Date: 05-Feb-24

vertical bar chart – distributed series

version: 1.1.2

# **Overview**

Vertical Bar Chart – distributed series is a component that creates a chart or graph with vertical bars, based on the data that you provide.

In the Vertical Bar Chart – distributed series 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 vertical bars. 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.

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

Graphical user interface, text, application

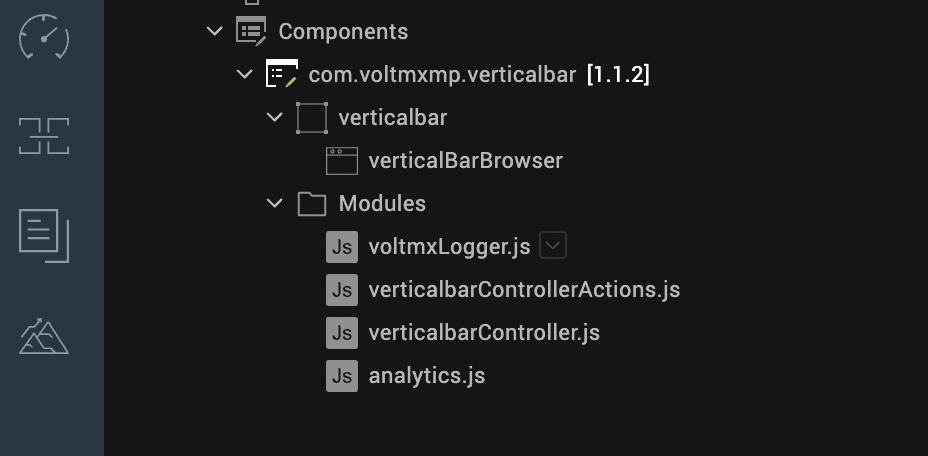
Description automatically generated

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

Graphical user interface, text, application, Teams

Description automatically generated

4.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 Vertical Bar Chart – distributed series 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 verticalbar = new com.voltmxmp.verticalbar(

{

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

"clipBounds": true,

"id": "verticalbar",

"height": "90%",

"width": "100%",

"top": "10%",

"left": "0%",

"isVisible": true,

"skin": "slFbox",

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

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

"zIndex": 1

}, {}, {});

/\* Setting the component s properties \*/

verticalbar.chartTitle = "Vertical Bar Chart";

verticalbar.bgColor = "#FFFFFF";

verticalbar.lowValue = "0";

verticalbar.highValue = "30";

verticalbar.chartData =

{

"data":

[

{"colorCode": "#1B9ED9", "label": "Data1", "value": "25"},

{"colorCode": "#76C044", "label": "Data2", "value": "20"},

{"colorCode": "#E8672B", "label": "Data3", "value": "10"},

{"colorCode": "#464648", "label": "Data4", "value": "30"},

{"colorCode": "#FFC522", "label": "Data5", "value": "40"}

]

};

verticalbar.enableStaticPreview = true;

verticalbar.titleFontColor = "#000000";

verticalbar.titleFontSize = "12";

verticalbar.xAxisTitle = "x-axis";

verticalbar.yAxisTitle = "y-axis";

verticalbar.enableGrid = true;

verticalbar.enableChartAnimation = true;

verticalbar.enableGridAnimation = true;

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

this.view.add(verticalbar);

1. **Save** the file.

## **Properties**

### **General**

1. **Background Color**

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

1. **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; |

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:** | Providing a negative value converts the bar chart into a bipolar chart. Low and High values must be passed according to the data passed to the chart. |
| **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 chart. |
| **Example:** | this.view.<componentID>.highValue= "30"; |

1. **Enable Chart with Static Data**

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

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 is static and cannot be assigned dynamically. |

### **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 = "Vertical Bar Chart"; |

1. **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"; |

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"; |

## **Events**

No Events

## **API’s**

### **createChart**

The API creates a vertical bar chart.

**Syntax**

createChart(data)

**Parameters**

*data:*  
JSON array contains the data based on which the vertical bar chart is generated. The JSON array should contain data of column names and the corresponding values to generate bars for the columns, in the key-value pair format. Here is the JSON array format:

var data =

[

  {"colorCode": "#1B9ED9", "label": "Data1", "value": "25"},

  {"colorCode": "#76C044", "label": "Data2", "value": "20"}

];

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

* **label:** The key accepts string values, so define the value within the qoutation marks. You can specify upto 04 characters as a column name. For example, "Jan". Specifying more than 04 characters results distortion in the component UI.
* **value:** The key accepts integer values.
* **colorCode:** The key accepts color codes. The value of the key must be a 6-character Hex color code preceded by the number sign (#).

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

**Return Value**

None

**Example**

var chartData =

[

  {"colorCode": "#1B9ED9", "label": "Data1", "value": "25"},

  {"colorCode": "#76C044", "label": "Data2", "value": "20"}

];

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

this.view.verticalbar.createChart(chartData);

**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).
* If you populate the data dynamically through the createChart(data) API and also using the data grid, the chart generates based on the data passed dynamically through the createChart(data) API.

# **Revision History**

App version 1.1.2:

## **Known Issues**

Following are the known issues in the component:

* Cannot handle layout properties of the component as per the device orientation. You must handle the properties at the form level.
* You cannot invoke createChart() API on forms preShow and postShow.

## **Limitations**

The following are the limitations with the component:

* The label names on the horizontal axis and vertical axis must not contain more than **three** characters. Providing more characters will cause an overlap.
* The maximum length of the data that can be passed to the chart is **seven**. Exceeding the limit leads to UI distortions.