Date : 05-Feb-24

line chart - scatter plot version: 1.0.2

# **Overview**

Line Chart - Scatter Plot Chart (also known as Scatter Plot) is a component that creates a chart by plotting data points on a horizontal and a vertical axis, based on the data that you provide.

In the Scatter Plot Chart component, the intervals are defined on y-axis and the labels are defined on the x-axis.



## **Use case:**

### You can use the component in your mobile app to represent the comparison between distinct items or data in the form of a line graph. For examples, sales growth on monthly basis.

## **Features**

### Easily to update the chart with Iris properties or by API.

## **Percentage of re-use:**

80-90% (Data can be customizable and skins are not customized but can be changed manually)

# **Getting Started**

## **Prerequisites**

Before you start using the component, ensure you have the following:

* [HCL Foundry](https://manage.hclvoltmx.com/)
* Volt MX Iris

## **Platforms Supported**

### Mobile

#### iOS

#### Android

### Tablets

#### iOS

#### Android

### PWA

## **Importing the app**

## You can import the Forge components only into the apps that are of the Reference Architecture type.

## **To import the component, do the following:**

## Open your app project in Volt MX Iris.

1. In the Project Explorer, click the **Templates** tab.



1. Right-click **Components**, and then select **Import Component**. The **Import Component** dialog box appears.



1. Click **Browse** to navigate to the location of the component, select the component, and then click **Import**. The component and its associated widgets and modules are added to your project.

 

Once you have imported a component to your project, you can easily add the component to a form. For more information, refer [Add a Component to a Form](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_user_guide/Content/C_UsingComponents.html#add-a-component-to-a-form).

## **Building and previewing the app**

## Refer to below links:

### [**Build & Generate Native Apps**](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_user_guide/Content/Cloud_Build_in_VoltMX_Iris.html#cloud)

### [**Publish Apps to Enterprise App Store**](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_user_guide/Content/EAS.html#accessing)

# **References**

## **Dynamic Usage**

1. You can add a scatter plot Chart component dynamically. To do so,

In the **Project Explorer**, on the **Projects** tab, click **Controllers** section to access the respective **formController**. Create a method and implement the code snippet similar to the sample code mentioned below.

In the code snippet, you can edit the properties of the component as per your requirement. For more information, see Setting Properties.

 var scatterplot = new com.voltmxmp.scatterplot(

 {

 "autogrowMode": voltmx.flex.AUTOGROW\_NONE,

 "clipBounds": true,

 "height": "85%",

 "id": "scatterplot",

 "isVisible": true,

 "layoutType": voltmx.flex.FREE\_FORM,

 "left": "0%",

 "masterType": constants.MASTER\_TYPE\_USERWIDGET,

 "skin": "slFbox",

 "top": "5%",

 "width": "100%"

 }, {}, {});

 /\*Setting the component s properties\*/

 scatterplot.enableLegends = true;

 scatterplot.chartTitle = "Scatter Plot";

 scatterplot.enableGrid = true;

 scatterplot.lowValue= "0";

 scatterplot.highValue= "50";

 scatterplot.chartData =

 {

 "data":

 [

 {

 "dataPoint1": "12",

 "dataPoint2": "2",

 "dataPoint3": "1",

 "dataPoint4": "9",

 "dataPoint5": "3",

 "label": "d1"

 },

 {

 "dataPoint1": "9",

 "dataPoint2": "1",

 "dataPoint3": "3",

 "dataPoint4": "2",

 "dataPoint5": "9",

 "label": "d2"

 },

 {

 "dataPoint1": "7",

 "dataPoint2": "3.5",

 "dataPoint3": "4",

 "dataPoint4": "5",

 "dataPoint5": "2",

 "label": "d3"

 },

 {

 "dataPoint1": "8",

 "dataPoint2": "7",

 "dataPoint3": "5",

 "dataPoint4": "12",

 "dataPoint5": "7",

 "label": "d4"

 },

 {

 "dataPoint1": "5",

 "dataPoint2": "3",

 "dataPoint3": "6",

 "dataPoint4": "3",

 "dataPoint5": "5",

 "label": "d5"

 },

 {

 "dataPoint1": "14",

 "dataPoint2": "5",

 "dataPoint3": "9",

 "dataPoint4": "7",

 "dataPoint5": "2.5",

 "label": "d6"

 },

 {

 "dataPoint1": "10",

 "dataPoint2": "8",

 "dataPoint3": "12",

 "dataPoint4": "1",

 "dataPoint5": "6",

 "label": "d7"

 }

 ]

 };

 scatterplot.xAxisTitle = "data";

 scatterplot.titleFontColor = "#000000";

 scatterplot.plotDetails =

 {

 "data":

 [

 {"color": "#1B9ED9", "legendName": "blue"},

 {"color": "#76C044", "legendName": "green"},

 {"color": "#F26B29", "legendName": "orange"},

 {"color": "#7A54A3", "legendName": "purple"},

 {"color": "#FFC522", "legendName": "yellow"}

 ],

 };

 scatterplot.legendFontSize = "95%";

 scatterplot.enableGridAnimation = false;

 scatterplot.yAxisTitle = "value";

 scatterplot.lowValue = "0";

 scatterplot.legendFontColor = "#000000";

 scatterplot.titleFontSize = "12";

 scatterplot.highValue = "15";

 scatterplot.bgColor = "#FFFFFF";

 scatterplot.enableChartAnimation = true;

 scatterplot.enableStaticPreview = true;

 /\*Adding the component to the form\*/

 this.view.add(scatterplot);

1. **Save** the file.

## **Properties**

The properties provided on the **Component** tab allow you to customize the UI elements in the Scatter Plot Chart component. You can set the properties directly on the **Component** tab or dynamically through code. This section provides information on how to set the properties dynamically through code.

### **General**

**Background Color**

|  |  |
| --- | --- |
| **Description:** | Specifies background color of the chart. |
| **Syntax:** | bgColor |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this.view.<componentID>.bgColor= "#FFFFFF"; |

**Low Value**

|  |  |
| --- | --- |
| **Description:** | Specifies the starting value on the vertical (y) axis. The minimum value is the start index on y-axis. |
| **Syntax:** | lowValue |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks**: | Low and High values must be passed according to the data passed to the charts |
| **Example:** | this.view.<componentID>.lowValue= "0"; |

**High Value**

|  |  |
| --- | --- |
| **Description:** | Specifies the maximum value on vertical (y) axis. The maximum value is the end index on y-axis. |
| **Syntax:** | highValue |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks**: | Low and High values must be passed according to the data passed to the charts. |
| **Example:** | this.view.<componentID>.highValue= "30"; |

**Enable Chart Animation**

|  |  |
| --- | --- |
| **Description:** | Controls whether or not to enable the chart animation. |
| **Syntax:** | enableChartAnimation |
| **Type:** | Boolean |
| **Read/Write:** | Write |
| **Remarks:** | Disabling the chart animation will also disable the grid animation. |
| **Example:** | this.view.<componentID>.enableChartAnimation= true; |

**Chart Data**

|  |  |
| --- | --- |
| **Description:** | Enables a user to provide the data to generate the chart. |
| **Syntax:** | chartData |
| **Type:** | Data Grid |
| **Read/Write:** | Write |
| **Remarks:** | The property cannot be changed dynamically. |

**Details of Scatter Plots**

|  |  |
| --- | --- |
| **Description:** | Specifies the area colors for the respective regions on the chart and their legend names. |
| **Syntax:** | plotDetails |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks:** | The property cannot be changed dynamically. |

**Enable Rendering Chart**

|  |  |
| --- | --- |
| **Description:** | Controls whether or not to enable rendering the chart. |
| **Syntax:** | enableStaticPreview |
| **Type:** | Boolean |
| **Read/Write:** | Write |
| **Remarks:** | The default value of the property is **true**. |
| **Example:** | this.view.<componentID>.enableStaticPreview = true; |

### **Axis Titles**

**X axis Title**

|  |  |
| --- | --- |
| **Description:** | Specifies the text to be displayed as the X-axis (horizontal axis) title. |
| **Syntax:** | xAxisTitle |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this.view.<componentID>.xAxisTitle= "Day"; |

**Y axis Title**

|  |  |
| --- | --- |
| **Description:** | Specifies the text to be displayed as the Y-axis (vertical axis) title. |
| **Syntax:** | yAxisTitle |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this.view.<componentID>.yAxisTitle= "y-axis"; |

### **Grid**

**Enable Grid**

|  |  |
| --- | --- |
| **Description:** | Controls whether or not to enable the chart grid. |
| **Syntax:** | enableGrid |
| **Type:** | Boolean |
| **Read/Write:** | Write |
| **Example:** | this.view.<componentID>.enableGrid = true; |

**Enable Grid Animation**

|  |  |
| --- | --- |
| **Description:** | Controls whether or not to enable the grid animation. |
| **Syntax:** | enableGridAnimation |
| **Type:** | Boolean |
| **Read/Write:** | Write |
| **Remarks**: | The component disables the Grid animation if you disable the Chart animation. |
| **Example:** | this.view.<componentID>.enableGridAnimation = false; |

### **Title**

**Chart Title**

|  |  |
| --- | --- |
| **Description:** | Specifies the text to be displayed as the Chart title. |
| **Syntax:** | chartTitle |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this.view.<componentID>.chartTitle = "Scatter Plot Chart"; |

**Title Font Color**

|  |  |
| --- | --- |
| **Description:** | Specifies the font color of the Chart title. |
| **Syntax:** | titleFontColor |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks:** | The property expects a Hex color code preceded by the number sign (#). |
| **Example:** | this.view.<componentID>.titleFontColor = "#000000"; |

**Title Font Size**

|  |  |
| --- | --- |
| **Description:** | Specifies the font size of the Chart title. |
| **Syntax:** | titleFontSize |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this.view.<componentID>.titleFontSize= "12"; |

### **Legends**

**Enable Legends**

|  |  |
| --- | --- |
| **Description:** | Controls whether or not to enable the Legends. |
| **Syntax:** | enableLegends |
| **Type:** | Boolean |
| **Read/Write:** | Write |
| **Remarks:** | The default value of the property is **true**. |
| **Example:** | this.view.<componentID>.enableLegends = true; |

**Legend Font Color**

|  |  |
| --- | --- |
| **Description:** | Specifies the font color of the Chart legend. |
| **Syntax:** | legendFontColor |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks:** | The default value of the property is "#000000". |
| **Example:** | this.view.<componentID>.legendFontColor= "#000000"; |

**Legend Font Size**

|  |  |
| --- | --- |
| **Description:** | Specifies the font size of the Chart legend. |
| **Syntax:** | legendFontSize |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks:** | The default value of the property is "95%".The font size must be between 95% and 120% for better UI. |
| **Example:** | this.view.<componentID>.legendFontSize= "95%"; |

## **Events**

No Events

## **API’s**

### **createChart**

**Syntax**

createChart(data, plotDetails)

**Parameters**

*data:*
JSON array contains the data based on which the Scatter Plot Chart is generated. The JSON array should contain data points and the corresponding values to generate plots, in the key-value pair format. Here is the JSON array format:

var chartData = [{“label”: col1 , “dataPoint1”: “val1”, “dataPoint2”: “val3”},

 {“label”: “col2” , “dataPoint1”: “val2”, “dataPoint2”: “val4”}];

In the above format, **label** and **dataPointi** are keys and they are case sensitive.

* **label:** The key accepts values for the label names on the Horizontal (x) axis. You can specify upto Four characters as a row name. For example, "col1". Specifying more than four characters results distortion in the component UI.
* **dataPointi:** The key accepts the values corresponding to the label name on the Vertical (y) axis.

*plotDetails:*
JSON array contains the data based on which the colors are assigned to the respective data points. Here is the JSON array format:

var plotDetails = [{“legendName”: “blue” , “color”: “#1B9ED9” },

 {“legendName”: “green” , “color”: “#76C044” } ];

In the above format, **legendName** and **color** keys are case sensitive.

* **legendName:** The key accepts value for the legend names. The key accepts string values, so define the value within the quotation marks.
* **color:** The key accepts color code values.

The component can conveniently handle a maximum of 07 key-value pairs in the JSON array. Defining more than seven key-value pairs results distortion in the component UI.

**Return Value**

None

**Example**

var chartData = [{“label”: col1 , “dataPoint1”: “val1”, “dataPoint2”: “val3”},

 {“label”: “col2” , “dataPoint1”: “val2”, “dataPoint2”: “val4”}];

var plotDetails = [{“legendName”: “blue” , “color”: “#1B9ED9” },

 {“legendName”: “green” , “color”: “#76C044” } ];

this.view.<componentID>.chartData = {data: chartData};

this.view.<componentID>.plotDetails = {data: plotDetails};

this.view.<componentID>.createChart(chartData, plotDetails);

**Note**:

* The createChart API must be invoked to reflect the dynamic change of any property after adding the chart to a form either by drag and drop method or dynamic creation.
* The data must be passed in a proper format (without any missing values).
* The number of color codes which are passed must be equal to the number of plots.
* Never pass an empty JSON for colors.

# **Revision History**

App version 1.0.2:

## **Known Issues**

• You cannot invoke createChart() API on forms preShow and postShow

* Cannot handle layout properties of the component as per the device orientation. You must handle the properties at the form level.
* If two or more plots have the same value the later plots cover the previous plots, as a result only the latest plot is visible.

## **Limitations**

* The label names on the vertical and horizontal axis must not contain more than **three** characters. More number of characters leads to overlap of characters.
* The maximum length of the data that can be passed to the chart is **seven**. The number of points in the JSON array should be between **one** and **five**. Each plot should have the number of points between **two** and **seven**. Exceeding the limit leads to distortions.