence containing the values for the new row. The length of the sequence must match the number of columns in the table, and each value must be coercible into the correct datatype of the corresponding column.
- Return
  - None

.deleteRow(rowIndex)

- Description
  - Deletes a row from the table's dataset.
- Parameters
  - int **rowIndex** - The index of the row to delete.
- Return
  - None

.exportCSV(filename, showHeaders)

- Description Prompts the user to save the table's data as a CSV file.
- Parameters
  - String **filename** - A suggested filename for the user. For example: "table_data.csv"
  - boolean **showHeaders** - If true, include headers in CSV file.
- Return

    - String - The path to the saved file, or null if the operation was cancelled.

.getDataAsHTML(title, width)

- Description
    - Creates an HTML page as a string in memory. This can then be written to a file, a database, emailed, etc.
- Parameters
    - String **title** - The title for the HTML page.
    - int **width** - The width (in pixels) for the "table" element in the resulting html page.
- Return
    - String - A string containing an HTML-formatted version of the table's data.

.getRowsInViewOrder()

- Description
    - Returns a list of ints that represent the underlying dataset's rows as they appear in the current sort order that the user is viewing.
- Parameters
    - none
- Return
    - List of Integers

.getSelectedColumn()

- Description
    - Returns the index of the currently selected column, or -1 if none is selected.
- Parameters
    - none
- Return
    - int

.getSelectedColumnCount()

- Description
    - Returns the number of columns that are currently selected.
- Parameters
    - none
- Return

- int

.getSelectedRow()

- Description
  - Returns the index of the currently selected row, or -1 if none is selected.
- Parameters
  - none
- Return
  - int

.getSelectedRows()

- Description
  - Returns a list of the indexes of the selected row, or none if none is selected.
- Parameters
  - none
- Return
  - List, None

.getSelectedRowCount()

- Description
  - Returns the number of rows that are currently selected.
- Parameters
  - none
- Return
  - int

.isCellSelected(row, column)

- Description
  - Tests whether the cell at the given row and column is currently selected or not.
- Parameters
  - int **row** - The row to test.
  - int **column** - The column to test.
- Return
  - boolean

.isColumnSelected(column)

- Description
  - Tests whether the given column is currently selected or not.
- Parameters
  - int **column**- The column to test.
- Return
  - boolean

.isRowSelected(row)

- Description
  - Tests whether the given row is currently selected or not.
- Parameters
  - int **row** - The row to test.
- Return
  - boolean

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

- Description
  - This specialized print function will paginate the table onto multiple pages.This function accepts keyword-style invocation.
- Keyword Args
  - boolean **fitWidth**- If true, the table's width will be stretched to fit across one page's width. Rows will still paginate normally. If false, the table will paginate columns onto extra pages. (default = true) [optional]
  - string **headerFormat**- A string to use as the table's page header. The substring "0" will be replaced with the current page number. (default = None) [optional]
  - string **footerFormat**- A string to use as the table's page footer. The substring "0" will be replaced with the current page number. (default = "Page 0") [optional]
  - boolean **showDialog**- Whether or not the print dialog should be shown to the user. Default is true. [optional]
  - boolean **landscape**- Used to specify portrait (0) or landscape (1) mode. Default is portrait (0). [optional]
- Return
  - boolean- True if the print job was successful.

.setColumnLabel(column, label)

- Description
  - Used to set a column's header label to a new string at runtime.
- Parameters
  - int **column** - The column index that will get a new headel label.
  - String **label** - The new header label.
- Return
  - None

.setColumnSelectionInterval(index0, index1)

- Description
  - Sets the given range of columns to be selected. If index0==index1, it will select a single column.
- Parameters
  - int **index0** - the first index.
  - int **index1** - the second index.
- Return
  - boolean - True if selection range is valid.

.setColumnWidth(column, width)

- Description
  - Used to set a column's width at runtime.
- Parameters
  - int **column** - The index of the column.
  - int **width** - The width to set it at in pixels.
- Return
  - None

.setRowSelectionInterval(index0, index1)

- Description
  - Sets the given range of rows to be selected. If index0==index1, it will select a single row.
- Parameters
  - int **index0** - The first index.
  - int **index1** - The second index.
- Return

- boolean - True if selection range is valid.

.setSelectedColumn(column)

- Description
  - Sets the given column to be the selected column.
- Parameters
  - int **column** - Column to select.
- Return
  - None

.setSelectedRow(row)

- Description
  - Sets the given row to be the selected row.
- Parameters
  - int **row** - Row to select.
- Return
  - None

.setValue(row, column, value)

- Description
  - Sets the value in the specified cell, altering the table's Data property. Will fire a propertyChange event for the "data" property, as well as a cellEdited event.
- Parameters
  - int **row** - The index of the row to set the value at.
  - int **column** - The index or name of the column to set a value at.
  - PyObject **value** - The new value to use at the given row/column location.
- Return
  - None

.sortByColumn(columnName [, asc])

- Description
  - Instructs the table to sort the data by the named column.
- Parameters
  - String **columnName** - The name of the column.
  - boolean **asc** - 1 means ascending, 0 means descending. (default = 1) [optional]

- Return
  - None

.sortOriginal()

- Description
  - Instructs the table to clear any custom sort columns and display the data as it is sorted in the underlying dataset.
- Parameters
  - None
- Return
  - None

.updateRow(rowIndex, changes)

- Description
  - Updates an entire row of the table's dataset.
- Parameters
  - int **rowIndex** - The index of the row to update.
  - PyDictionary **changes** - A sequence containing the updated values for the row. The length of the sequence must match the number of columns in the table, and each value must be coercible into the correct datatype of the corresponding column.
- Return
  - None

Extension Functions

getBackgroundAt

- Description
  - Called for each cell, returns the appropriate background color. Do not block, sleep, or execute any I/O; called on painting thread.
- Keyword Args
  - Component **self** - A reference to the component that is invoking this function.
  - int **row** -The row index of the cell.
  - int **col** -The column index of the cell.
  - boolean **isSelected** - A boolean representing if the cell is currently selected.
  - Object **value** -The value in the table's dataset at index [row, col].

    - Color **defaultColor** -The color the table would have chosen if this function was not implemented.
- Return
    - Color

getForegroundAt

- Description
    - Called for each cell, returns the appropriate foreground (text) color. Do not block, sleep, or execute any I/O; called on painting thread.
- Keyword Args
    - Component **self** - A reference to the component that is invoking this function.
    - int **row** -The row index of the cell.
    - int **col** -The column index of the cell.
    - boolean **isSelected** - A boolean representing if the cell is currently selected.
    - Object **value** -The value in the table's dataset at index [row, col].
    - Color **defaultColor** -The color the table would have chosen if this function was not implemented.
- Return
    - Color

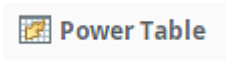
getDisplayTextAt

- Description
    - Called for each cell, returns a String which will be used as the text of the cell. Do not block, sleep or execute any I/O; called on the painting thread.
- Keyword Args
    - Component **self** - A reference to the component that is invoking this function.
    - int **row** -The row index of the cell.
    - int **col** -The column index of the cell.
    - boolean **isSelected** - A boolean representing if the cell is currently selected.
    - Object **value** -The value in the table's dataset at index [row, col].
    - String **defaultText** -The string the table would have chosen if this function was not implemented.
- Return
    - String

Vision - Power Table

Int Column	String Column	Float Column	Boolean Column	Date Column
0	EC44CC70	0.74	☐	Oct 5, 2015 12:48 ...
0	41F485FF	0.8	☐	Oct 5, 2015 12:48 ...
1	D0E52A70	0.52	☑	Oct 5, 2015 12:48 ...
1	95D58D81	0.36	☐	Oct 5, 2015 12:48 ...
1	E2830442	0.93	☑	Oct 5, 2015 12:48 ...
1	5EB774CF	0.87	☐	Oct 5, 2015 12:48 ...
2	2587FD4A	0.57	☐	Oct 5, 2015 12:48 ...
3	73A861C2	0.85	☑	Oct 5, 2015 12:48 ...
3	F45F3671	0.34	☐	Oct 5, 2015 12:48 ...
3	8C2E75C8	0.03	☑	Oct 5, 2015 12:48 ...
5	C16ADCF4	0.7	☐	Oct 5, 2015 12:48 ...
6	0ED022F5	0.77	☑	Oct 5, 2015 12:48 ...
6	A7B59529	0.98	☑	Oct 5, 2015 12:48 ...
6	DEEEDE4F	0.57	☑	Oct 5, 2015 12:48 ...
7	D1D896F6	1	☐	Oct 5, 2015 12:48 ...
8	3BE188EB	0.67	☑	Oct 5, 2015 12:48 ...
8	F6A828D5	0.11	☑	Oct 5, 2015 12:48 ...
9	1C11763F	0.87	☑	Oct 5, 2015 12:48 ...
9	24CC77B9	0.36	☐	Oct 5, 2015 12:48 ...
9	2827173B	0.44	☑	Oct 5, 2015 12:48 ...
9	DB2A2EC5	1	☐	Oct 5, 2015 12:48 ...
10	DB1DE68F	0.86	☑	Oct 5, 2015 12:48 ...
11	2F25FDEF	0.69	☐	Oct 5, 2015 12:48 ...
11	E498FA42	0.65	☑	Oct 5, 2015 12:48 ...

Component Palette Icon:



Description

The power table is a more customizable version of the table component. The power table contains advanced features such as drag-and-drop rows, multi-column sorting, column filtering, and cell-spanning.

Power Table Features

- **Multi-column sorting**: To sort multiple columns, select the header of the first column, hold down the Control key, then select the header of the next column. Click on the header again to reverse the sort order, and click a third time to remove sorting on the column.

- **Column filtering**: Columns can be temporarily hidden from view using column filtering. Right-click on the header of the table, and uncheck columns that you would like to hide. You can disable this feature by disabling the Column Chooser Menu property on the table.
- **Column reordering**: You can switch the locations of columns on the table using column reordering. Drag the header of the column that you would like to move to a new location on the table. You can disable this feature by disabling the Columns Re-Orderable property on the table.
- **Cell spanning**: A cell can be spanned across multiple columns and rows. Keep in mind that you must explicitly define the locations of cells that must be spanned. This means that if you would like to use cell spanning, any other table features that change how the table is displayed will be disabled automatically (such as sorting, column filtering and column reordering). Click on the Cell Span Data dataset to configure spanning. Within the dataset, add a row for each new span. The "row" column controls the row in the table where the span will start. The "column" column controls the column where the span will start. The "width" column controls how many columns the span will cover. The "height" column controls how many rows the span will cover. Adding a row where "row=4, column=1, width=2, height=3" results in a span starting on the fifth row of the table and the second column (using 0-based indexing). The span will cover the second and third columns in the row and will also cover two rows below the fifth row, as shown below.
- **Drag and Drop**: This feature allows you to drag rows from one power table to another power table. In order to perform drag and drop, you must implement the onRowsDropped() extension function on the destination table. This is so that you can adapt the data from one table to the other within the function. You must also enable the Row Dragging Enabled property on both tables.
- **Row Copying**: This feature allows you to select rows and copy them to the clipboard using the standard keyboard shortcut Ctrl + C. These can then be pasted anywhere, even outside of Ignition.

> **CAUTION**
>
> Even if a column is set to be editable, the edit must be handled by the onCellEdited extension function. If that extension function is not enabled and properly set up, the cell will revert back to its previous value.

Properties

Property	Description	Property Type	Scripting	Categ
Auto Row Height	Enables automatic resizing of row height.	boolean	.rowResizeEnabled	Behav
Auto-Resize Mode	Determines how the table resizes the columns.	int	.autoResizeMode	Behav
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	Appea
Cell Span Data	This dataset holds information about how cells in the table span multiple rows and/or columns. Incompatible with column sorting and re-ordering.	Dataset	.cellSpanData	Data
Column Attributes Data	The dataset describing the column attributes.	Dataset	.columnAttributesData	Appea

Property	Description	Property Type	Scripting	Categ
	Note: The data in this property doesn't get initialized until the customizer is opened and the OK button is pressed.			
Column Chooser Menu	Enables a right-click popup menu on the column headers with options to show and hide columns. For full menu options, see **Column Chooser Menu Options**.	boolean	.headerColumnChooserMenus	Behav
Column Resize Menu	Enables a right-click popup menu on the column headers with resizing options.	boolean	.headerResizeMenus	Behav
Column Selection Allowed	This flag is used in conjunction with the Row Selection Allowed flag to determine whether not whole-rows, whole-columns, or both (single-	boolean	.columnSelectionAllowed	Behav

Property	Description	Property Type	Scripting	Categ
	cells) are selectable.			
Column Sizing	Represents column sizing and position to preserve user-selected ordering.	String	.defaultColumnView	Appea
Columns Re-Orderable	Enables the re-ordering of columns by dragging the column headers.	boolean	.columnReorderingAllowed	Behav
Columns Resizable	Enables the resizing of columns by dragging the margins of the column headers.	boolean	.columnResizingAllowed	Behav
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	Comm

Property	Description	Property Type	Scripting	Categ
Data	The data for this table.	Dataset	.data	Data
Edit Click Count	The number of clicks required to start editing a cell.	int	.clickCountToStart	Behav
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Comm
Font	Font of text on this component.	Font	.font	Appea
Foreground Color	The foreground color of the component. See **Color Selector**.	Color	.foreground	Appea
Grid Line Color	The color used to draw grid lines. See **Color Selector**.	Color	.gridColor	Appea
Header Font	Font of the table's header text.	Font	.headerFont	Appea
Header Visible	Allows for hiding of the table's header.	boolean	.headerVisible	Appea
Inter Cell Spacing	The space (in pixels) between the cells.	Dimension	.interCellSpacing	Appea

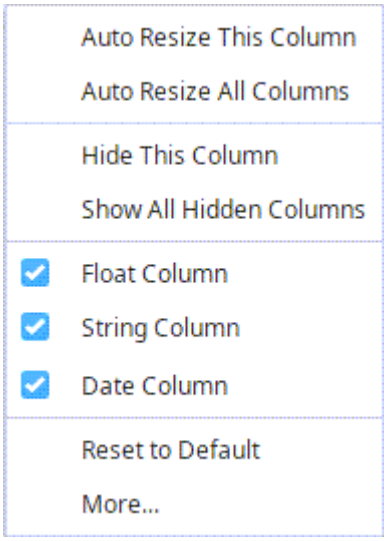
Property	Description	Property Type	Scripting	Categ
Name	The name of this component.	String	.name	Comm
Non-Contiguous Selection	Enables totally non-contiguous selection in the table.	boolean	.nonContiguousCellSelection	Behav
Properties Loading	The number of properties currently being loaded. (Read only. Usable in bindings and scripting.)	int	.propertiesLoading	Uncat
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Row Dragging Enabled	Enables drag-and-drop re-ordering for table rows. Implementing the 'onRowsDropped' extension function is also required to have functional drag-and-drop.	boolean	.rowDragEnabled	Behav
Row Height	If row resizing is disabled, this will set the height of all rows.	int	.rowHeight	Behav

Property	Description	Property Type	Scripting	Categ
Row Selection Allowed	This flag is used in conjunction with the Column Selection Allowed flag to determine whether not whole-rows, whole-columns, or both (single-cells) are selectable.	boolean	.rowSelectionAllowed	Behav
Selected Column	The index of the first selected column, or -1 if none.	int	.selectedColumn	Data
Selected Row	The index of the first selected row, or -1 if none.	int	.selectedRow	Data
Selection Background	The default background color of selected cells. See **Color Selector**.	Color	.selectionBackground	Appea
Selection Foreground	The default foreground color of selected cells. See **Color Selector**.	Color	.selectionForeground	Appea
Selection Mode	This mode determines if only one	int	.selectionMode	Behav

Property	Description	Property Type	Scripting	Categ
	row/cell/column can be selected at once, or single or multiple intervals.			
Show Horizontal Grid Lines?	Shows horizontal grid lines.	boolean	.showHorizontalLines	Appea
Show Vertical Grid Lines?	Shows vertical grid lines.	boolean	.showVerticalLines	Appea
Sorting Enabled	Enables automatic multi-column sorting by clicking and CTRL-clicking on the table header.	boolean	.sortingEnabled	Behav
TestData	Toggle this property to fill in the table's data with random data.	boolean	.test	Misc
View Dataset	A read-only copy of the data as it appears on screen in the table. The purpose of this property is to preserve the column ordering, column visibility, and applied sorting order. Other attributes,	Dataset	.viewDataset	Data

Property	Description	Property Type	Scripting	Categ
	such as formatting, will not be preserved in this dataset.			
Visible	If disabled, the component will be hidden.	boolean	.visible	Comm

Column Chooser Menu Options



Property	Description
Auto Resize This Column	Resizes the selected column as determined by the Auto-Resize Mode property.
Auto Resize All Columns	Resizes all columns as determined by the Auto-Resize Mode property.
Hide This Column	Hides the selected column.
Show All Hidden Columns	Shows all columns that were hidden in the current Client Session. Option is only visible after a column has been hidden.
Available Column List	Lists all available table columns with the option to show or hide each column.
Reset to Default	Resets the Power Table to the Designer configuration, overriding any Client Session changes.
More...	Opens a popup window with the complete column list to select the columns to display.

Scripting

See the **Power Table Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

The Power Table Customizer can be opened by double-clicking on the Power Table in the Designer.

- **Power Table Customizer**
- **Component Customizers**

Example

Code Snippet

```
# Example of an onRowsDropped() extension script for two power tables with
identical columns:

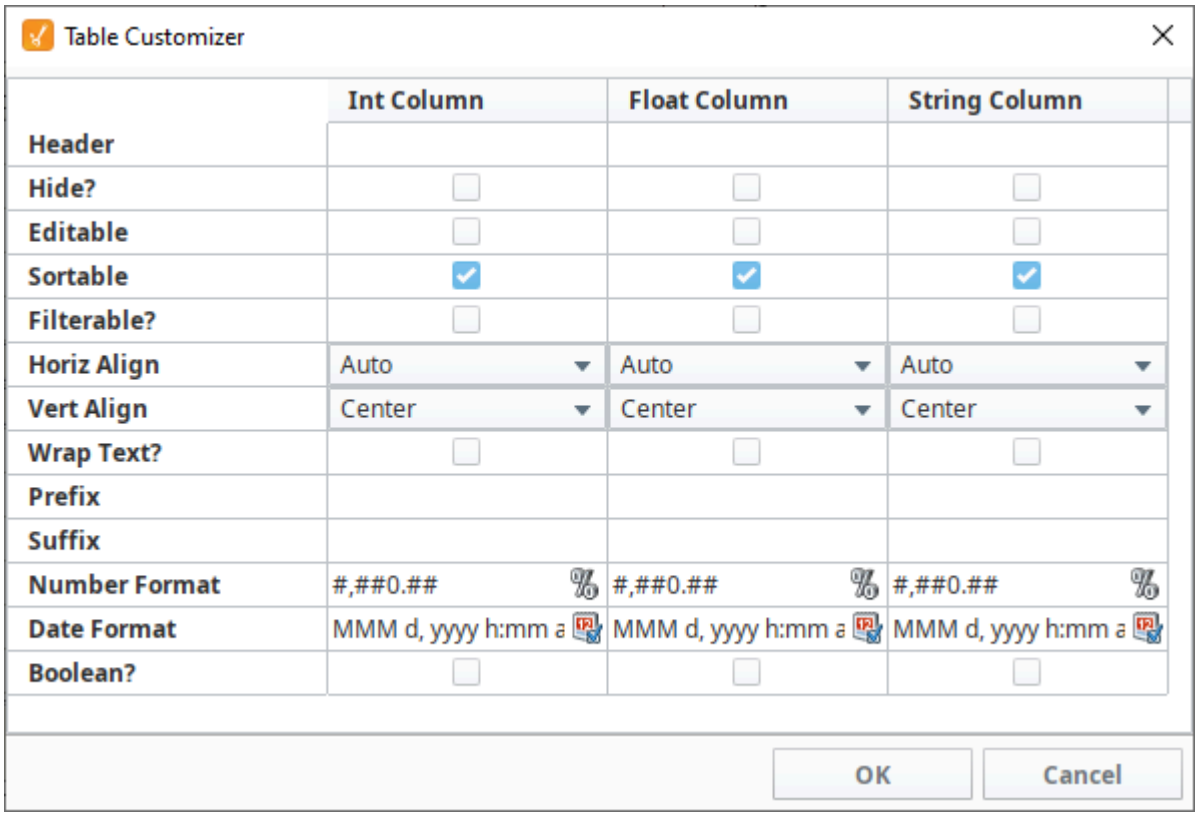
def onRowsDropped(self, sourceTable, rows, rowData, dropIndexLocation):
    if self != sourceTable:
        destDataset = self.getData()

        insertIndex = dropIndexLocation
        for row in rowData:
            newRow = list(row)  # convert row into a list
            destDataset = system.dataset.addRow(destDataset, insertIndex, newRow)
            insertIndex += 1

        # Adds the rows to the destination table.
        self.setData(destDataset)

        # Optional. Deletes the dragged rows from the source table.
        sourceDataset = system.dataset.deleteRows(sourceTable.getData(), rows)
        sourceTable.setData(sourceDataset)
    else:
        system.vision.showMessage("Dropping on to same table not supported")
        # To drop onto the same table, the new row indices must be calculated
        # for both the dropped and deleted rows, taking changes into account.
```

Vision - Power Table Customizer



Description

The **Vision - Power Table** offers the same functionality as the classic **Vision - Table** component, but has more features. Just like the classic Table, it not only provides a Table Customizer that allows you to make changes to the table columns, but coupled with its data properties and use of extension functions, it lets you configure how each cell in the table looks and behaves.

Customizers

The Table Customizer allows you to configure how you want the table to look to users. When you open the customizer, you'll notice that the data is formatted into different columns. The left column contains all the Table Customizer properties. For each column in the customizer, you can assign a header name, hide the column, make it editable and sortable, change the horizontal and vertical alignment of text, select a number format and date format style, and more.

> **TESTDATA PROPERTY**

If you want to test how the Table Customizer works in the Power Table, drag a Power Table on to your workspace, go to the Test Data property in the Property Editor, and check the 'false' checkbox. It will automatically fill the table with some test data so you get test out the Table Customizer.

- **Power Table**
- **Component Customizers**

Table Customizer Properties

Property	Description
Header	Provide a custom name to the column header.
Hide	Hides the column.
Editable	Allows the editing of the cell pertaining to the column. While the cell will be editable, the edit won't do anything and the cell will revert back to its previous value unless the edit is handled by the onCellEdited extension function.
Sortable	Allows the user to sort the table according to the selected column.
Filterable	Allows the user to filter the table according to the selected column.
Horiz Align	Aligns the contents of the column: Auto, Left, Center, Right.
Vert Align	Aligns the contents of the column: Top, Center, Bottom.
Wrap Text	The text will wrap if its contents are longer than the width of the cell.
Prefix	A custom text that proceeds the contents of each cell.
Suffix	A custom text that follows the contents of each cell.
Number Format	A format of the cell if the contents of the cell are number types.
Date Format	A format of the cell if the contents of the cell are date types.
Boolean	Changes the contents of the cell to reflect a 'check box' look and feel.

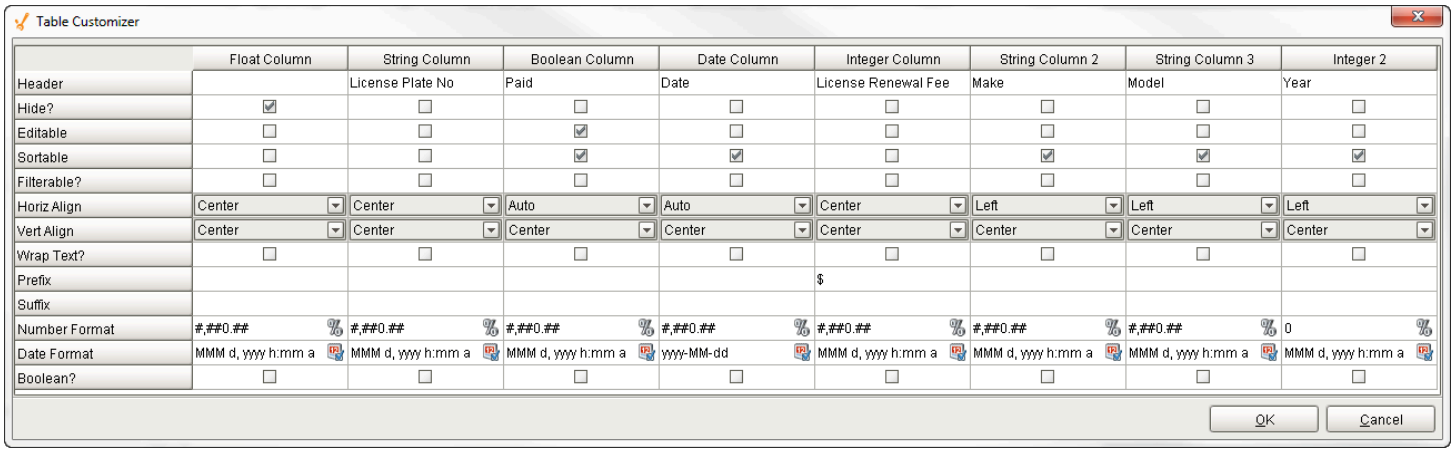
Power Table Customizer Example

In this example, compare the columns in the dataset and the table customizer to see how the individual columns were customized to create the chart below.

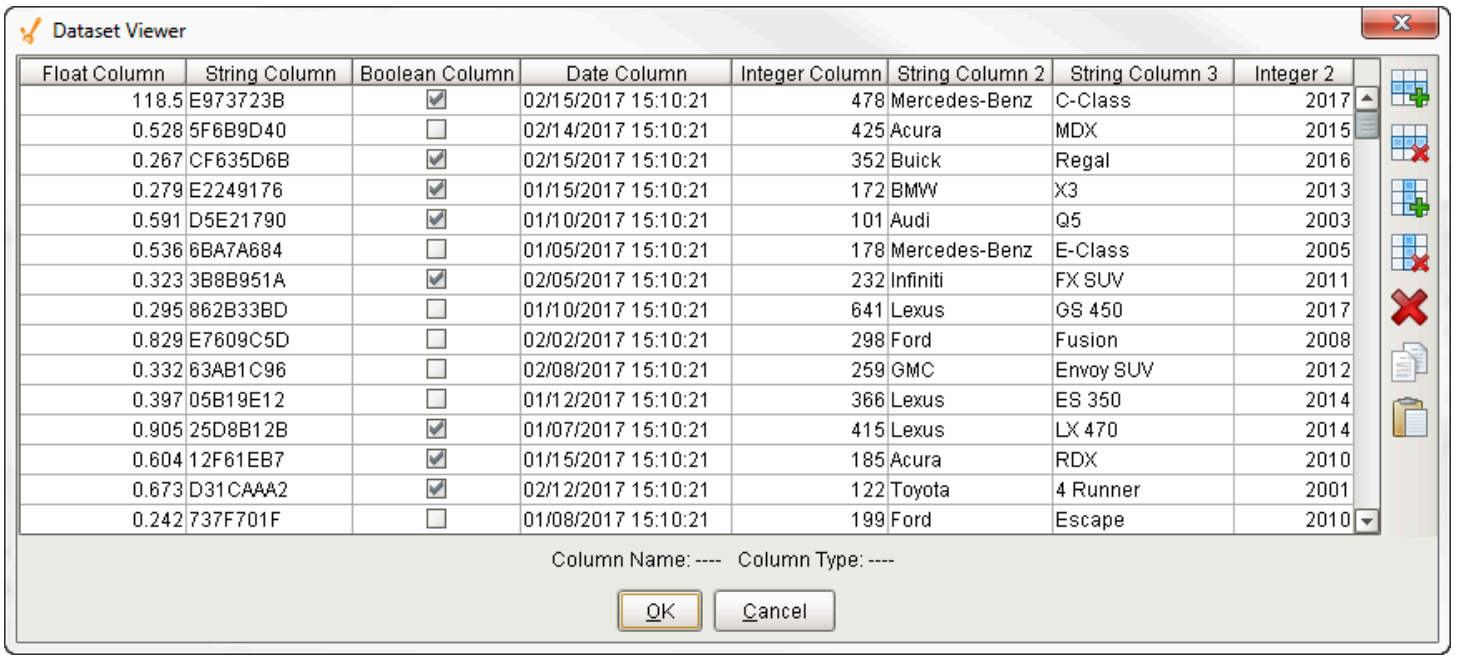
Power Table

Date / Time	Paid	License Renewal Fee	License Plate No	Make	Model	Year
2017-02-15	☑	$ 478	E973723B	Mercedes-Benz	C-Class	2017
2017-02-14	☐	$ 425	5F6B9D40	Acura	MDX	2015
2017-02-15	☑	$ 352	CF635D6B	Buick	Regal	2016
2017-01-15	☑	$ 172	E2249176	BMW	X3	2013
2017-01-10	☑	$ 101	D5E21790	Audi	Q5	2003
2017-01-05	☐	$ 178	6BA7A684	Mercedes-Benz	E-Class	2005
2017-02-05	☑	$ 232	3B8B951A	Infiniti	FX SUV	2011
2017-01-10	☐	$ 641	862B33BD	Lexus	GS 450	2017
2017-02-02	☐	$ 298	E7609C5D	Ford	Fusion	2008
2017-02-08	☐	$ 259	63AB1C96	GMC	Envoy SUV	2012
2017-01-12	☐	$ 366	05B19E12	Lexus	ES 350	2014
2017-01-07	☑	$ 415	25D8B12B	Lexus	LX 470	2014
2017-01-15	☑	$ 185	12F61EB7	Acura	RDX	2010
2017-02-12	☑	$ 122	D31CAAA2	Toyota	4 Runner	2001
2017-01-08	☐	$ 199	737F701F	Ford	Escape	2010

Table Customizer



Data Property Dataset



Vision - Power Table Scripting Functions

This page details the various component and extension functions available for **Vision's Power Table component**.

Component Functions

.getSelectedColumns()

- Description
  - Returns a list of ints representing the currently selected columns.
- Parameters
  - none
- Return
  - Object of Integers - An object containing integers that represent the indices of the selected columns. Can be iterated over in a similar manner to a Python List.

.getSelectedRows()

- Description
  - Returns a list of ints representing the currently selected rows.
- Parameters
  - none
- Return
  - Object of Integers - An object containing integers that represent the indices of the selected rows. Can be iterated over in a similar manner to a Python List.

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

- Description
  - This specialized print function will paginate the table onto multiple pages.This function accepts keyword-style invocation.
- Keyword Args
  - boolean **fitWidth** - If true, the table's width will be stretched to fit across one page's width. Rows will still paginate normally. If false, the table will paginate columns onto extra pages. (default = true) [optional]
  - String **headerFormat** - A string to use as the table's page header. The substring "0" will be replaced with the current page number. (default = None) [optional]
  - String **footerFormat** - A string to use as the table's page footer. The substring "0" will be replaced with the current page number. (default = "Page 0") [optional]
  - boolean **showDialog** - Used to determine if the print dialog should be shown to the user. Default is true. [optional]
  - boolean **landscape** - Used to specify portrait (0) or landscape (1) mode. Default is portrait (0). [optional]
- Return
  - boolean - True if the print job was successful.

.setColumnWidth(column, width)

- Description
  - Used to set a column's width at runtime.
- Parameters
  - int **column** - Column to adjust.
  - int **width** - Width in pixels.
- Return

- None

Extension Functions

configureCell

- Description
  - Provides a chance to configure the contents of each cell. Returns a dictionary of name-value pairs with the desired attributes. Available attributes (and their Java types) include: 'background' (color), 'border' (border), 'font' (font), 'foreground' (color), 'horizontalAlignment' (int), 'iconPath' (string), 'text' (string), 'toolTipText' (string), 'verticalAlignment' (int).

    You can also specify the attribute 'renderer', which is expected to be a javax.swing.JComponent which will be used to render the cell.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - Object **value** - The value in the dataset at this cell.
  - string **textValue** - The text the table expects to display at this cell (may be overriden by including 'text' attribute returned in dictionary).
  - boolean **selected** - A boolean indicating whether this cell is currently selected.
  - int **rowIndex** - The index of the row in the underlying dataset.
  - int **colIndex** - The index of the column in the underlying dataset.
  - string **colName** - The name of the column in the underlying dataset.
  - int **rowView** - The index of the row, as it appears in the table view (affected by sorting).
  - int **colView** - The index of the column, as it appears in the table view (affected by column re-arranging and hiding).
- Return
  - Dictionary of Attributes

configureEditor

- Description
  - Provides a chance to configure how each column is edited. Returns a dictionary of name-value pairs with desired editor attributes. Visual attributes to modify existing editors include: 'background', 'border', 'font', 'foreground', 'horizontalAlignment', 'toolTipText', and 'verticalAlignment'.

    If the attribute 'options' is specified, it is expected to be a list of tuples representing (value, label). The editor in this case will become a dropdown list.

    If the attribute 'editor' is specified, it is expected to be an instance of javax.swing.table.TableCellEditor, and other attribute will be ignored.

  The 'options' editor on the Power Table's configureEditor Extension Function accepts a rowHeight key allowing you to change the height of items in the dropdown. For example:

```
return {'options': [(0, 'Option A'), (1, 'Option B')], 'rowHeight':100}
```

- Parameters
  - Component **self** - A reference to the component that is invoking this function
  - int **colIndex** - The index of the column in the underlying dataset
  - string **colName** - The name of the column in the underlying dataset
- Return
  - Dictionary of name value pairs

configureHeaderStyle

- Description
  - Provides a chance to configure the style of each column header. Return a dictionary of name-value pairs with the designed attributes. Availible attributes include: 'background', 'border', 'font', 'foreground', 'horizontalAlignment', 'toolTipText', 'verticalAlignment'
- Parameters
  - Component **self** - A reference to the component that is invoking this function

    - int **colIndex** - The index of the column in the underlying dataset
    - string **colName** - The name of the column in the underlying dataset
- Return
    - Dictionary of name value pairs

initialize

- Description
    - Called when the window containing this table is opened, or the template containing it is loaded. Provides a chance to initialize the table further, for example, selecting a specific row.
- Parameters
    - Component **self** - A reference to the component that is invoking this function
- Return
    - None

isCellEditable

- Description
    - Returns a boolean that determines whether or not the current cell is editable.
- Parameters
    - Component **self** - A reference to the component that is invoking this function.
    - int **rowIndex** - Index of the row that was edited, relative to the underlying dataset.
    - int **colIndex** - Index of the column that was edited, relative to the underlying dataset.
    - string **colName** - Name of the column in the underlying dataset.
    - Object **value** - The value at the cell location.
- Return
    - boolean

onCellEdited

- Description
  - Called when the user has edited a cell in the table. It is up to the implementation of this function to alter the underlying data that drives the table. This might mean altering the dataset directly, or running a SQL UPDATE query to update data in the database.

> **NOTE**
>
> If the script on this extension function causes the Power Table to lose focus, the cell commit will occur twice. For example, if system.vision.showConfirm() is called, then two confirmation boxes will appear. In cases where the script will cause the focus to switch between multiple objects, the script should be placed in a function, and wrapped in a call to system.vision.invokeLater()

```
def myFunction():
    """
    Do your work here
    """
    system.vision.showMessage("Assuming you don't change focus outside of this script\nYou will only see this message once per cell edit")
system.vision.invokeLater(myFunction)
```

- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - int **rowIndex** - Index of the row that was edited, relative to the underlying dataset.
  - int **colIndex** - Index of the column that was edited, relative to the underlying dataset.
  - string **colName** - Name of the column in the underlying dataset.
  - Object **oldValue** - The old value at the location, before it was edited.
  - Object **newValue** - The new value input by the user.
- Return
  - None

onMousePress

- Description
  - Called when the user clicks on a table cell.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - int **rowIndex** - Index of the row, starting at 0, relative to the underlying dataset.
  - int **colIndex** - Index of the column starting at 0, relative to the underlying dataset.
  - Object **value** - The value at the location clicked on.
  - MouseEvent **event** - The MouseEvent object that caused this click event.
- Return
  - None

onMouseRelease

- Description
  - Called when the user releases the mouse button on a table cell.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - int **rowIndex** - Index of the row, starting at 0, relative to the underlying dataset.
  - int **colIndex** - Index of the column starting at 0, relative to the underlying dataset.
  - Object **value** - The value at the location that the mouse is released on.
  - MouseEvent **event** - The MouseEvent object that caused this released event.
- Return
  - None

onMouseClick

- Description
  - Called when the user clicks on a table cell.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - int **rowIndex** - Index of the row, starting at 0, relative to the underlying dataset.
  - int **colIndex** - Index of the column starting at 0, relative to the underlying dataset.
  - Object **value** - The value at the location clicked on.
  - MouseEvent **event** - The MouseEvent object that caused this click event.
- Return
  - None

onDoubleClick

- Description
  - Called when the user double-clicks on a table cell.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - int **rowIndex** - Index of the row, starting at 0, relative to the underlying dataset.
  - int **colIndex** - Index of the column starting at 0, relative to the underlying dataset.
  - Object **value** - The value at the location clicked on.
  - MouseEvent **event** - The MouseEvent object that caused this double-click event.
- Return
  - None

onPopupTrigger

- Description
  - Called when the user right-clicks on a table cell. This would be the appropriate time to create and display a popup menu.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - int **rowIndex** - Index of the row, starting at 0, relative to the underlying dataset.
  - int **colIndex** - Index of the column starting at 0, relative to the underlying dataset.
  - string **colName** - Name of the column in the underlying dataset.
  - Object **value** - The value at the location clicked on.
  - MouseEvent **event** - The MouseEvent object that caused this popup trigger event
- Return
  - None

onRowsDropped

- Description
  - Called when the user has dropped rows on this table. Note that the rows may have come from this table or another table. The source table must have dragging enabled.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - Component **sourceTable** - A reference to the table that the rows were dragged and dropped in the same table.
  - list **rows** - An array of the rows indices that were dragged, in the order they were selected.
  - Dataset **rowData** - A dataset containing the rows that were dragged.
  - int **dropIndexLocation** - Row index where the rows were dropped.

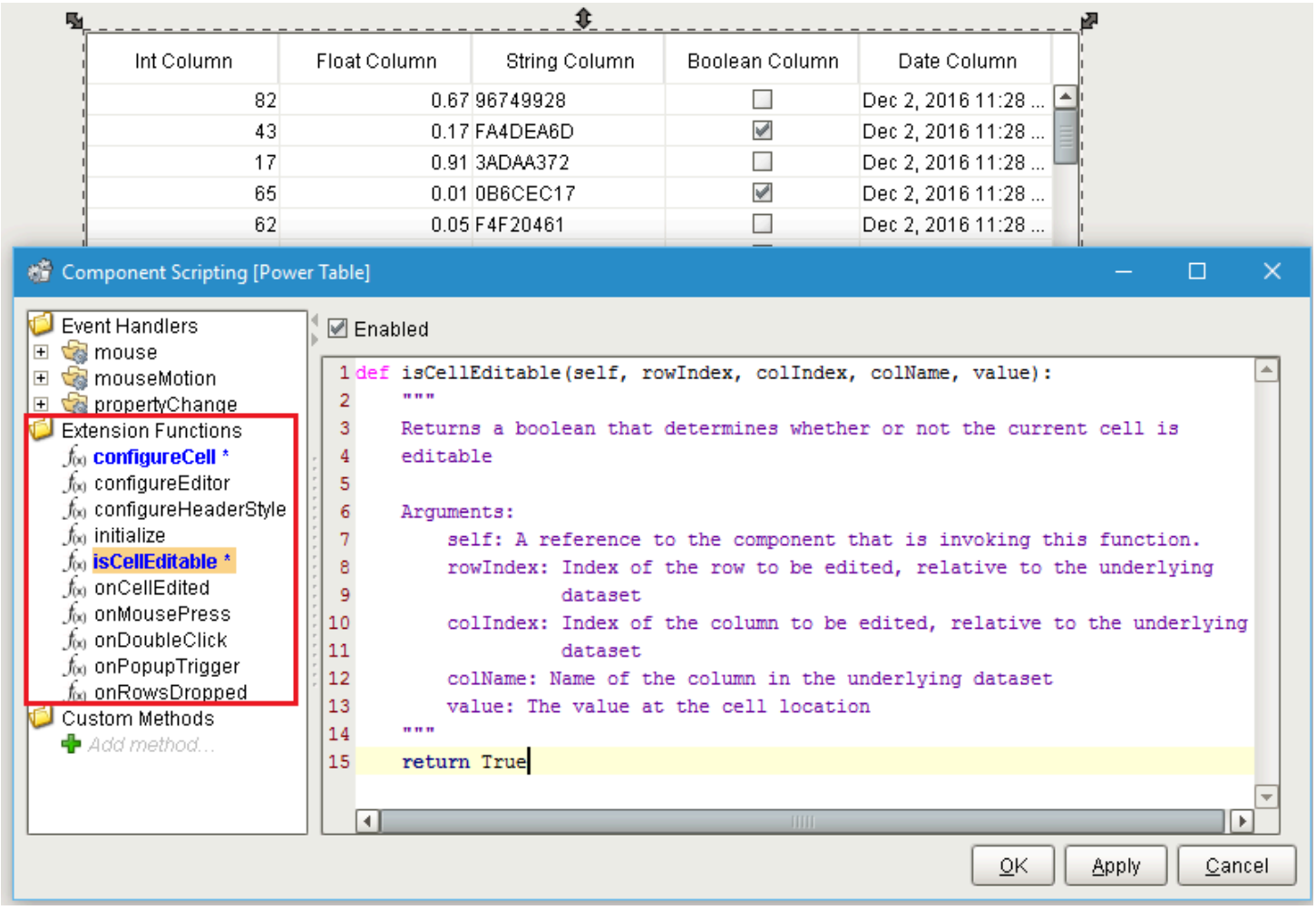
- Return
    - None

Working With Power Table Extension Functions

The Power Table offers significantly more configuration options than the classic Table, including a number of extension functions, which allow further configuration of the Power Table's functionality and appearance using scripting. This page describes the configuration options offered by the Power Table extension functions.

Overview

The Power Table component offers the ability to greatly expand available configuration options by enabling and defining its extension functions. Available functions are listed in the left pane of the Component Scripting window. Any number (or none at all) of the functions may be defined on the same Power Table component as needed.



Each extension function has a Python-style Docstring documenting the parameters the function takes and what the function returns. The parameters are passed into the function by the table component

and their values are available for use inside the function definition. This means that each function has a predefined list of parameters and a specific return value, neither of which can be changed.

Extension Functions vs. Event Handlers

Unlike an event handler function, an extension function is called by the table component itself. This means that the function does not need to be called separately. If the Enabled box is checked, the table will call the function and execute the code inside the function definition.

Calling Extension Functions

Another difference between extension functions and event handlers is when they are called. While some extension functions, like `onMousePress` or `onPopupTrigger`, are called in response to a specific event, other functions, like `configureHeaderStyle` or `configureCell`, will run every time the table's dataset is refreshed, and yet other functions like `initialize` or `configureEditor` will run only when the table first loads. When implementing a particular extension function, make sure to consider when and how many times each function is called in order to avoid introducing undesirable results, such as extended execution times.

Filtering Using Function Parameters

Extension function parameters can be used to to selectively apply the formatting defined in the function body. Below is the list of parameters and examples of some of the common conditions that can be used. When using examples offered below, make sure to verify the specific parameter used in the filter condition is available in the function you are defining.

Filter by rowIndex, colIndex, colName

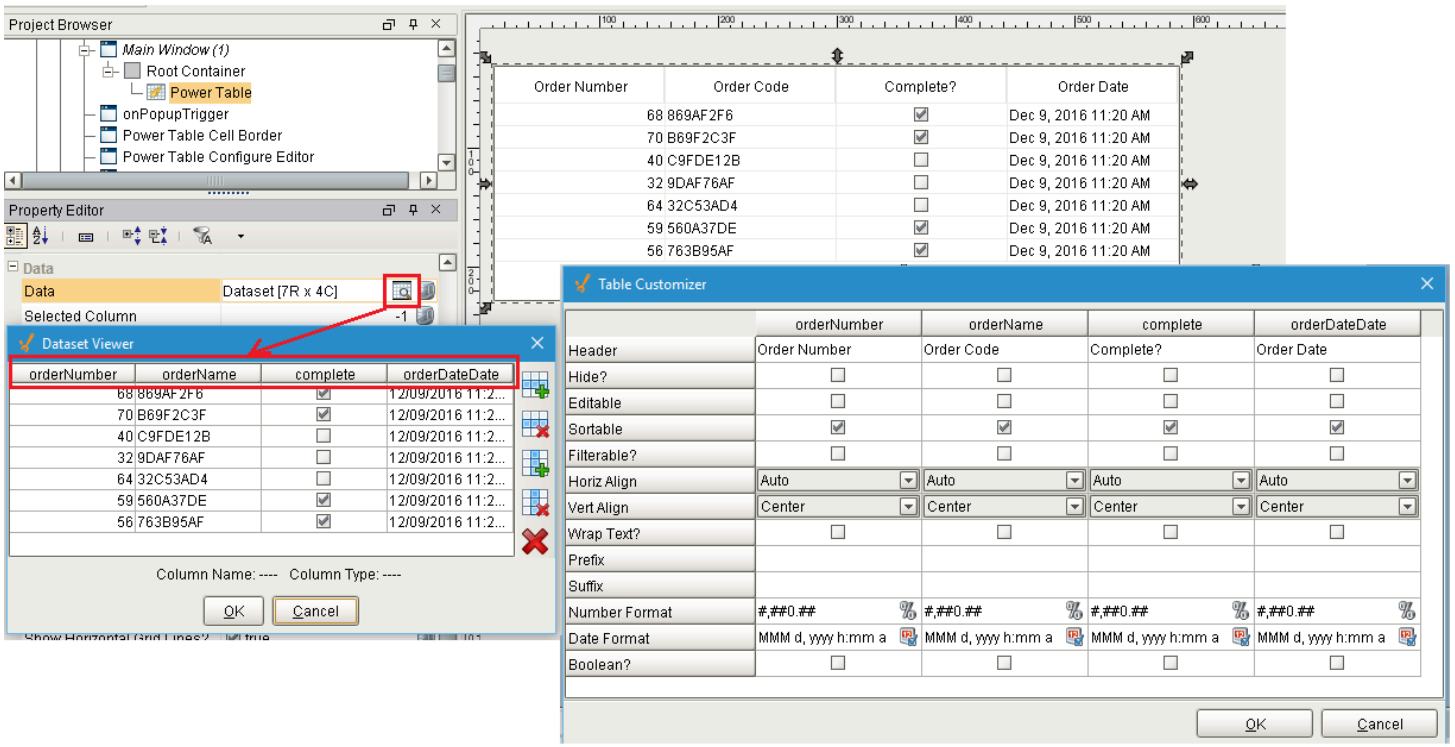
This allows you to apply modifications only to specific row(s) and/or column(s).

Purpose	Code
Apply the changes only to the leftmost column of the table's dataset.	`if colIndex == 0:` `  #do something`
Apply the changes only to the first ("zero-th") row of the table's dataset.	`if rowIndex == 0:` `  #do something`
Apply the changes to every even-numbered column of the table's dataset.	`if colIndex % 2 == 0:` `  #do something`
Apply the changes to every 10th row of the table's dataset.	`if rowIndex % 10 == 0:` `  #do something`

Instead of `colIndex`, you can filter for `colName`, which refers to the **name** of the table dataset column. This is useful when you are not certain what the order or the column would be (e.g. the query on the **Data** property binding is dynamic, and column order may change). Another common use case is applying the changes to a number of columns with names following a particular naming convention.

Purpose	Code
Apply changes to the *orderNumber* column.	`if colName == "orderNumber":` `  #do something`
Apply changes to all columns whose names contain the word "order", such as *orderName*, *orderDate*, *orderNumber*.	`if "order" in colName:` `  #do something`

When filtering for `colName`, make sure to use the name as shown in the table's Data property dataset viewer. This is especially important if the column names have been changed using the Table Customizer.



It is also possible to filter using row or column order as they appear on the screen, which can be different from their order in the dataset if sorting was applied, or if columns were hidden or otherwise rearranged:

Purpose	Code
Apply changes to every even-numbered row as shown in the table, even if the row order was re-arranged by sorting or filtering.	`if rowView % 2 == 0:` `  #do something`

Filter for the Cell Currently Selected by the User

This allows you to define changes based on whether the cell is selected or not.

Purpose	Code
Apply changes only to the currently selected cell.	`if selected:` `  #do something`
Apply changes to all cells except the currently selected cell	`if not selected:` `  #do something`

Filter Based on the Cell Value

Some extension functions have `value`, `textValue`, `newValue`, and `oldValue` parameters available.

The value parameter refers to the value as defined in the table's *Data* dataset.

In contrast, the `textValue` parameter applies to the value as it is displayed in the table column after any formatting used has been applied. Filtering on this parameter value is useful when, for example, a certain date format has been set for the date column using the Table Customizer.

The `newValue` and `oldValue` parameters, when available, can be used to filter based on what the value being edited changed *from* or *to*.

Purpose	Code
Apply changes only if the value in the cell is greater than 100.	`if value > 100:` `  #do something`
Apply changes only if the value in the cell is less than the value in the cell to its right.	`if value < self.data.getValueAt(rowIndex, colIndex+1):` `  #do something`
Apply changes only if the displayed cell value contains *Sunday*.	`if "Sunday" in textValue:` `  #do something`
Apply changes only if the cell value is being changed to a lesser value.	`if newValue < oldValue:` `  #do something`

TIP

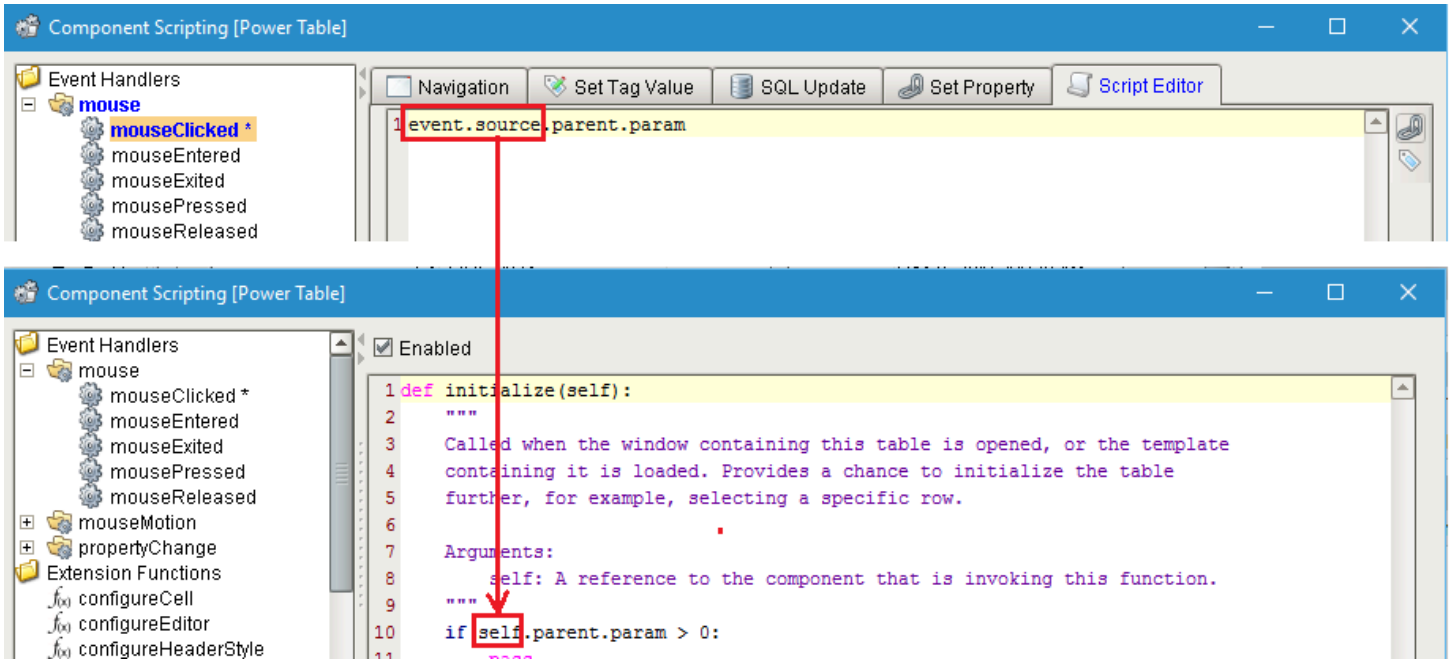
Be careful comparing across the board if table has non-numeric columns as you will likely get unexpected results. For details on comparisons between different datatypes in Python, see https://docs.python.org/2/library/stdtypes.html#comparisons

Filter Based on Properties of the Power Table Component

Filtering options are not limited to the parameters of the extension function being defined. Extension functions also allow filtering based on the properties of the Power Table component itself, or properties of other components on the screen.

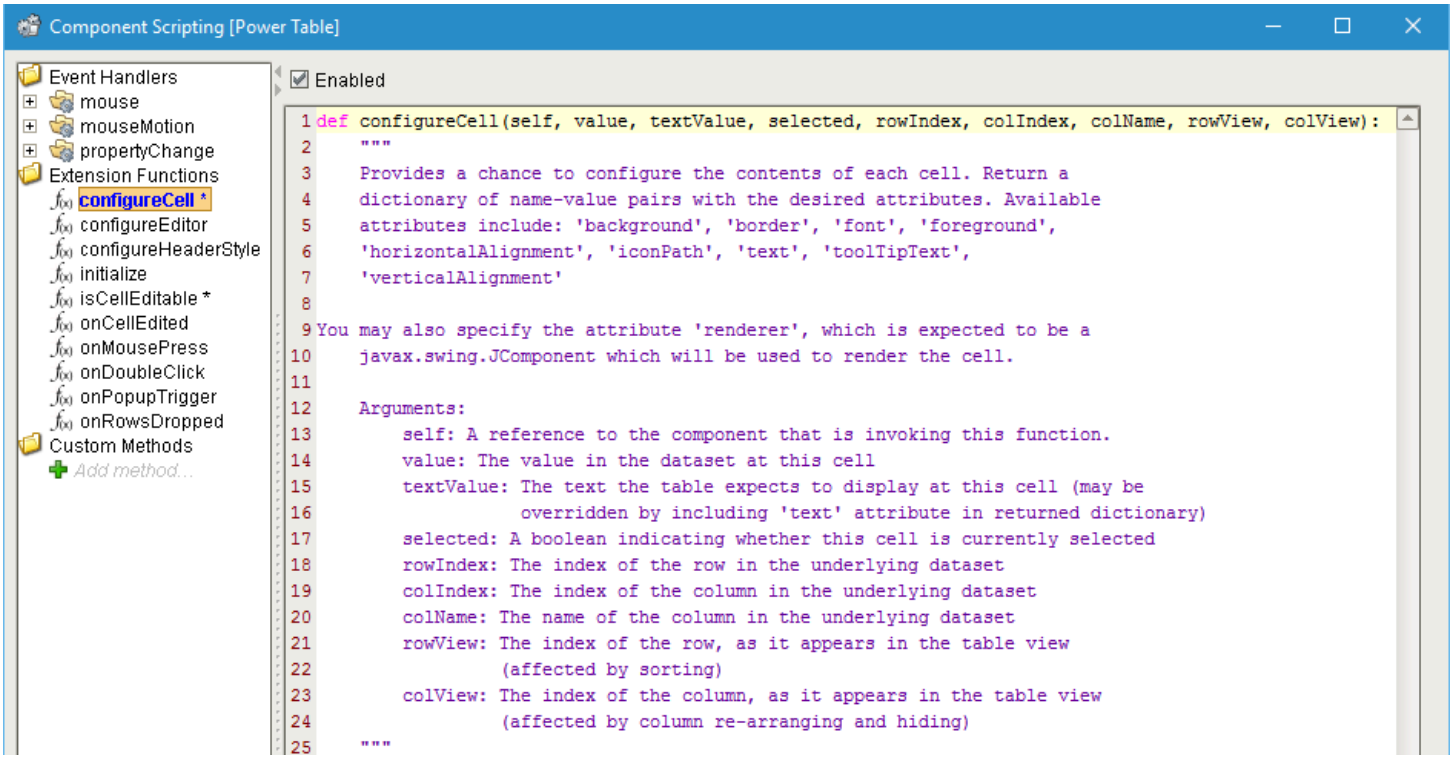
Purpose	Code
Apply changes only if the table's *Data* property dataset has more than 10 columns.	`if self.data.columnCount > 10:` `  #do something`
Apply changes only if the table's *Sorting Enabled* property is set to True (the checkbox is checked).	`if self.sortingEnabled:` `  #do something`
Apply changes only if the value of the *param* property on the table's parent component is greater than 5.	`if self-parent.param > 5:` `  #do something`

You can check the scripting names of the properties listed in the Property Editor by hovering over the property name or by using an event handler function on the same Power Table, and then replacing the `event.source` part of the reference with `self` (both of which are references to the Power Table component you are working with).



Working with Specific Functions

configureCell



The `configureCell` extension function provides a way to configure a cell's background and foreground colors, font properties, vertical and horizontal alignment of the text in the cell, cell text itself, icon displayed along with or instead of cell text, and any tooltip text shown when the cell is hovered over.

This function runs once for each cell, every time the cell needs to render. This means that for a table with 10 rows and 10 columns, this function will be called at least 100 times (at least once for each of the 100 cells) when the table initializes, or when the dataset is refreshed, etc. This should be taken into account when implementing this function, and any potentially long running code should be avoided.

The value given to the return statement of the function must be a dictionary of name-value pairs. The general format for such dictionary is: `{"attribute1": value1, "attribute2": value2}`

Configure Cell Background and Foreground Colors

Cell background and foreground (font) color are some of the most common attributes to configure.

The color value used to set these properties should be of type Color (java.awt.Color), or a 6-digit hexadecimal color string preceded by the hash (#) sign (e.g. '#FFFFFF'). On Ignition 8.3+, you can use the `system.vision.color` function to create a Color object.

Purpose	Code
Set background color to light blue for every even column.	`if rowIndex % 2 == 0:` `  return {'background': '#EAF2F8'}`
Invert colors for the selected row: set the background color to a darker shade of blue, set the foreground (font) color to a light shade of blue.	`if selected:` `  return {'background': Color(21, 67, 96),'foreground': Color(169`
Use different shades of blue to set the row background color to alternate between white, light blue, and slightly darker blue. Keep the selected row background default.	`if selected:` `  return {'background': self.selectionBackground}` `if rowIndex % 3 == 1:` `  return {'background': system.vision.color(234, 242, 248)}` `elif rowIndex % 3 == 2:` `  return {'background': system.vision.color(169, 204, 227)}`

Configure Cell Border

The `border` attribute allows to modify how the cell border is rendered. It expects a value of type **javax.swing.border.Border**.

Purpose	Code
Display a 2-pixel thick red border around the currently selected cell.	`from javax.swing.border import LineBorder` `if colIndex == self.selectedColumn and rowIndex == self.selected` `  return {'border': LineBorder(system.vision.color(255, 0, 0), 2)`

Please refer to the class API documentation at the link above for detailed information on available border types and available constructors.

Configure Cell Font

The `font` attribute allows changing font (font face, variant, color, thickness, and some other properties). This attribute expects a value of type **java.awt.Font**.

Purpose	Code
Set font face, weight, and size for cells whose value is less than 25.	`if colName == "results" and value < 25:` `  return {'font': Font("Georgia", Font.BOLD, 15)}`

To ensure visual consistency across all systems running the project, specify only fonts that you know will be available on the Client machines.

Configure Horizontal and Vertical Alignment

- Horitontal Alignment: By default, Power Table cell content is right-aligned for numeric column types, left-aligned for String and Date column types and centered for Boolean column types or Integer column types set to display as Boolean using the Table Customizer. This can be customized by setting the horizonalAlighment property to LEFT, CENTER, or RIGHT constants from the **javax.swing.SwingConstants** class.

Purpose	Code
Right-align text in all cells.	`from javax.swing import SwingConstants` `return {'horizontalAlignment': SwingConstants.RIGHT}`

- Vertical Alignment: By default, Power Table cell content is vertically centered. This can be customized by setting the `verticalAlignment` property to TOP, CENTER, or BOTTOM constants from the **javax.swing.SwingConstants** class. Row height may need to be increased for the visual difference to show.

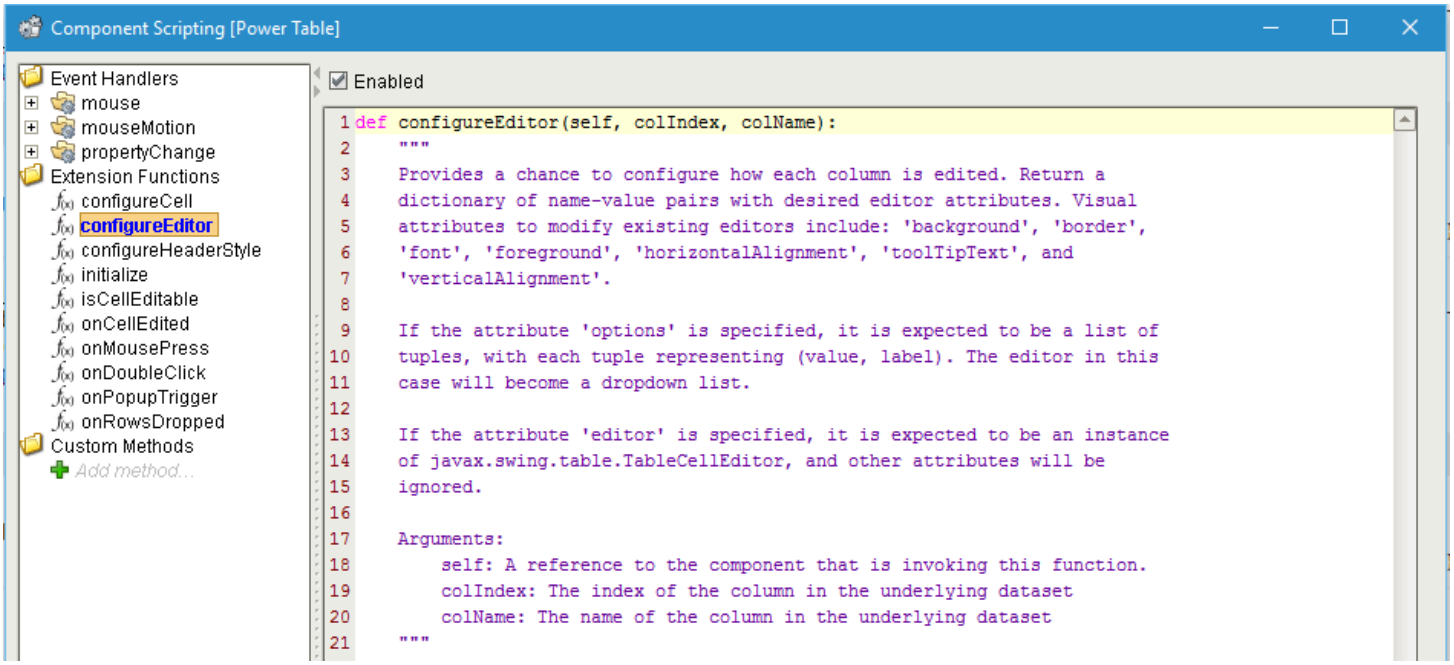
Purpose	Code
Bottom-align text in all cells.	`from javax.swing import SwingConstants` `return {'verticalAlignment': SwingConstants.BOTTOM}`

Modify Displayed Text, Add Icons and Tooltips

The `iconPath` property allows to specify a path to the icon stored using the Image Management tool.

Purpose	Code
Change the text of cells in the specified column if the cell value is less than 10.	`if colName == "Results" and value < 10:` `  return {'iconPath': 'Builtin/icons/16/check2.png'}`
Add an icon to the text of cells in the specified column if the cell value is less than 10.	`if colName == "Results" and value < 10:` `  return {'text': 'Success!'}`
Replace text in the cells in the specified column with an icon if the cell value is less than 10.	`if colName == "Results" and value < 10:` `  return {'text': '', 'iconPath' : 'Builtin/icons/16/check2.png'}`
Specify tooltips for the cells in the column depending on the value.	`if colName == "Results":` `  if value < 10:` `  return {'toolTipText': 'Pass'}` `  else:` `  return {'toolTipText': 'Fail'}`
Move the cell text to tooltip; make the cell display a specific icon depending on the value in the cell.	`if colName == "Results":` `  if value < 10:` `  iconPath = 'Builtin/icons/16/check2.png'` `  else:` `  iconPath = 'Builtin/icons/16/warning.png'` `  return {'text': '', 'iconPath': iconPat, 'toolTipText': value}`

configureEditor

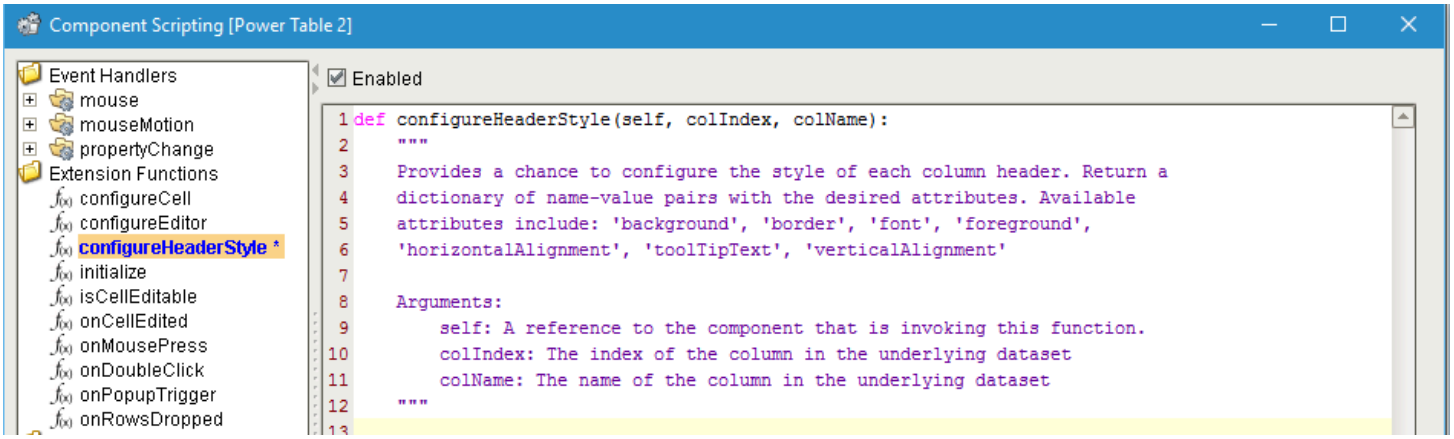


The `configureEditor` extension function allows to configure the visual attributes of the cell editor for all cells in the specified editable column. Unlike the changes applied using the `configureCell` function, this function allows to customize the appearance and behavior of an editable cell when it is being edited.

Purpose	Code
When a cell is being edited, set its background to light blue, use dark blue 15-pixel bold Sans-serif font, horizontally center cell contents, and specify the tooltip.	`from java.awt import Font` `from javax.swing import SwingConstants` `from javax.swing.border import LineBorder` `return {'background': system.vision.color(235, 245, 251), 'foreg`
Make the table cell display a dropdown list.	`shifts = [(0, 'AM Shift'), (1, 'PM Shift')]` `leads = [(0, 'Joe'), (1, 'Jane'), (2, 'Jim')]` `if colName =='Shift':` `return {'options': shifts}` `elif colName =='Supervisor':` `return {'options': leads}`

For advanced configuration options, the `editor` property can be specified. This is advanced functionality requiring the ability to work with Java Swing API. For more information, please refer to **javax.swing.table.TableCellEditor** class documentation.

configureHeaderStyle

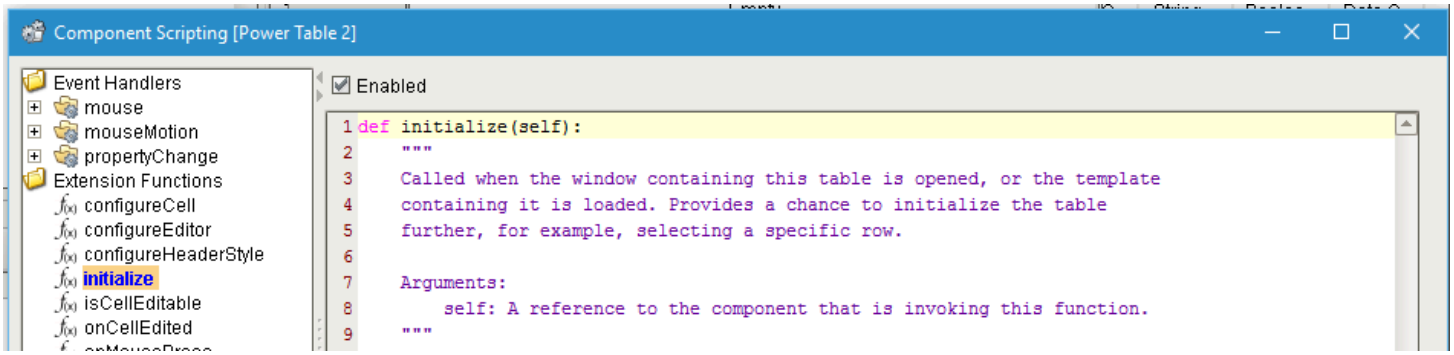


The `configureHeaderStyle` extension function provides a way to configure the background and foreground colors, border style and color, font properties, vertical and horizontal alignment of the text, as well as any tooltip text of the Power Table's header.

The value given to the return statement of the function must be a dictionary of name-value pairs. The general format for such dictionary is `{"attribute1": value1, "attribute2": value2}`

Purpose	Code
Configure the visual attributes of cell headers: make the background light blue, the font and border color dark blue; set font to 15 pixel bold Sans-serif, align text to the bottom of the header row. This example also shows how to set different border width for the 4 sides of the border box using the javax.swing.border.MatteBorder class. In this case the top border is 3 pixels wide, the bottom border is 2 pixels wide and side borders are invisible (0 pixels wide). For more information on border options, see **https://docs.oracle.com/javase/8/docs/api/javax/swing/border/package-summary.html**	`from java.awt im` `from javax.swing` `from javax.swing` `lightBlue = syst` `darkBlue = syste` `return {'backgrc`

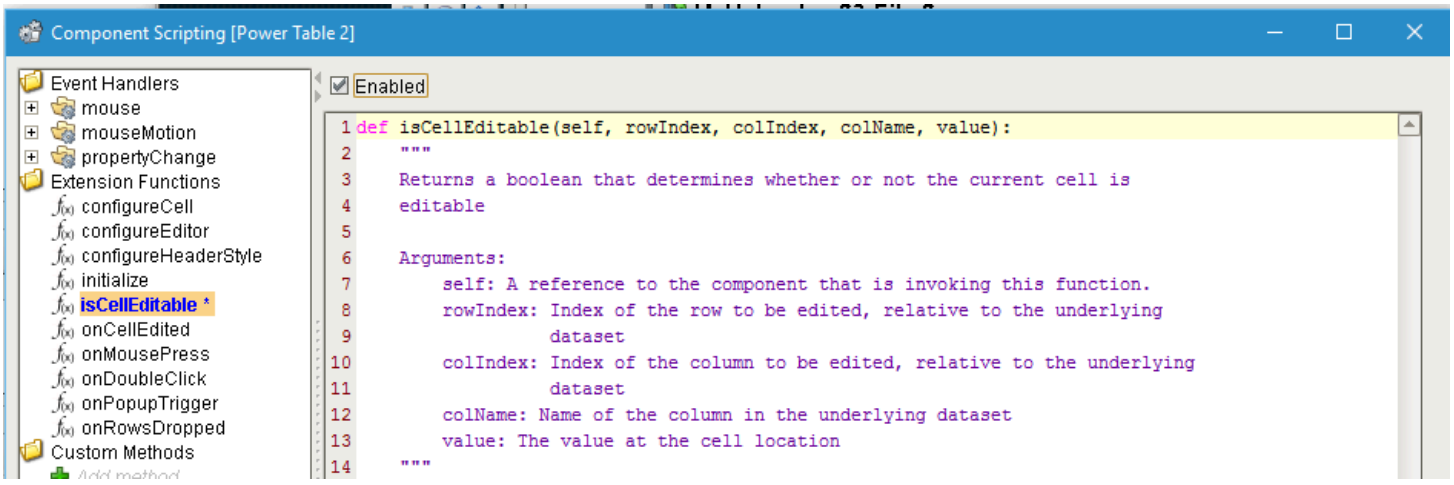
initialize



The `initialize` extension function allows to set up the table when it is first loaded (e.g. when the window first opens). It is not called again until next time the table is loaded (e.g. the window containing the table is closed and reopened).

Purpose	Code
If the table's Data dataset has more than 10 rows, adjust row height and font and enable row resizing. This example also shows a way to set font face using the Font class constants.	`from java.awt import Font` `if self.data.rowCount > 10:` `  self.rowResizeEnabled = True` `  self.rowHeight = 12` `  self.font = Font(Font.SANS_SERIF, Font.BOLD, 10)`

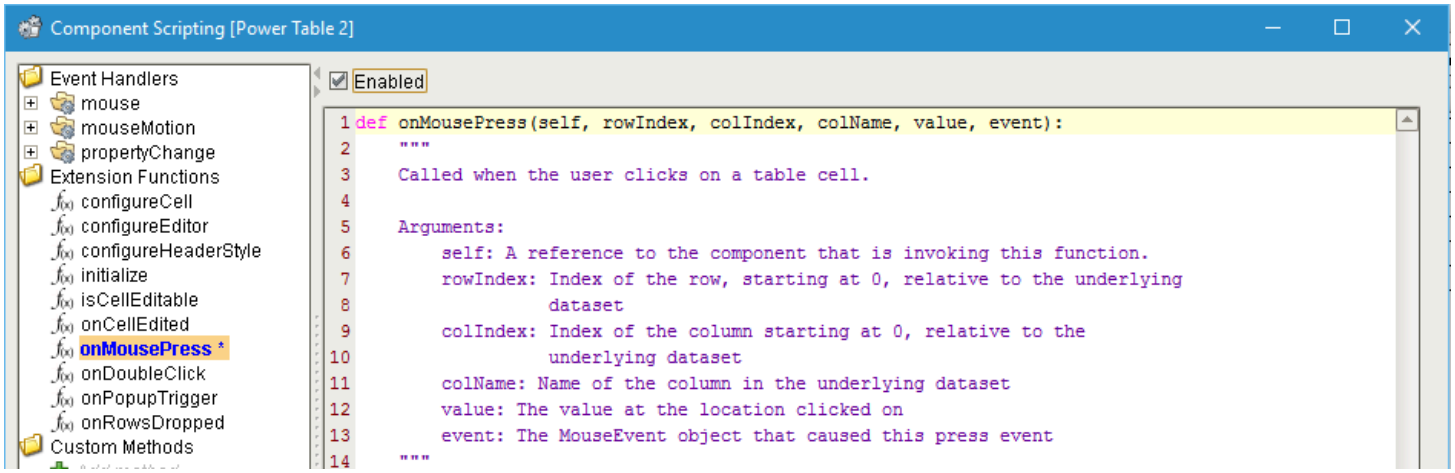
isCellEditable

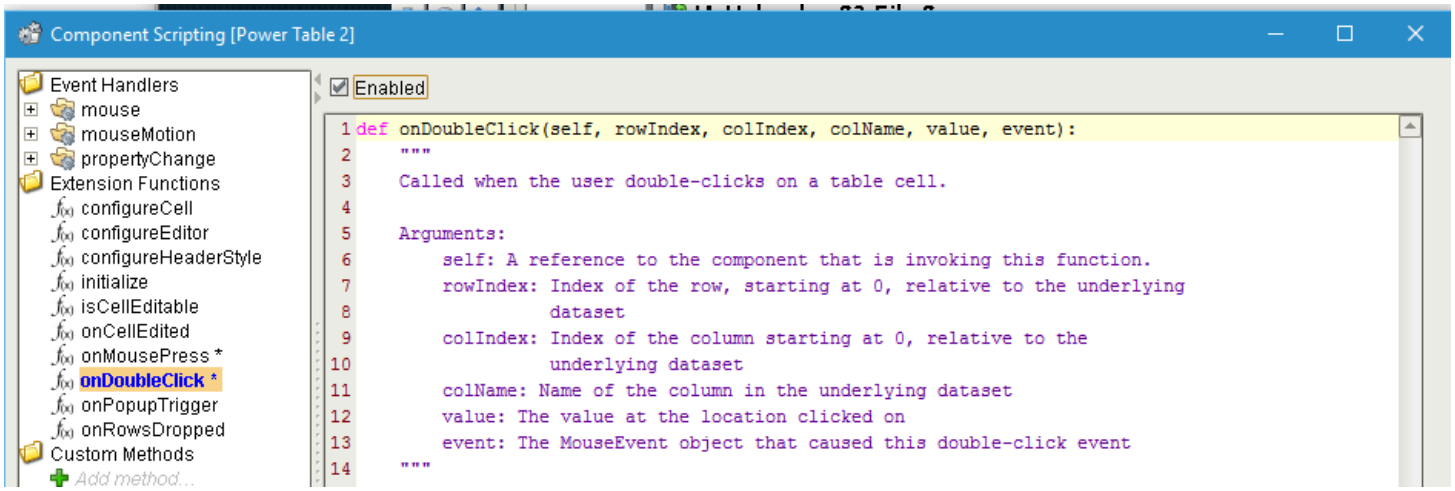


The `isCellEditable` extension function allows to make the current cell editable based on whether the specified condition is met.

Purpose	Code
Make a cell in the "Result" column editable if its value is < 50.	`if colName == "Result" and value < 50:` `  return True` `else:` `  return False`

onMousePress and onDoubleclick

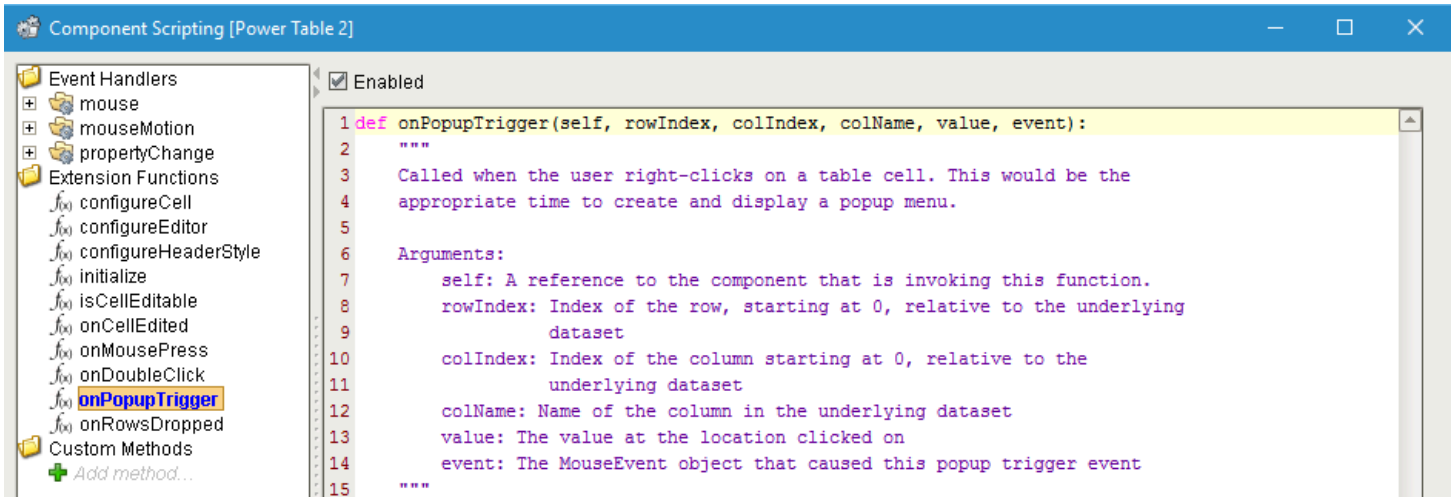




The `onMousePress` extension function allows to define actions for single left click event. The `onDoubleClick` extension function allows to define actions for the double left click event. Note that on systems using touch screens the double-click may not be available. Make sure to use these function (not the `mouseClicked` event handler function) for single or double left click events for better reliability, and to future-proof your code.

Purpose	Code
If the length of the URL string with surrounding white space removed in the table's "URL" column is greater than 0, open that URL in the Web Browser component when the URL is clicked.	`url = value.strip()` `if colName == "URL" and len(url) > 0:` `  self.parent.getComponent('Web Browser').currentUrl = url`

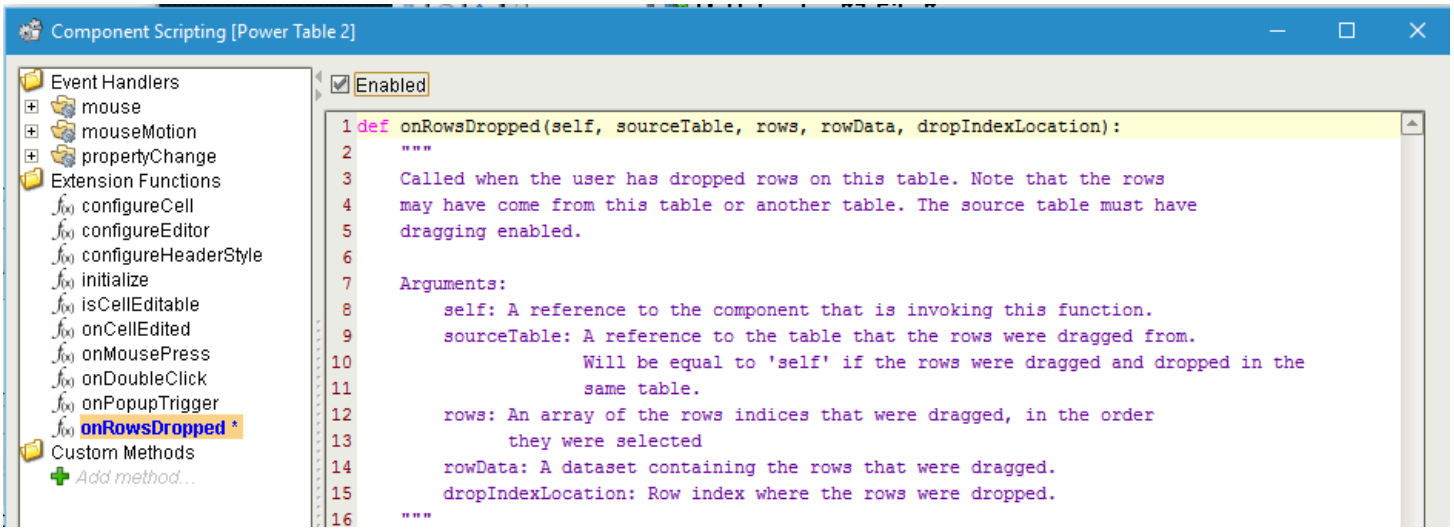
onPopupTrigger



The `onPopupTrigger` extension function allows to create a right-click popup menu. The menu will be aware of what cell called it. This information may be used to create a menu specific to the row, column, or cell, or can be ignored to create a generic menu for the whole table.

Purpose	Code
Create a generic right-click popup menu with options to export the table data as CSV or to print the table.	`def exportCSV(event):` ` filePath = system.dataset.exportCSV("data.csv", 1, self.data)` ` if filePath:` `  system.vision.showMessage("Saved as %s" % filePath)` ` else:` `  system.vision.showMessage("Failed to save")` `def printTable(event):` ` self.print(showDialog = True)` `# create a menu containing two items` `menuItemNames = ["Export data as CSV", "Print table"]` `# assign the functions defined above to the menu items` `menuItemFunctions = [exportCSV, printTable]` `# create a popup menu from the items and actions above` `menu = system.vision.createPopupMenu(menuItemNames, menuItemFunc` `# show the menu on right-click` `menu.show(event)`
Create right-click popup menu with options to delete or duplicate the selected row.	`def deleteRow(event):` ` self.data = system.dataset.deleteRow(self.data, self.selectedRo` `def duplicateRow(event):` ` # create the new row` ` newRow = []` ` # fill the new row with the data from the selected row` ` for colNdx in range(self.data.columnCount):` `  newRow.append(self.data.getValueAt(self.selectedRow, colNdx))` ` # insert the duplicate row after the selected row` ` self.data = system.dataset.addRow(self.data, rowIndex + 1, newR` `if self.selectedRow > 0:` ` # create a menu containing two items` ` menuItemNames = ["Delete selected row", "Duplicate selected row` ` # assign the functions defined above to the menu items` ` menuItemFunctions = [deleteRow, duplicateRow]` ` # create a popup menu from the items and actions above` ` menu = system.vision.createPopupMenu(menuItemNames, menuItemFun` ` # show the menu on right-click` ` menu.show(event)`

onRowsDropped



The `onRowsDropped` extension function allows to define what should happen when a row from another Power Table on the same window is dropped onto the Power Table that has this function defined. It must be defined on the destination table for row drag-and-drop to work. The function definition should have the code to append the new row to the existing dataset.

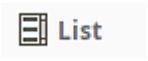
Purpose	Code
Drop one or more rows from source table into the destination table when both tables have the same number and type of columns, in the same order:	

```
for rowNdx in range(rowData.rowCount):
 if rowData.columnCount == self.data.columnCount:
  newRow = []
  for colNdx in range(rowData.columnCount):
   newRow.append(rowData.getValueAt(rowNdx, colNdx))
  self.data = system.dataset.addRow(self.data, dropIndexLocation
 else:
  system.vision.showError("Drop failed: columns do not match")
```

Vision - List



Component Palette Icon:



Description

The List component displays a list of options, allowing freeform selection of the items. It is powered by a Dataset, from which it displays the first column.

Properties

Property	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), and Field 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	int	.cursorCode	Common

Property	Description	Property Type	Scripting	Category
	Resize, or SE Resize.			
Data	A dataset that The data for the list. If multiple columns exist, the first will be used.	Dataset	.data	Data
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See **Color Selector**.	Color	.foreground	Appearance
Layout Orientation	This property defines the orientation of the list elements.	int	.layoutOrientation	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Opaque	If false, backgrounds are	boolean	.opaque	Common

Property	Description	Property Type	Scripting	Category
	not drawn. If true, backgrounds are drawn.			
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Row Height	An integer specifying the row height, or -1 for automatic row height.	int	.rowHeight	Appearance
Selected Background	The color of the background for the selected cell(s).	Color	.selectedBackground	Appearance
Selected Focus Border	The border for the selected, focused cell.	Border	.selectedFocusBorder	Appearance
Selected Foreground	The color of the foreground for the selected cell(s). See **Color Selector**.	Color	.selectedForeground	Appearance
Selected Index	The index of the selected cell, or -1 if none.	int	.selectedIndex	Data
Selection Mode	This mode determines if only one cell can be selected at once,	int	.selectionMode	Behavior

Property	Description	Property Type	Scripting	Category
	or single or multiple intervals.			
Styles	Contains the component's styles.	Dataset	.styles	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Visible Row Count	An integer specifying the preferred number of rows to display without requiring scrolling.	int	.visibleRowCount	Appearance

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

See the **Vision - List Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

- **Vision Component Customizers**
- **Style Customizer**

Examples

Code Snippet

```
# This example will create a dataset, and assign the dataset to
#  the List component's Data property.
headers = ["my header"]
data = [["Thing 1"],["Thing 2"],["Thing 3"]]

dataset = system.dataset.toDataset(headers, data)

# Assign the dataset. The path below may need to change depending on
# what component is triggering this script.
event.source.data = dataset
```

Code Snippet

```
# The following code will print the selected value to the console when called on
the 'mouseClicked' event handler.
value = event.source.getSelectedValue()
print(value)
```

Code Snippet

```

```

```
# The following code uses setSelectedValues to set the selection on the component.
# Assuming the List component contains string values of either "Thing 1" or "Thing 2", both items will be selected.

# Build a Python list of things to check for in the List component
valueList = ["Thing 1", "Thing 2"]

# Locate the List component in the window, and call setSelectedValues, passing the valueList as an argument.
event.source.setSelectedValues(valueList)
```

Vision - List Scripting Functions

This page details the various component and extension functions available for **Vision's List component**.

Component Functions

.addSelectionInterval(start, end)

- Description
  - Adds the options at indexes start through end (inclusive) to the selected options.
- Parameters
  - int **start** - The first index (starting at 0) to add to the selection.
  - int **end** - The last index (starting at 0) to add to the selection.
- Return
  - None

.clearSelection()

- Description
  - Clears the current selection, making nothing selected.
- Parameters
  - None
- Return
  - None

.getSelectedIndices()

- Description
  - Returns a list of the selected indices in increasing order. Returns an empty list if nothing is selected.
- Parameters
  - None
- Return
  - List of Integers

.getSelectedValue()

- Description
  - Returns the currently selected value, or None if the selection is empty.
- Parameters
  - None
- Return
  - Object

.getSelectedValues()

- Description
  - Returns a list of the currently selected values. Returns an empty list if the selection is empty.
- Parameters
  - None
- Return
  - Object[]

.isSelectedIndex(index)

- Description
  - Checks whether or not the given index is currently selected.
- Parameters
  - int index
- Return
  - boolean

.isSelectionEmpty()

- Description
  - Checks to see if anything is selected in the list or not.
- Parameters
  - None
- Return
  - boolean

.setSelectedValue(value)

- Description

    - Sets the currently selected value to the argument, if found in the list.
- Parameters
    - Object **value**
- Return
    - None

.setSelectedValues(valueList)

- Description
    - Sets the currently selected values in the component, selecting multiple options. The options selected are determined by the valueList parameter, which is expected to be a list of literal values that map to options in the list.
- Parameters
    - Object **valueList** - Python list containing values that should map to options in the component.
- Return
    - None

Extension Functions

This component does not have extension functions associated with it.

Vision - Tree View



Component Palette Icon:



Description

The Tree View component can display any tree hierarchy. It is configured by filling in a dataset. Each column title in the dataset is a property of the Tree View Customizer.

Each row in the dataset will become a node in the tree. Each node has a path that determines its location in the tree, for example, "West Area/Process/Valve1". The Separation Character property dictates how the paths are broken up. Any missing folder nodes needed by a leaf node are created implicitly. The other columns in the dataset besides "Path" are used to configure the look for the node, both when it is selected and when it is not. All column properties in the dataset are described in the **Tree View Customizer**.

Properties

Property	Description	Property Type	Scripting	Category
Auto Expand	If true, the tree will automatically expand the tree structure up to the level specified by Auto Expansion Level.	boolean	.autoExpand	Behavior
Auto Expansion Level	If Auto Expand is true, this is the depth level that will be expanded. Zero means expand-all.	int	.autoExpansionLevel	Behavior
Auto Sort	Whether or not to automatically sort the tree.	boolean	.autoSort	Behavior
Background Color	The background color of the component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL	Color	.background	Appearance

Property	Description	Property Type	Scripting	Category
	value. See **Color Selector**.			
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Default Closed Icon	The default closed icon if no icon is set.	String	.defaultClosedIconPath	Appearance
Default Leaf Icon	The default leaf icon if no icon is set.	String	.defaultLeafIconPath	Appearance
Default Node Background	The default background of a node if no background is	Color	.defaultBackground	Appearance

Property	Description	Property Type	Scripting	Category
	set. See **Color Selector**.			
Default Node Border	The default border of a node if no border is set.	Border	.defaultBorder	Appearance
Default Node Foreground	The default foreground of a node if no foreground is set. See **Color Selector**.	Color	.defaultForeground	Appearance
Default Node Selected Background	The default selected background of a node if no background is set. See **Color Selector**.	Color	.defaultSelectedBackground	Appearance
Default Node Selected Border	The default selected border of a node if no border is set.	Border	.defaultSelectedBorder	Appearance
Default Node Selected Foreground	The default selected foreground of a node if no foreground is set. See **Color Selector**.	Color	.defaultSelectedForeground	Appearance

Property	Description	Property Type	Scripting	Category
Default Open Icon	The default open icon if no icon is set.	String	.defaultOpenIconPath	Appearance
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Full Width Selection	Whether to paint the selection across the width of the tree. Default is true.	boolean	.fullWidthSelection	Appearance
Items	Contains the items of the tree view.	Dataset	.data	Data
Line Style	The tree's line style.	int	.lineStyle	Appearance
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common

Property	Description	Property Type	Scripting	Category
Name	The name of this component.	String	.name	Common
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Row Height	The height of each row in the tree.	int	.rowHeight	Appearance
Selected Item	The index of the currently selected item, or -1 if no selection.	int	.selectedItem	Data
Selected Path	The path of the currently selected item, or "" if no selection.	String	.selectedPath	Data
Selection Fill Color	The background color to fill the selection width with. See **Color Selector**.	Color	.selectionFillColor	Appearance
Selection Mode	What kind of selection	int	.selectionMode	Behavior

Property	Description	Property Type	Scripting	Category
	regions does the tree allow. Options are Single, Multiple - Contiguous, and Multiple - Discontiguous.			
Separation Character	The separation character for the path.	String	.separationCharacter	Behavior
Show Root Handles	Whether or not to show handles next to parent nodes.	boolean	.showRootHandles	Appearance
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

See the **Vision - Tree View Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

- **Tree View Customizer**
- **Component Customizers**

Examples

Expression Snippet

```
//The Selected Item property will be updated as the user selects different nodes
in the tree.
//It represents the index in the Items dataset at which the node is defined. If
the selected
//node was implicitly created, the Selected Item will be -1.
//You can use this index to get the path and name of the selected node with an
expression binding like this:
if ({Root Container.Tree View.selectedItem}<0,"n/a",{Root Container.Tree
View.data}[{Root Container.Tree View.selectedItem},"text"])
```

Script Snippet

```
# This script will swap to the window that was double-clicked when placed on the
# mouseClicked event handler for the Tree View.
# It uses an extra "windowPath" column in the Items dataset containing the full
window path.
# (You can add extra columns to Items as long as the name doesn't match a reserved
column.)
if event.clickCount == 2:
    row = event.source.selectedItem
    data = event.source.data
    if row != -1:
        # Grab the window path value out of the tree view's items dataset
        windowPath = data.getValueAt(row, "windowPath")
        system.vision.swapTo(windowPath)
```

Tree View Scripting Functions

This page details the various component and extension functions available for Vision's **Tree View** component.

Component Functions

.clearSelection()

- Description
  - Clears the current selection.
- Parameters
  - Nothing
- Return
  - Nothing
- Scope
  - Client

.collapseAll()

- Description
  - Collapses all nodes in the tree.
- Parameters
  - Nothing
- Return
  - Nothing
- Scope
  - Client

.expandAll()

- Description
  - Expands all nodes in the tree.
- Parameters
  - Nothing
- Return

    - Nothing
- Scope
    - Client

.getSelectedItems()

- Description
    - Returns a list of the selected item's indexes. These are the row indexes that the selected tree nodes were found in the underlying dataset. Implicitly created folder nodes that have no index will not be included.
- Parameters
    - Nothing
- Return
    - List of Integers
- Scope
    - Client

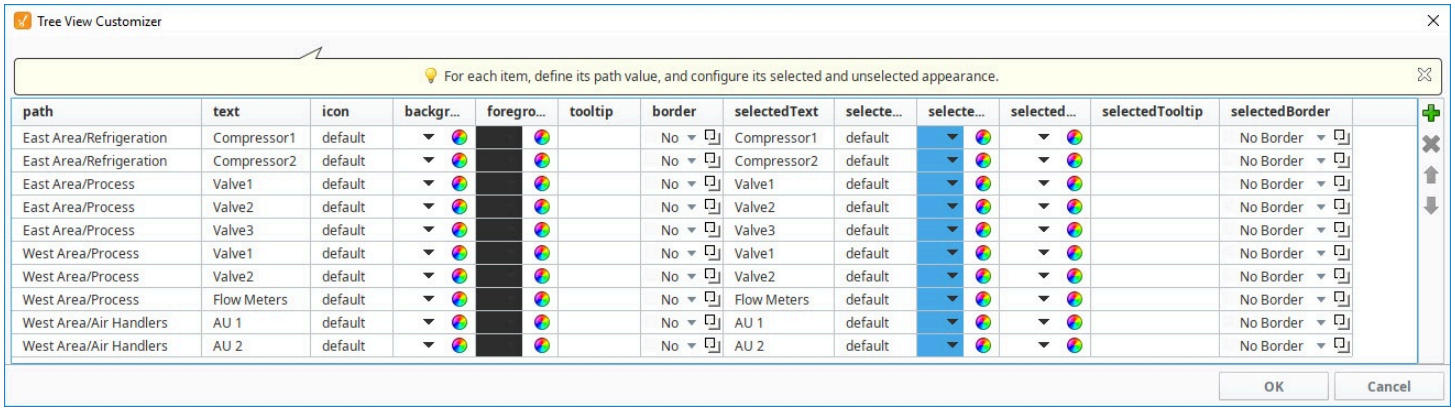
.getSelectedPaths()

- Description
    - Returns a list of the selected item's paths. A path to an item is the path to its parent plus its normal (non-selected) text.
- Parameters
    - Nothing
- Return
    - List of Strings
- Scope
    - Client

Extension Functions

This component does not have extension functions associated with it.

Vision - Tree View Customizer



Description

The Tree View has its own customizer which allows you to easily configure the items dataset property. The customizer provides some useful dropdowns and color selectors for certain properties that require more than just a name or a path. You can add and remove nodes, and change the node hierarchy and appearance through the properties in the dataset.

While the Customizer allows you to configure the columns of the Items dataset, the customizer will not display any columns that the user adds to the dataset. However, user added columns are still configurable in the dataset itself, and can be used to store additional information about each item such as a window path.

Customizers

The Tree View Customizer allows you to easily configure how you want the tree view to look to users. When you open the customizer for the first time, you'll notice the dataset contains some predefined nodes and settings. Each row in the dataset represents a node in the tree. Each column in the dataset represents properties that configure the appearance of the tree to look a certain way.

Configuring the Tree View Customizer is very straightforward. To add a node to the tree, click the green icon on the right side of the window, and a new row will be added at the the bottom of the dataset. All the columns will default to the predefined properties with the exception of the "path" to the node's location. This field will be blank so you need to enter a path to the node. You can edit any of the of the preset properties. At a minimum, you should always edit the **Text** and **SelectedText** properties replacing the default names with a more appropriate name so the item is easily identifiable when it is selected and unselected in the tree. You can also move a node up or down the tree

hierarchy using the **Move Up** or **Move Down** arrows on the right side of the window. To delete a node from the tree, simply select the node and hit **Delete**.

The additional properties are optional, but can enhance your tree view for your users. For example:

- To change an icon for any node in the tree, choose an icon from the Image Management Tool. All you need to do is right click on the icon in the Image Management tool and select Copy Path, and paste it in the Icon field for that node.
- Add a tooltip for any item in the tree by simply typing in your tooltip in the Tooltip field for that node. When you hover over the item in the tree view, you'll see your tooltip.
- Add a foreground and background color for any item in the tree when it is selected or unselected.
- Add a border for any item in the tree when it selected or unselected.

The references to optional properties in the table below mean that a dataset does not need to have them present in the dataset for the tree to render and function.

Properties

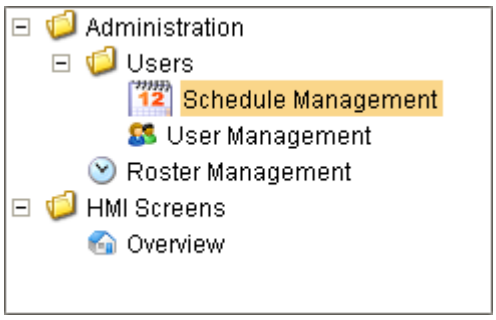
Property	Description
Path	Path that determines the node's locaton. Broken up into a list by splitting on the separation character.
Text	Text of the node while not selected.
Icon	Path to an icon for the node. Use the value: "default" to use the tree automatic folder/leaf icons. (optional)
Background	Controls the background appearance of the unselected item. A string column that will be coerced into a color for the unselected background. (e.g., "white" or "(255,255,255)". Use an empty string to use the default color. (optional)
Foreground	Control the foreground appearance of the unselected item. A string representation of the unselected foreground color. (optional)
Tooltip	If not empty, will be use as the tooltip for the node. (optional)
Border	A string that will be coerced into a border for the node while unselected. May be empty. (optional)
SelectedText	Text of the node while selected. (optional)
SelectedIcon	A path to an icon for the node while selected. Use the value: "default" to use the tree automatic folder/leaf icons. (optional)
SelectedBackground	Controls the background appearance of the selected item. A string representation of the the selected background color. (optional)
SelectedForeground	Controls the background appearance of the selected item. A string representation of the selected foreground color. (optional)
SelectedTooltip	If not empty, will be used as the tooltip for the node while selected. (optional)

Property	Description
SelectedBorder	A string that will be coerced into a border for the node while selected. May be emplty. (optional)

Examples

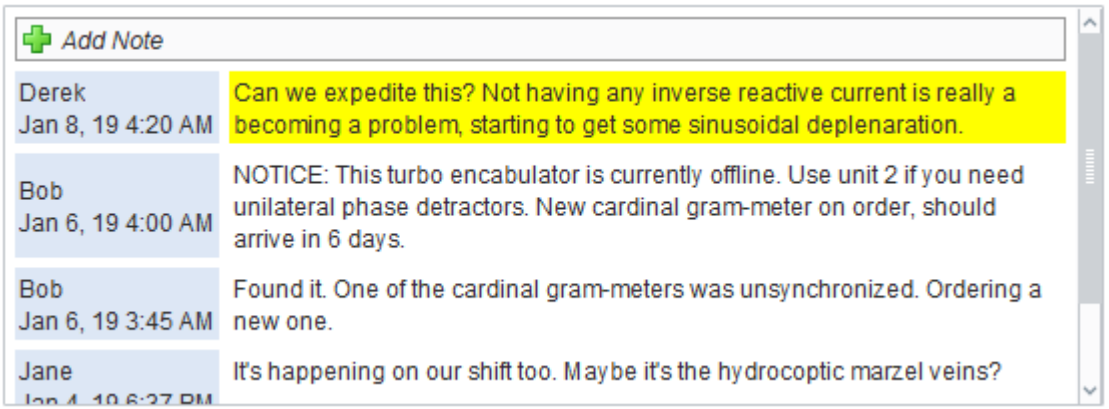
Tree View with Larger Version of SelectedIcons

Below is an example configuration of the tree view's items property. Notice how not all of the fields listed in the property table above are used, because there are certain properties that are not necessary to build our tree view. A larger version of the images was chosen for the SelectedIcon, so that when an item gets selected, not only does the background color change, but the size of the image changes as well.



Path	Text	Icon	Background	Foregrou
HMI Screens	Overview	Builtin/icons/16/home.png	color(255, 255, 255, 255)	color(0, 0, 255)
Administration/Users	User Management	Builtin/icons/16/users3.png	color(255, 255, 255, 255)	color(0, 0, 255)
Administration/Users	Schedule Management	Builtin/icons/16/calendar.png	color(255, 255, 255, 255)	color(0, 0, 255)
Administration	Roster Management	Builtin/icons/16/clock.png	color(255, 255, 255, 255)	color(0, 0, 255)

Vision - Comments Panel



Component Palette Icon:

Comments Panel

Description

The comments panel is used to power a blog-style comments system within your project. This can be useful for ad-hoc collaboration and communication between shifts, remote users, etc. This component is driven by a dataset that should be bound to a SQL query. Unlike most components, this component has built-in functionality to alter an external database. It expects three tables in the database, and that they are queried properly on the data property.

You can opt out of this three-table default system by simply making use of the Extension Functions on the component. See below for more details.

> **NOTE**
>
> The following section assumes the default configuration: all Extension Functions on the component are disabled.

Three-Table (Default) Configuration

Required Database Tables

The default behavior of the component expects three database tables be present under the same database connection, and each table needs to have certain columns with specific names.

Table: Notes

Stores all of the notes across the board.

Column Name	Description	Data Type
id	An auto-incrementing integer that is the primary key. This maps to the ID field in the dataset.	Integer
whoID	A mapping to the Username field in the dataset	Integer
tStamp	A mapping to the Timestamp field in the dataset	Date or Datetime
note	A mapping to the NoteText field in the dataset	Varchar
filename	A mapping to the AttachmentFilename in the dataset	Varchar
sticky	A mapping to the Sticky field in the dataset	Boolean or Integer
attachment	A column to hold the attachment data. LongBlobs do not exist in MSSQL, so a varbinary type must be used	LongBlob or Varbinary (depending on database)

Table: ItemNotes

Used to associate notes with other things. This allows you to have different sets of notes for different screens/objects.

Column Name	Description	Data Type
accountId	An automatically generated UUID for the Comment Panel instance. You can use the accountId in a WHERE clause on the data property so that the component only shows notes from a particular Comments Panel in the project.	Varchar
noteId	An integer that maps to the ID column on the Notes table	

Table: Users

A user mapping table that assigns an ID to each user on the table. This is easiest to do if a database authentication profile is used as the _users table automatically creates the required columns, but non-database authentication profiles can be used as long as the table is manually created and maintained.

Column Name	Description	Data Type
id	An integer that is inserted into the whoID column on the Notes table	Integer
username	The username of the user that created the note	Varchar

Configuring the Component

This component expects that its data property is populated with the following columns. The dataset in the Data property is very specific, and expects certain datatypes at precise positions. The order of **expected column positions** is listed below. Should the order of datatypes in the dataset differ from the order below, the names of the columns must match the **column names** below. Aliasing can be used to modify the names of the columns in the dataset.

The names do need to be exact, but different names can be used as long as the query that builds the dataset uses aliases. The data type for each column in your notes table must match the table below.

Column Name	Description	Data Type	Expected Column Position
id	an integer that should be the primary key for the notes table. Used for deleting and looking up attachments	integer	0
username	the user who added the note	string/varchar	1
timestamp	when the note was added	dateTime	2
notetext	The text of the note itself	string/varchar	3
attachmentname	filename for a file attached to the note	string/varchar	4
issticky	0 or 1 indicating whether or not the note is "sticky", which means it gets highlighted and put at the top	boolean or integer	5

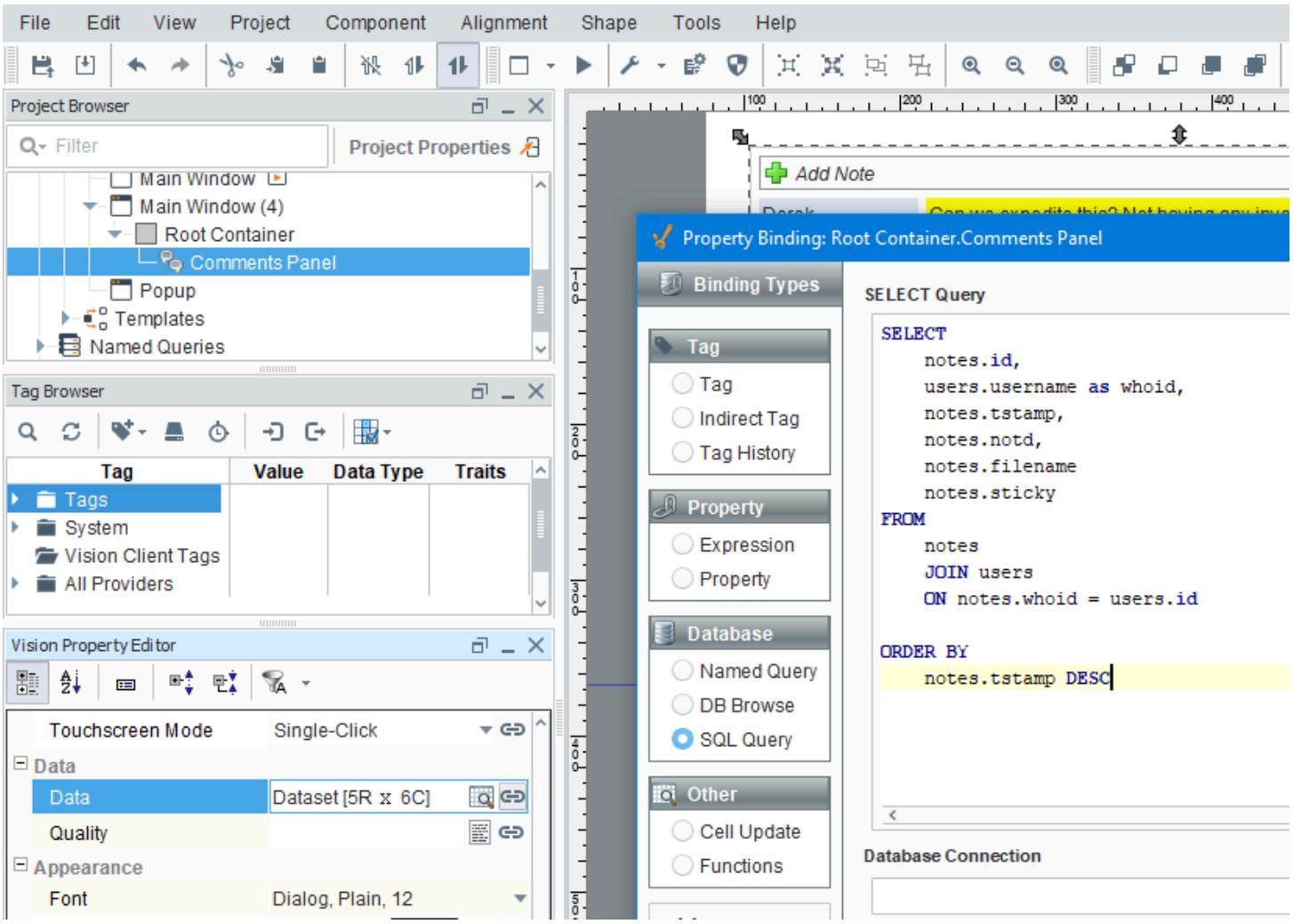
Example

The following query returns note data from the above tables, and displays the data on a Comments Panel component. This query should be placed in a SQL Query binding on the Data property

```
 SELECT
    notes.id,
    users.username as whoid,
    notes.tstamp,
    notes.note,
    notes.filename,
    notes.sticky

FROM
    notes
    JOIN users
    ON notes.whoid = users.id

ORDER BY
    notes.tstamp DESC
```



By default, users can remove their own comments, and comments can have files attached.

Custom Configuration

Enabling the Extension Functions on the component will allow for custom functionality on the component. Some examples are:

- **Store all note data on a single database table** - modify each Extension Function to run queries against a single database table
- **Save the attachment to a shared drive instead of a database column** - modify insertNote to save the attachment to a hard drive.
- **Allow users to delete all notes by role** - check the role of the user in canDelete and return True if the user has a specific role.

Properties

Property	Description	Property Type	Scripting	Category
Add Note Text	The word(s) used for the "Add Note" button.	String	.addNoteText	Appearance
Attach File Text	The word(s) used for the "Attach File" link.	String	.attachText	Appearance
Attachments Enabled	Controls whether or not files can be attached to notes.	boolean	.attachmentsEnabled	Behavior
Border	The border surrounding this component. Options are No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.	Border	.border	Common
Cancel Text	The word(s) used for the "Cancel" button.	String	.cancelText	Appearance
Data	Fill this DataSet in with the notes for the desired entity. Columns are: ID,	Dataset	.data	Data

Property	Description	Property Type	Scripting	Category
	Username, Timestamp, Note, Filename, IsSticky.			
Database Connection	Name of the database connection to run the queries against. Leave blank to use project's default connection.	String	.datasource	Behavior
Date Format	The format string to use for the date of the note.	String	.dateFormat	Appearance
Display Mode	Horizontal display mode will layout so that the comment header will be positioned to the left of the comment. Vertical display mode will have the comment header above the comment.	int	.displayMode	Behavior
Enabled	If disabled, a component cannot be used.	boolean	.componentEnabled	Common
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the	Color	.foreground	Appearance

Property	Description	Property Type	Scripting	Category
	component. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See **Color Selector**.			
Header Color	The background color of the header notes. See **Color Selector**.	Color	.headersColor	Appearance
Maximum Attachment Size	The maximum attachment size in bytes that will be accepted. A value of 0 means no limit.	long	.maxAttachmentSize	Behavior
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Note Color	The background color for notes. See **Color Selector**.	Color	.noteColor	Appearance

Property	Description	Property Type	Scripting	Category
Padding	The amount of padding between the notes.	int	.padding	Appearance
Quality	The data quality code for any Tag bindings on this component.	QualityCode	.quality	Data
Skip Audit	If true, update queries originating from this component will skip the audit system. Can be important when attachments are turned on.	boolean	.skipAudit	Behavior
Sticky Header Color	The background color of the header for sticky notes. See **Color Selector**.	Color	.stickyHeaderColor	Appearance
Sticky Note Color	The background color for sticky notes. See **Color Selector**.	Color	.stickyNoteColor	Appearance
Sticky Text	The word(s) used for the "Sticky" checkbox.	String	.stickyText	Appearance
Touchscreen Mode	Controls when this input component responds if	int	.touchscreenMode	Behavior

Property	Description	Property Type	Scripting	Category
	touchscreen mode is enabled.			
Touchscreen Keyboard Layout	Sets the touchscreen keyboard layout to use for this component.	String	.keyboardName	Behavior
Visible	If disabled, the component will be hidden.	boolean	.visible	Common

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

See the **Vision - Comments Panel Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

- **Vision Component Customizers**

Examples

> **CAUTION**
>
> The following examples may need to be modifed to match the table and column names in your database. These examples are written for a MySQL database connection. If you are using a different database, some things may need to be changed. For example, using MS SQL Server requires:
>
> - The python value None may not be used when inserting into a byte array. NULL must be used in its place.
> - Binary data must be converted to a varbinary type when inserting. See the examples below

insertNote: using default table configuration

```
# Inserts a note using the three default tables: notes, users, and itemNotes.
# Also stores only the file name in the database instead of the full path to the
file.
# Assumes a User ID is used in the notes table.

# determine the ID for the logged in user
user = system.db.runScalarPrepQuery("SELECT id from users where username = ?",
[system.security.getUsername()])

# determine if a file is being attached
if filename is None:
    # a file was not attached, provide a blank for the bytes
    attachmentBytes = None
else:
    # get the bytes of the file at the path the user selects
    attachmentBytes = system.file.readFileAsBytes(filename)

    # splits the file name from the file path. This way we can show just the file
name on the component
    # Using '\' as a delimiter, but python requires 2 since it's an escape
character
    pathAndFile = filename.rsplit('\\', 1)
    filename = pathAndFile[1]

# build the query and the arguments
query = "INSERT INTO Notes (note, whoid, tstamp, attachment, filename, sticky)
VALUES (?, ?, CURRENT_TIMESTAMP, ?, ?, ?)"
arguments = [note, user, attachmentBytes, filename, sticky]
# insert the note
insertId = system.db.runPrepUpdate(query, arguments)

# insert a row onto the itemNotes table
# replace 'MYID' with the proper code to fetch your id
myId = 'MYID'
system.db.runPrepUpdate("INSERT INTO ItemNotes (AccountId, NoteId) VALUES (?, ?)",
[myId, insertId])
```

insertNote: using a single table

```
# Similar to the above example, but only a single database table is required.
# Assumes a User Name is used in the notes table.

# determine the name for the logged in user
user = system.security.getUsername()

# determine if a file is being attached
if filename is None:
    # a file was not attached, provide a blank for the bytes
    attachmentBytes = None

else:
    # get the bytes of the file at the path the user selects
    attachmentBytes = system.file.readFileAsBytes(filename)

    # splits the file name from the file path. This way we can show just the file
name on the component
    # Using '\' as a delimiter, but python requires 2 since it's an escape
character
    pathAndFile = filename.rsplit('\\', 1)
    filename = pathAndFile[1]

# insert the note. You may need to modify this query depending on how your
database handles binary data.


#MySQL query
query = "INSERT INTO Notes (note, whoid, tstamp, attachment, filename, sticky)
VALUES (?, ?, CURRENT_TIMESTAMP, ?, ?, ?)"

#MSSQL Server query
#We're converting the binary data into a VARBINARY datatype on this line, which is
the prefered datatype for binary data in MSSQL.
#query = "INSERT INTO Notes (note, whoid, tstamp, attachment, filename, sticky)
VALUES (?, ?, CURRENT_TIMESTAMP, CONVERT(VARBINARY(MAX),?), ?, ?)"

arguments = [note, user, attachmentBytes, filename, sticky]
system.db.runPrepUpdate(query, arguments)
```

Vision - Comments Panel Scripting Functions

This page details the various component and extension functions available for **Vision's Comments Panel component**.

Component Functions

This component does not have component functions associated with it.

Extension Functions

insertNote

- Description
  - Called when a note is added.
- Parameters
  - component **self** - A reference to the component that is invoking this function
  - string **note** - The text contents of the note
  - string **filename** - The full filepath to the the attachment
  - string **sticky** - A boolean indicating whether this note should be flagged as stickied
- Return
  - None

deleteNote

- Description
  - Called when a user clicks the 'delete' link on a note.
- Parameters
  - component **self** - A reference to the component that is invoking this function
  - integer **id** - The id of the note
- Return
  - None

unstickNote

- Description
  - Called when a user clicks the 'unstick' link on a note.
- Parameters
  - component **self** - A reference to the component that is invoking this function
  - integer **id** - The id of the note
- Return
  - None

downloadAttachment

- Description
  - Called when a user attempts to download an attachment from a note.
- Parameters
  - component **self** - A reference to the component that is invoking this function
  - integer **id** - The id of the note
- Return
  - None

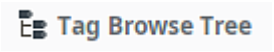
canDelete

- Description
  - Returns whether or not a note with the given id can be deleted. Notes that return True will show a 'delete' link.
- Parameters
  - component **self** - A reference to the component that is invoking this function
  - integer **id** - The id of the note
- Return
  - boolean - Notes with a True return can be deleted by the user, False return can not be deleted.

Vision - Tag Browse Tree



Component Palette Icon:



The Tag Browse Tree component is similar to the Tag Browser in the Designer, allowing tags to be browsed in both the Designer and the Client, and dragged on to other components like the Easy Chart. Unlike the Tag Browser, tags can not be edited, tag properties can not be displayed, and UDT definitions can not be displayed. Tags in the component can be refreshed through scripting by calling refresh().

Properties

Property	Description
Border	The border surrounding this component. Options are No border, Etched (Lowe (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), and Field Border. **Note**: The border is unaffected by rotation.
Font	Font of text on this component.
Include Historical Tags	Whether or not to display historical Tags.
Include Realtime Tags	Whether or not to display non-historical Tags.
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this co
Name	The name of this component.
Quality	The data quality code for any Tag bindings on this component.

Property	Description
Root Node Path	The path of the root of this tree structure, or "" if no selection. When intentiona node, the exact syntax changes depending on what the Tag Tree Mode proper • **Realtime Tag Tree**: `[TagProvider]FolderPath/` The example below is using the "default" Tag Provider, and a folder name `[default]machine_1/` • **Historical Tag Tree**: `[HistoricalProvider/GatewayName:TagProvider]` The example below applies to DB Table Historian Providers and will set th named "machine_1". It uses a historical provider named "DB", a Gateway "ignition", and a Tag Provider named "default": `[DB/ignition:default]machine_1/` The Historical Tag Tree mode also accepts the following formats that will only Historian Providers: `histprov:HistoricalProvider:/sys:GatewayName:/prov:TagProvider:/` `histprov:HistoricalProvider:/sys:GatewayName:/prov:TagProvider`
Selected Paths	Contains the paths that should be selected on the tree which should be in the string column.
Selection Mode	What kind of selection regions does the tree allow. Options are Single, Multiple Multiple - Discontiguous.
Show Root Handles	Whether or not to show handles next to parent nodes.
Show Root Node	Whether or not to show the root node of the tree.
Tag Tree Mode	Choose whether the tree is built using Tags from the default provider or the his
Visible	If disabled, the component will be hidden.

Deprecated Properties

Property	Description	Property Type	Scripting	Category
Data Quality	The data quality code for any Tag bindings on this component.	int	.dataQuality	Deprecated

Scripting

See the **Vision - Tag Browse Tree Scripting Functions** page for the full list of scripting functions available for this component.

Event Handlers

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

Customizers

- **Vision Component Customizers**

Example

Code Snippet

```
```

```
# The following code shows a right-click popup menu.
# Add these lines after the """ """ section of the createPopupMenu extension
function.
# Note how lines below are indented, the first def command should line up with the
# indentation of the """ """ section of the Extension Function.

    def showValue(self):
        value = str(clickedTag.value.value)
        system.vision.showMessage(value)

    def showLastChange(self):
        lastChange = str(clickedTag.value.timestamp)
        system.vision.showMessage(lastChange)

    itemsDict = {"Show Value": showValue, "Show Last Change": showLastChange}
    JPopupMenu = system.vision.createPopupMenu(itemsDict)
    return JPopupMenu
```

Vision - Tag Browse Tree Scripting Functions

This page details the various component and extension functions available for Vision's **Tag Browse Tree** component.

Component Functions

This component does not have component functions associated with it.

Extension Functions

filterTag

- Description
  - Called for each tag loaded into tag browse tree. Return false to hide this tag from the tree.

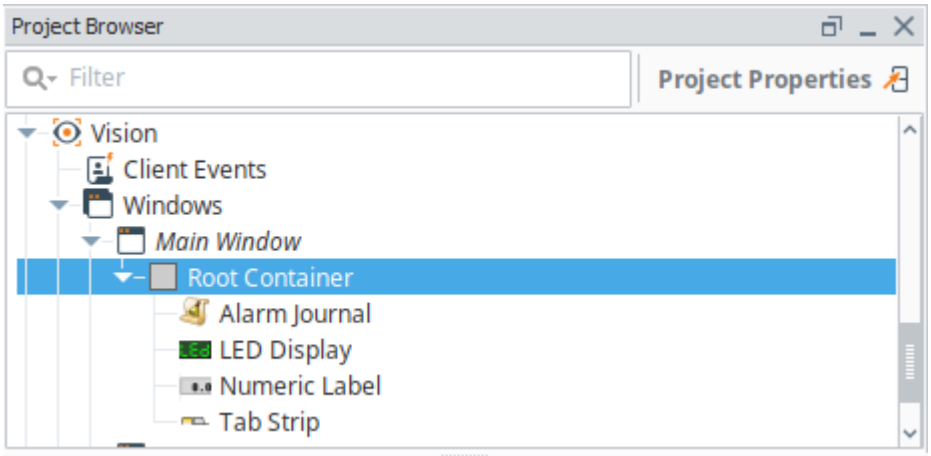
    Note that this is called for each **Tag**, not any folders that appear in the component.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - Tag Object **tag** - The tag itself.
- Return
  - Boolean

createPopupMenu

- Description
  - Returns a popup menu that will be displayed when the user triggers a popup menu (right click) on the tree. Use **system.vision.createPopupMenu** to create the popup menu.
- Parameters
  - Component **self** - A reference to the component that is invoking this function.
  - Tag Object **clickedTag** - The tag of the clicked on tree path.
  - List **selectedTags** - The tags of the selected paths of the tree.
- Return

- JPopupMenu

Vision - The Window Object



Window

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

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

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. • 0 - Always • 1 - Never • 2 - When Not Maximized	int	.borderDisplayPolicy	Behavior
Cache Policy	By default this property is set to `Auto`, which keeps a window in a memory cache for a while after it is closed, so that if it is opened again it will be quick. The window isn't "active" while it is closed: all of its bindings and scripts are shut down. Setting this property to `Never` causes a fresh copy of the window to be deserialized every time it is opened. This is a performance hit, but it also is a convenient way to "clear out" the values of the window from	int	.cachePolicy	Behavior

Name	Description	Property Type	Scripting	Category
	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. • 0 - Auto • 1 - Never • 2 - Always			
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	int	.dockIndex	Layout

Name	Description	Property Type	Scripting	Category
	is docked to), and higher numbers are closer to the center.			
Dock Position	Determines the position this window is docked to, or if it is floating. • 0 - Floating • 3 - West • 4 - South • 2 - East • 1 - North	int	.dockPosition	Layout
Layer	Sets the layer that this window is in. Default layer is 0, which is the bottom layer. Windows in higher layers will always be shown on top of windows in layers beneath them. A common strategy for using the layer property is to set Main Windows and Docked windows to 0, Popups to 1 and very important popups to 2.	int	.layer	Layout
Location	The starting location that this window will open up at. Only applicable to floating	Point	.startingLocation	Layout

Name	Description	Property Type	Scripting	Category
	windows that are not set to start maximized. This value will be overridden when an open window script specifies where to open.			
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

Name	Description	Property Type	Scripting	Category
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. • 0 - Always • 1 - Never • 2 - When Not Maximized	int	.titlebarDisplayPolicy	Appearance
Titlebar Font	The font of the window title in the titlebar.	Font	.titlebarFont	Appearance
Titlebar Height	The height of the window's titlebar.	int	.titlebarHeight	Appearance

Scripting

See the **Vision - The Window Object Scripting Functions page** for the full list of scripting functions available for this component.

Event Handlers

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

Examples

For examples of windows, please see the **Vision Windows** section.

Vision - The Window Object Scripting Functions

This page details the various component and extension functions available for **Vision's Window Object components**.

Component Functions

.getRootContainer

- 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**.

.getComponentForPath

- 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.

Extension Functions

This component does not have extension functions associated with it.