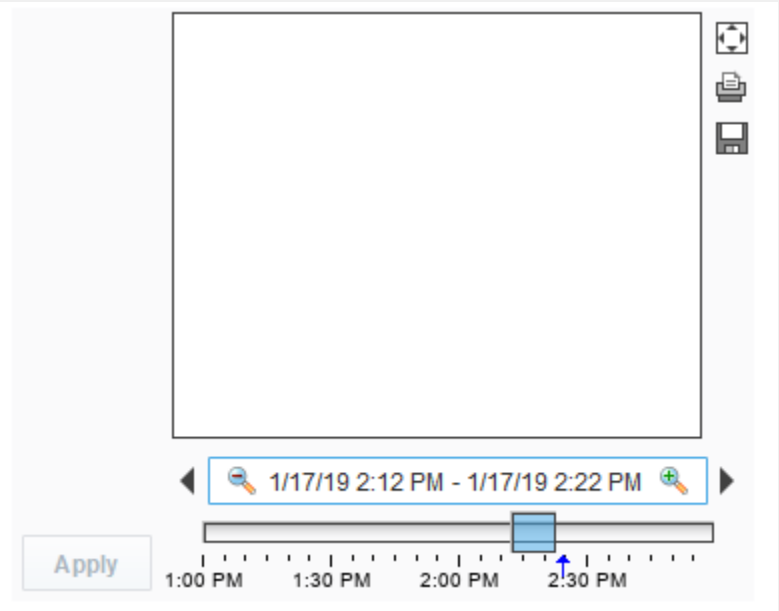
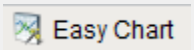
Vision - Easy Chart

General



Component Palette Icon:



Description

Description

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

Features

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

Pens

There are three kinds of pens in the Easy Chart:

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

Modes: Realtime vs Historical

The Easy Chart can operate in three different modes. These modes affect the range of data that is displayed, the controls the user is shown, and whether or not the chart polls for data.

1. Historical Mode. In this mode, the user is shown a Vision - Date Range component to pick the range of data to fetch and display. The initial values of this component are set through properties on the chart. In historical mode, the chart does not poll.
2. Realtime Mode. In this mode, the user is given the opportunity to pick the amount of time in the past to display. For example, the last 5 minutes or the last 2 hours. The chart will poll at a rate according to the Poll Rate parameter.
3. Manual Mode. In this mode, the chart will use the values if its Start Date and End Date parameters to govern what data is displayed. Polling is controlled by having the Poll Rate at zero (polling off) or greater than zero.

Basic Chart Configuration

The Easy Chart has many properties, like other components, that control its behavior. Things like its Mode, Polling Rate, etc are configured via the properties. All of the setup for adding pens, axes, subplots, and so forth is done through its Customizer. You can also drag and drop Historian-enabled tags onto the chart directly in the Designer to add those tags as chart pens. For an example, see Using the Vision - Easy Chart.

Y-Axes

The easy chart supports any number of Y-axes. To add an axis, go to the Axes tab of the chart customizer. When adding an axis, you get a number of options such as the type (numeric or logarithmic), label, color, autorange vs fixed range, and auto-ticks vs fixed ticks. You can also modify the position of the axis, but note that by default the Chart's Auto Axis Positioning property is enabled, which means that the chart will balance the axes automatically between left and right depending on demand. As pens are turned on and off by the user, only the axes that are used by visible pens are shown.

After you add your axes, you edit any pens that you want to use your new axes. Simply choose the new axis in the axis dropdown of the pen editing window.

Subplots

The Subplots feature lets you break up the chart's plot area into multiple distinct subplots that share the X axis, but have their own Y axes. This is often useful for digital data, as shown in the screenshot above. By default the chart has 1 subplot (the main plot). To add a new subplot, simply hit the add button in the Subplots tab of the chart customizer.

Subplots have relatively few options. The Weight option determines how much room the subplot gets relative to the other subplots. For example, in the screenshot above subplot #1's weight is 5, and subplot #2's weight is 1, leading to a 5-to-1 distribution of space. Just like axes, once you add your subplots you should go back to your pens and modify you pens' subplot property for any pens you want to appear on the subplot.

Pen Groups

You can put your pens in groups to break up the pens into some logical separation. For instance, in the screenshot above there are three pen groups: C1, C2, and Valves. The group name is used as the titled border for the pens' grouping container. Groups also have another purpose, but it is more advanced and most people won't have to worry about it. For more, read the Dynamic Pens section below.

Advanced Configuration

Dynamic Pens

In is often the case that you'll want to make one chart window that services many similar pieces of equipment. For instance, if you have 30 tanks and they all have the same datapoints, you want to be able to use one window for all 30 of them and simply pass the tank number into the chart window as a parameter. There are actually a number of ways to accomplish this, each method suitable for different scenarios.

Database pens have 2 ways to be made dynamic. The first is the Chart's Where Clause property. This is a snippet of SQL where clause syntax, like "machine_num = 28" that will be included for alldatabase pens in their queries. The second is to use a dynamic group. Any group can be made a dynamic group in the customizer. For each dynamic group, the easy chart will get a special dynamic property associated with that group. That property is another snippet of SQL where clause that will be applied to all database pens in that group.

The other way to make your pens (and anything else about the chart) dynamic at runtime is to use dynamic configuration. Read on...

Dynamic Configuration

The Easy Chart is not just meant to be easy to configure, but also very powerful. In particular, there is an emphasis on the ability to make any configuration change dynamically in a client - not just statically in the Designer. While a bit of scripting or clever property binding may be required, the technique is very powerful. This is achieved by storing all of the settings that you alter in the customizer in a set of expert-level dataset properties. So altering the datasets alters the chart configuration. You can inspect these various datasets, which hold the pens, axes, and subplot information, to see their format. They all look up information by column name (case-insensitive). So, if you have pen configuration stored in a database, you can bind an indirect SQL Query binding to alter the chart's pen set at runtime.

Properties

Name	Description	Property Type	Scripting	Category
3D X Offset	The offset to use in the x direction for the '3D Line' pen style.	int	.xOffset3D	Pen Style Options
3D Y Offset	The offset to use in the y direction for the '3D Line' pen style.	int	.yOffset3D	Pen Style Options
Allow Color Changes	If true, pen colors can be set to different values.	boolean	.allowColorChanges	Behavior
Allow Tag History Interpolation	If enabled and the query mode is not raw, the data will be interpolated for time spans with no data available.	boolean	tagHistoryAllowInterpolation	Tag History
Auto Apply	If true, user changes to pen visibility will occur immediately.	boolean	.autoApply	Behavior
Auto Axis Positioning	If true, axes alternate automatically between left and right, rather than being placed explicitly.	boolean	.autoPositionAxes	Behavior
Auto Color List	The list of colors to use if auto pen coloring is enabled.	Color[]	.autoColorList	Behavior
Auto Pen Coloring	If true, pens are assigned different colors automatically.	boolean	.autoColorPens	Behavior
Axes	This Dataset defines all axes that can be used by the pens.	Dataset	.axes	Chart Configuration
Axis Font	The font for axis labels.	Font	.axisLabelFont	Appearance
Background Color	The background color of the component. Se e Color Selector.	Color	.background	Appearance
Bar Margin	The margin to use for the 'Bar' pen style.	double	.barMargin	Pen Style Options
Border	The border surrounding this component. Options are: No border, Etched (Lowered), Etched (Raised), Bevel (Lowered), Bevel (Raised), Bevel (Double), Button Border, Field Border, Line Border, and Other Border. Note: The border is unaffected by rotation.	Border	.border	Appearance
Box Fill	For historical-mode date range. The fill color for the selection box. Can be chosen from color wheel, chosen from color palette, or entered as RGB or HSL value. See Color Selector.	Color	.boxFill	Historical Range
Button Size	The size of the utility button icons.	int	.utilityButtonSize	Utility Buttons
Bypass Tag History Cache	If true, tag history queries will not use the client history cache.	boolean	.tagHistoryBypassCache	Tag History
Calculated Pens	This Dataset defines the calculated pens for the chart.	Dataset	.calcPens	Chart Configuration
Chart Border	The border for the chart itself.	Border	.chartBorder	Appearance

Chart Mode	Affects the mode that the chart operates in; Manual Mode, Historical Mode, Realtime Mode. **Integer Value** / **Corresponding Mode** 0 – Manual 1 – Historical 2 – Realtime	int	.chartMode	Behavior
Chart Title	Sets an optional title to be displayed above the chart.	String	.title	Appearance
Cursor	The mouse cursor to use when hovering over this component. Options are: Default, Crosshair, Text, Wait, Hand, Move, SW Resize, or SE Resize.	int	.cursorCode	Common
DB Pens	This Dataset defines all of the database pens for the chart.	Dataset	.pens	Chart Configuration
Date Editor Background	The background color for the date editor. See Color Selector.	Color	.editorBackgroundColor	Appearance
Date Editor Foreground	The foreground color for the date editor. See Color Selector.	Color	.editorForegroundColor	Appearance
Date Range	Affects the position of the date range control.	int	.dateRangeLocation	Layout
Date Range Border	The border for the date range control, if visible.	Border	.dateRangeBorder	Appearance
Date Style	The style to display dates in. For international support.	int	.dateStyle	Historical Range
Digital Gap	The size of the gap to use between digital pens.	double	.digitalGap	Pen Style Options
Empty Group Name	The group name to use for pens that are not in a pen group.	String	.emptyGroupName	Behavior
End Date	For manual-mode. The end date to use for selecting pen data	Date	.endDate	Data
Font	Font of text on this component.	Font	.font	Appearance
Foreground Color	The foreground color of the component. See Color Selector.	Color	.foreground	Appearance
Gap Threshold	The relative threshold to use for determining continuity breaks for the 'Discontinous Line' pen style.	double	.gapThreshold	Pen Style Options
Gridline Color	The color of the gridlines. See Color Selector.	Color	.gridlineColor	Appearance
Gridline Dash Pattern	Enter a string of comma-delimited numbers which indicate the stroke pattern for a dashed line. For instance, "3,5" means three pixels on, five pixels off.	String	.gridlineDashPattern	Appearance
Gridline Width	The width (thickness) of the gridlines.	float	.gridlineWidth	Appearance
Group Pens	If true, pens will be grouped by their group name.	boolean	.penGrouping	Behavior

High Density Color	For historical-mode date range. The color used to indicate high data density. See Color Selector.	Color	.highDensityColor	Historical Range
Horiz Gap	The horizontal spacing to use for the pen checkboxes.	int	.hGap	Layout
Ignore Bad Quality Data	If true, causes the system to ignore any bad quality data.	boolean	tagHistoryIgnoreBadData	Tag History
Invert Time Axis	If true, the time axis values will increase from the right to left or from top to bottom depending on the Plot Orientation.	boolean	.invertTimeAxis	Layout
Legend	Where the legend should appear, if any.	int	.legend	Layout
Max Selection	For historical-mode date range. The maximum size of the selected date range.	String	.maxSelectionSize	Historical Range
Maximize Plot	If true, displays maximized plot.	boolean	.currentlyMaximized	Layout
Mouseover Text	The text that is displayed in the tooltip which pops up on mouseover of this component.	String	.toolTipText	Common
Name	The name of this component.	String	.name	Common
Outer Range End	For historical-mode date range. The end date for the outer range.	Date	.outerRangeEnd	Historical Range
Outer Range Start	For historical-mode date range. The start date for the outer range.	Date	.outerRangeStart	Historical Range
Pen Control Border	The border for the pen control panel, if visible.	Border	.penBorder	Appearance
Pen Control Mode	The style in which the pen control panel alters the chart configuration. In heavyweight mode, unchecked pens are not queried, so checking and unchecking pens refreshes the chart. In lightweight mode, all pens are constantly queried, so checking and unchecking pens is quick.	int	.penControlMode	Behavior
Pen Control?	Controls whether or not end-users can turn on and off pens.	boolean	.allowPenManipulation	Behavior
Plot Background	The background color for all plots, unless they override it. See Color Selector.	Color	.plotBackground	Appearance
Plot Orientation	The plot orientation for all plots.	int	.plotOrientation	Layout
Plot Outline	The color to use for the plot outline. See Color Selector.	Color	.plotOutlineColor	Appearance
Poll Rate	The rate (in milliseconds) at which this chart's queries poll. Historical charts don't use this property.	int	.pollRate	Behavior
Properties Loading	The number of properties currently being loaded. (Read only. Usable in bindings and scripting.)	int	.propertiesLoading	Uncategorized
Realtime Text	For realtime-mode date range. The text to display on the realtime date control.	String	.rtLabel	Realtime Range
Selected X Value	The selected domain axis value for X-Trace and Mark modes. (Read only. Usable in bindings and scripting.)	String	.selectedXValue	Uncategorized
Selection Highlight	For historical-mode date range. The focus highlight color for the selection box. See Color Selector.	Color	.selectionHighlight	Historical Range

Show Density	For historical-mode date range. If true, a data density histogram will be shown in the date range.	boolean	.showHistogram	Historical Range
Show Loading	If true, an animated indicator will be shown when data is loading.	boolean	.showLoading	Behavior
Show Maximize Button?	If true, a small maximize button will be displayed next to the chart.	boolean	.showMaximize	Utility Buttons
Show Popup?	If true, a popup menu will be shown on right-click that allows the user to change mode, print, save, etc.	boolean	.showPopup	Behavior
Show Print Button?	If true, a small print button will be displayed next to the chart..	boolean	.showPrint	Utility Buttons
Show Save Button?	If true, a small save button will be displayed next to the chart.	boolean	.showSave	Utility Buttons
Show Tooltips?	If true, tooltips showing point values will be displayed on the chart.	boolean	.tooltips	Behavior
Show Warnings	If true, warnings generated during chart configuration will be printed to the console.	boolean	.showWarnings	Behavior
Sort Pens	If true, pens visibility checkboxes will be sorted.	boolean	.alphabetizePens	Layout
Start Date	For manual-mode. The start date to use for selecting pen data.	Date	.startDate	Data
Startup Range	For historical-mode date range. If startup mode is Automatic, this will be the starting range of time available for selection.	String	.startupRange	Historical Range
Startup Selection	For historical-mode date range. If startup mode is Automatic, this will be the starting selected range.	String	.startupSelection	Historical Range
Subplot Gap	The gap between subplots.	double	.subplotGap	Layout
Subplots	This Dataset defines all subplots' relative size and color.	Dataset	.subplots	Chart Configuration
Tag History Resolution	The number of datapoints to request for tag history pens. -1 means raw data, 0 means automatic, which uses the width of the chart.	int	.tagHistoryResolution	Tag History
Tag History Resolution Mode	The mode used for the number of requested points. Fixed will use the Tag History Resolution Size, Natural will return a value per scanclass execution, Chart Width will be based on the actual width of the chart component, and Raw will be the raw data.	int	tagHistoryResolutionMode	Tag History
Tag Pens	This Dataset defines all of the Tag History pens for the chart.	Dataset	.tagPens	Chart Configuration
Tick Density	For historical-mode date range. This is multiplied by the width to determine the current ideal tick unit.	float	.tickDensity	Historical Range
Tick Font	The font for tick labels.	Font	.axisTickLabelFont	Appearance
Time Style	The style to display times of day. For international support.	int	.timeStyle	Historical Range
Title Font	The font for the optional chart title.	Font	.titleFont	Appearance
Today Color	For historical-mode date range. The color of the "Today Arrow" indicator. See Color Selector.	Color	.todayIndicatorColor	Historical Range

Total Datapoints	The number of datapoints being displayed by the graph. (Read only. Usable in bindings and scripting.)	int	.datapoints	Uncategorized
Track Margin	For historical-mode date range. The amount of room on either side of the slider track. May need adjusting of default font is changed.	int	.trackMargin	Historical Range
Unit	For realtime-mode date range. The selected unit of the realtime date control.	int	.unit	Realtime Range
Unit Count	For realtime-mode date range. The number of units back to display.	int	.unitCount	Realtime Range
Validate Scan Class Executions	Causes the tag history query to verify the scan class execution records, generating bad data for the time periods where the scanclasses did not execute.	boolean	tagHistoryValidateScanclass	Tag History
Vert Gap	The vertical spacing to use for the pen checkboxes.	int	.vGap	Layout
Visible	If disabled, the component will be hidden.	boolean	.visible	Common
Where Clause	A snippet of where clause that will be applied to all pens, like "TankNum = 2".	String	.globalWhereClause	Data
X Axis AutoRange?	If true, the X axis will automatically fit the range of available data, if false, it will display a fixed range based on the start date and end date.	boolean	.xAxisAutoRange	Behavior
X Axis Label	The label shown on the X Axis (time axis).	String	.xAxisLabel	Appearance
X Axis Margin	A margin for the upper and lower ends of the x axis, expressed as a percentage of the total range.	double	.xAxisMargin	Behavior
X Axis Visible	Should the x-axis be displayed?	boolean	.xAxisVisible	Appearance
X-Trace Large Number Format	The large decimal format for the x-trace value in the Easy Chart.	String	.xTraceLargeNumberFormat	Appearance
X-Trace Number Format Threshold	If the magnitude of the to-be-formatted value is below this threshold, then the X-Trace Small Number Format will be used.	double	.xTraceNumberFormatThreshold	Appearance
X-Trace Small Number Format	The small decimal format for the x-trace value in the Easy Chart.	String	.xTraceSmallNumberFormat	Appearance
X-Trace Track Mouse	If set enabled, and the chart is set to X-Trace mode, the X-Trace will auto track the mouse position while the cursor is over the component. This is particularly useful when displaying the Easy Chart on a touchscreen.	boolean	xTraceTrackMouse	Appearance

Scripting

Scripting Functions

exportExcel(filename)

- Description

  This function save the chart's datasets as an Excel file. Returns a String of the complete file path chosen by the user, or None if the user canceled the save.

- Parameters

  String filename - The default file name for the Save dialog.

- Return

  String

- Scope

  Client

print()

- Description

  This function will print the chart.

- Parameters

  Nothing

- Return

  Nothing

- Scope

  Client

setMode(mode)

- Description

  Sets the current mode for the chart.

- Parameters

  Int mode - The mode to set the chart to. The mode options are as follows:

  0 : Zoom Mode. This is the default mode, where the user can draw a zoom rectangle with the mouse pointer.

  1 : Pan Mode. This mode lets the user use the mouse pointer to pan the chart to the left and right.

  3 : Mark mode. This mode lets the user click near a datapoint to annotate the point with its X and Y value.

  4 : X-Trace mode. This mode lets the user click and drag on the chart to see all values that fall along that X value.

- Return

  Nothing

- Scope

  Client

exportDatasets()

- Description

  returns an Array List of datasets, representing the time series data of each type of pen.

- Parameters

  None

- Return

  Array List of datasets. Each dataset represents timeseries data for set of pens. The order of the datasets are listed below.

- Scope

  Client

- Index order of datasets

Index	Dataset
0	Tag Pens
1	Calculated Pens
2	Database Pens

Python - Accessing the Tag Pens Dataset

```
# This example will extract the Tag Pens series data that is
already present in an Easy Chart, and pass it to a Power Table on
the same window.
# This script could be placed on the Easy Chart's propertyChanged
event.

# Filter on the name of the property
if event.propertyName == 'tagPens':

 # Wrap our dataset behavor in a function, so we can pass it to
system.util.invokeLater
 def func():
  chart = event.source


  # Extract the datasets
  datasets = chart.exportDatasets()


  # Pass the first dataset (index 0 contains data for Tag Pens) to
the Power Table
  event.source.parent.getComponent('Power Table').data =
datasets[0]

 # Using invokeLater to provide a delay. We want this to run after
the chart has finished loading the new tag.
 system.util.invokeLater(func, 1000)
```

Extension Functions

configureChart

- Description

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

- Parameters

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

  JFreeChart chart - A JFreeChart object. Refer to the JFreeChart documentation for API details.

- Return

  Nothing

- Scope

  Client

getXTraceLabel

- Description

  Provides an opportunity to configure the x-trace label. Return a string to override the default label.

- Parameters

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

  JFreeChart chart - A JFreeChart object. Refer to the JFreeChart documentation for API details.

  String penName - The name of the pen the x-trace label applies to.

  int yValue - The y-value of the pen at the x-trace location.

- Return

  Nothing

- Scope

  Client

onPowerTableRowsDropped

- Description

  Called when the user has dropped rows from a power table on the chart. The source table must have dragging enabled.

- Parameters

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

  Component sourceTable - A reference to the table that the rows were dragged from.

  List rows - An array of the row indicies that were dragged, in the order they were selected.

  Dataset rowData - A dataset containing the rows that were dragged.

- Return

  Nothing

- Scope

  Client

onTagsDropped

- Description

  Called when the user has dropped tags from the tag tree onto the chart. Normally, the chart will add pens automatically when tags are dropped, but this default behavior will be suppressed if this extension function is implemented.

- Parameters

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

  List paths - A list of the tag paths that were dropped on the chart.

- Return

  Nothing

- Scope

  Client

Example - Pen Name Replacement

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

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

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

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

Example - Group Name

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

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

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

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

Event Handlers

mouse

mouseClicked

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

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

mouseEntered

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

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

mouseExited

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

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

mousePressed

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

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

mouseReleased

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

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

mouseMotion

mouseDragged

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

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

mouseMoved

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

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

propertyChange

propertyChange

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

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

Customizers

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

- Axes
- Subplots
- Pen Groups
- Pen Display
- Offsets
- Calculated Pens
- Ad-Hoc Charting
- Indirection

Examples

There are no examples associated with this component.