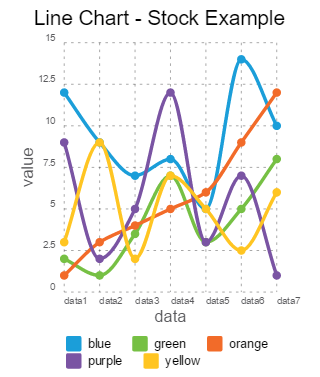
Date : 05-Feb-24

Line Chart - Multi Series version: 1.1.2

# **Overview**

Line Chart - Multi Series (also known as Multi-Line Chart) is a component that creates a Multi-Line Chart, based on the data that you provide.

In the Multi-Line 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.

Graphical user interface, text, application

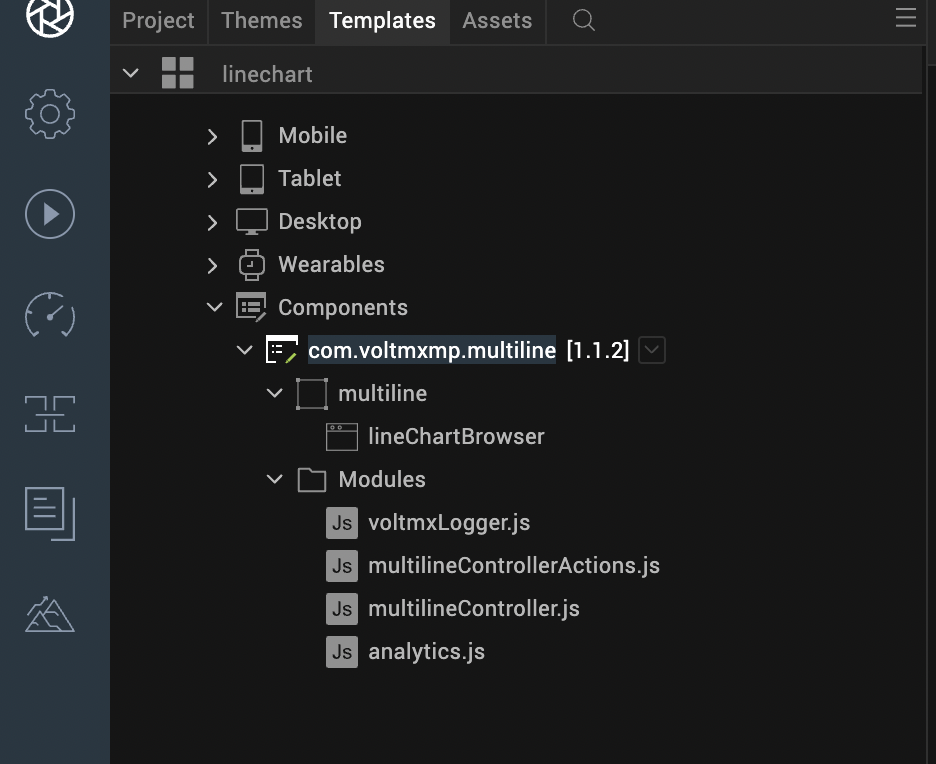
Description automatically generated

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

Graphical user interface, text, application, Teams

Description automatically generated

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 Multi-Line 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 like 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 multiline = new com.voltmxmp.multiline(

{

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

"clipBounds": true,

"height": "90%",

"id": "multiline",

"isVisible": true,

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

"left": "0%",

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

"skin": "slFbox",

"top": "10%",

"width": "100%"

}, {}, {});

/\* Setting the component s properties \*/

multiline.chartTitle = "Multi Line";

multiline.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"

}]

};

multiline.enableGrid = true;

multiline.enableLegends = true;

multiline.xAxisTitle = "data";

multiline.lineDetails =

{

"data":

[

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

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

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

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

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

],

};

multiline.yAxisTitle = "value";

multiline.enableStaticPreview = true;

multiline.legendFontSize = "95%";

multiline.enableGridAnimation = false;

multiline.titleFontSize = "12";

multiline.lowValue = "0";

multiline.titleFontColor = "#000000";

multiline.legendFontColor = "#000000";

multiline.highValue = "15";

multiline.bgColor = "#FFFFFF";

multiline.enableChartAnimation = true;

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

this.view.add(multiline);

1. **Save** the file.

## **Properties**

### **General**

1. **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. |

1. **Line Details**

|  |  |
| --- | --- |
| **Description:** | Specifies the color of the lines generated on the chart. |
| **Syntax**: | lineDetails |
| **Type:** | DataGrid |
| **Read/Write:** | Write |
| **Remarks:** | The property cannot be changed dynamically. |

1. **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"; |

1. **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"; |

1. [**Background Color**](javascript:void(0);)

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

1. [**Enable Chart Animation**](javascript:void(0);)

|  |  |
| --- | --- |
| **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; |

1. [**Enable Chart with Static Data**](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Specifies whether or not the chart should render in the data grids, with the given data. |
| Syntax: | enableStaticPreview |
| **Type:** | Boolean |
| **Read/Write:** | Write |
| **Example:** | this.view.<componentID>.enableStaticPreview = true; |

### **Axis Titles**

* + - 1. **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"; |

* + - 1. **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= "Rate"; |

### **Grid**

* + - 1. **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; |

* + - 1. **Enable Grid Animation**

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

### **Title**

* + - 1. **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 = "Multi Line Chart"; |

* + - 1. **Title Font Color**

|  |  |
| --- | --- |
| **Description:** | Specifies the font color of the Chart title. |
| Syntax: | titleFontColor |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this.view.<componentID>.titleFontColor = "#000000"; |

* + - 1. [**Title Font Size**](javascript:void(0);)

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

### **Legends**

* + - 1. **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; |

* + - 1. **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"; |

* + - 1. **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%". |
| **Example:** | this.view.<componentID>.legendFontSize= "95%"; |

## **Events**

No Events

## **API’s**

### **createChart**

The API creates a Multi-Line Chart.

**Syntax**

createChart(data, lineDetails)

**Parameters**

*data:*  
JSON array contains the data based on which the Area Chart is generated. The JSON array should contain data of column names and the corresponding values to generate lines, in the key-value pair format. Here is the JSON array format:

var chartData =

[{

"dataPoint1": "12",

"dataPoint2": "2",

"label": "d1"

},

{

"dataPoint1": "9",

"dataPoint2": "1",

"label": "d2"

}];

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 up to Four characters as a row name. For example, "col1". Specifying more than 04 characters results distortion in the component UI. The maximum limit for the number of lines is five.
* **dataPointi:** The key accepts the values corresponding to the label names.

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

var lineDetails =

[

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

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

];

In the above format, **legendName** key is case sensitive.

* **legendName:** The key accepts values for the color of the lines. The line1 and line 2 consume values for the line1 and line 2 respectively. The maximum limit for the number of lines is five.
* **color:** The key accepts color code values.

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

**Return Value**

Boolean (returns **True** if the chart is successfully generated).

**Example**

var chartData =

[{

"dataPoint1": "12",

"dataPoint2": "2",

"label": "d1"

},

{

"dataPoint1": "9",

"dataPoint2": "1",

"label": "d2"

}];

var lineDetails =

[

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

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

];

this.view.multiline.chartData = {data: chartData};

this.view.multiline.lineDetails = {data: lineDetails};

this.view.multiline.createChart(chartData,lineDetails);

**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 lines.
* Never pass an empty JSON for colors.

# **Revision History**

App version 1.1.2

## **Known Issues**

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

## **Limitations**

N/A