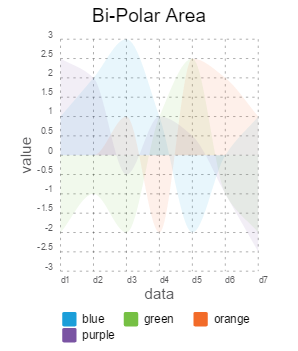
Date :  05-Feb-24

BIPOLAR Area Chart

version: 1.1.2

# **Overview**

Bipolar Area Chart is a Volt MX Iris component that creates a Bipolar Area chart, based on the data that you provide. You can use the component in your mobile app to represent the comparison between distinct items or data in the form of area chart. For examples, sales growth on monthly basis.

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


## **Use case:**

### Consider a scenario that you want to provide the stock market information in a use case in your application. In the app, you want to build a feature to represent variation in the stock values of companies on monthly basis in the form of an area. You can use the Bipolar Area Chart component to represent the variations of stock values in the form of area chart.

### You can also use the Bipolar Area Chart component to represent the variations of monthly department incomes.

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

85%-90% (Data can be customizable and skins are customized and also can be changed manually).

# **Getting Started**

## **Prerequisites**

Before you start using the Bipolar Area Chart component, ensure you have the following:

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

## **Platforms Supported**

### Mobile

#### iOS

#### Android

### Tablets

### PWA

## **Importing the app**

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

**To import the Bipolar Area Chart component, do the following:**

1. Open your app project in Volt MX Iris.
2. In the Project Explorer, click the **Templates** tab.

Graphical user interface, text, application

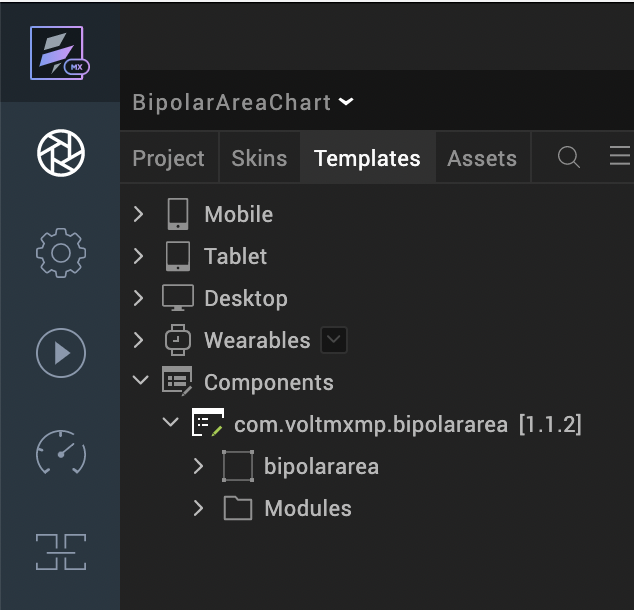
Description automatically generated

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).

After adding a component to a form, you can configure the component the way you want it using the **Look**, **Skin**, and **Action** tabs on the **Properties** pane. Configuring the properties on the **Properties** pane is similar to configuring the properties of any widget in VoltMX Iris.

You can also see that a new tab, **Component**, is added on the **Properties** pane. The **Component** tab contains assorted properties relevant to the component that allow you to customize the component as required. The properties on the **Component** tab are categorized as **General**, **Axis Titles**, **Grid** and **Title** properties. The **General** properties are the default properties of individual widgets in the component.

# **References**

## **Dynamic Usage**

You can also add a Bipolar Area Up Chart component dynamically. To do so:

1. 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.

/\* Creating the component s object \*/

var bipolararea = new com.voltmxmp.bipolararea(

{

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

"clipBounds": true,

"height": "70%",

"id": "bipolararea",

"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 \*/

bipolararea.chartTitle = "Bi-Polar";

bipolararea.enableGrid = true;

bipolararea.enableLegends = false;

bipolararea.chartData =

{

"data":

[

{

"dataPoint1": "1",

"dataPoint2": "-2",

"dataPoint3": "0",

"dataPoint4": "2.5",

"dataPoint5": "",

"label": "d1"

},

{

"dataPoint1": "2",

"dataPoint2": "-1",

"dataPoint3": "0",

"dataPoint4": "2",

"dataPoint5": "",

"label": "d2"

},

{

"dataPoint1": "3",

"dataPoint2": "-2",

"dataPoint3": "1",

"dataPoint4": "-0.5",

"dataPoint5": "",

"label": "d3"

},

{

"dataPoint1": "1",

"dataPoint2": "1",

"dataPoint3": "-2",

"dataPoint4": "1",

"dataPoint5": "1",

"label": "d4"

},

{

"dataPoint1": "-2",

"dataPoint2": "2.5",

"dataPoint3": "2.5",

"dataPoint4": "0.5",

"dataPoint5": "",

"label": "d5"

},

{

"dataPoint1": "0",

"dataPoint2": "-1",

"dataPoint3": "2",

"dataPoint4": "-1",

"dataPoint5": "",

"label": "d6"

},

{

"dataPoint1": "1",

"dataPoint2": "-2",

"dataPoint3": "1",

"dataPoint4": "-2.5",

"dataPoint5": "",

"label": "d7"

}

]

};

bipolararea.xAxisTitle = "data";

bipolararea.legendFontSize = "95%";

bipolararea.areaDetails =

{

"data":

[

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

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

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

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

],

};

bipolararea.titleFontColor = "#000000";

bipolararea.enableGridAnimation = false;

bipolararea.yAxisTitle = "value";

bipolararea.legendFontColor = "#000000";

bipolararea.titleFontSize = "12";

bipolararea.lowValue = "-10";

bipolararea.highValue = "10";

bipolararea.bgColor = "#FFFFFF";

bipolararea.enableChartAnimation = true;

bipolararea.enableStaticPreview = true;

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

this.view.add(bipolararea);

In the code snippet, you can edit the properties of the component as per your requirement.

1. **Save** the file.

## **Properties**

The properties provided on the **Component** tab allow you to customize the UI elements in the Bipolar Area 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**

**Details for Areas**

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

**Background Color**

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

**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. |

**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= "-5"; |

**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= "5"; |

**Enable Chart with Static Data**

|  |  |
| --- | --- |
| **Description:** | Controls whether or not to enable the Static data in 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 |
| **Example:** | this.view.componentID.enableGridAnimation= true; |

### **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 = Bi-Polar Area Chart ; |

**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 ; |

**Title Font Color**

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

### **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**

-- None of the events are exposed.

## **API’s**

The following API pertains to the Bipolar Area Chart component.

**createChart**

The API creates a Bipolar Area Chart.

**Syntax**

createChart(data)

**Parameters**

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

var data = [{label:"col1", dataPoint1:"val1", dataPoint2: "val3"},

{label:"col2", dataPoint1:"val2", dataPoint2: "val4"}, ..];

In the above format, **label** and **dataPoint1** 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, "Jan". 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.

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

var areaDetails = [{legendName: "blue" , color: "#1B9ED9" }, {legendName: "green" , color: "#76C044" } ..];

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 = [{lblName: "Jan", dataVal: "12"},

{lblName: "Feb", dataVal: "5"},

{lblName: "Mar", dataVal: "8"}];

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

this.view.componentID.createChart(chartData);

# **Revision History**

App version 1.1.2:

## **Known Issues**

Following are the known issues in the Bipolar Area Chart component:

* On Android, when adding the component dynamically, the layout of the component does not show up as expected.
* Cannot handle layout properties of the component as per the device orientation. You must handle the properties at the form level.

## **Limitations**

Following are the limitations in the Bipolar Area Chart component:

* The label names on the horizontal axis and vertical axis should not be more than 3 to 4 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.