Date :  06-Feb-24

Vertical Bar Chart - stacked bars

version: 1.0.2

# **Overview**

Vertical Bar Chart – Stacked Bars (also known as Stacked Vertical Bar Chart )is a Volt MX Iris component that creates a chart or graph with vertical bars stacked on each other 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 vertical bars. For examples, sales growth on monthly basis.

In the Vertical Bar Chart – Stacked Bars 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 chart. You can use the Stacked Vertical Bar Chart component to represent the variations of stock values in the form of chart with vertical bars.

### You can also use the Stacked Vertical Bar Chart component to represent the variations of yearly revenue 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 Stacked Vertical Bar Chart 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 Stacked Vertical Bar Chart component, do the following:**

1. Open your app project in Volt MX Iris.
2. 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).

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 Volt MX 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.

**Note:**

* Changes made to the properties are reflected only during run time, but not during design time.

# **References**

## **Dynamic Usage**

You can also add a Stacked Vertical Bar 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 similar to the sample code mentioned below.

/\* Creating the component s object \*/

var stackverticalbar = new com.voltmxmp.stackverticalbar({

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

 "clipBounds": true,

 "height": "100%",

 "id": "stackverticalbar",

 "isVisible": true,

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

 "left": "0%",

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

 "skin": "CopyslFbox1",

 "top": "0Dp",

 "width": "100%"

 }, {}, {});

/\* Setting the component s properties \*/

stackverticalbar.chartData =

{ "data": [

 {

 "label": "d1Code",

 "stackValue1": "5",

 "stackValue2": "2",

 "stackValue3": "1",

 "stackValue4": "3",

 "stackValue5": "1"

 },

 {

 "label": "d2",

 "stackValue1": "1",

 "stackValue2": "4",

 "stackValue3": "2",

 "stackValue4": "3",

 "stackValue5": "2"

 },

 {

 "label": "d3",

 "stackValue1": "4",

 "stackValue2": "5",

 "stackValue3": "4",

 "stackValue4": "2",

 "stackValue5": "3"

 },

 {

 "label": "d4",

 "stackValue1": "3",

 "stackValue2": "3",

 "stackValue3": "6",

 "stackValue4": "1",

 "stackValue5": "4"

 }]

 };

stackverticalbar.stackDetails =

{

 "data": [

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

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

 {"color": "#f2b54b", "legendName": "updatedYellow"},

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

 {"color": "#987654", "legendName": "blabla"}

 ]

 };

 stackverticalbar.enableLegends = true;

 stackverticalbar.enableGrid = true;

 stackverticalbar.chartTitle = "Stack Vertical Bars";

 stackverticalbar.xAxisTitle = "data";

 stackverticalbar.legendFontSize = "95%";

 stackverticalbar.titleFontSize = "12";

 stackverticalbar.yAxisTitle = "value";

 stackverticalbar.enableGridAnimation = false;

 stackverticalbar.legendFontColor = "#000000";

 stackverticalbar.titleFontColor = "#000000";

 stackverticalbar.lowValue = "0";

 stackverticalbar.highValue = "30";

 stackverticalbar.bgColor = "#FFFFFF";

 stackverticalbar.enableChartAnimation = true;

 stackverticalbar.enableStaticPreview = true;

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

this.view.add(stackverticalbar);

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 Stacked Vertical Bar Chart component. You can set the properties directly on the **Component** tab or by writing a JavaScript. This section provides information on how to set the properties by writing JavaScript.

**1. 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**: | 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. **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. **Details of Stacks in Bars**

|  |  |
| --- | --- |
| **Description:** | Enables a user to provide color codes for the stacks in the bars and their respective legend names. |
| **Syntax**: | stackDetails |
| **Type:** | Data Grid |
| **Read/Write:** | Write |
| **Remarks:** | The property cannot be changed dynamically. |

1. **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; |

**2. 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"; |

**3. 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= false; |

**4. 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 Stacked Bars"; |

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"; |

**5. 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%".The font size must be between 95% and 120% for better UI. |
| **Example:** | this.view.componentID.legendFontSize= "95%"; |

**Note:**In few scenarios, the Legends may behave in a weird manner.

## **Events**

 -- None of the events are exposed.

## **API’s**

The following API pertains to the Stacked Vertical Bar Chart component.

******createChart**

The API creates a Stacked Vertical Bar Chart.

**Syntax**

createChart(data, stackDetails)

**Parameters**

*data:*
JSON array contains the data based on which the Stacked 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 =

[ {"label": "lbl1", "stackValue1": "val1", "stackValue2": "val2"},

 {"label": "lbl2", "stackValue1": "val3", "stackValue2": "val4"},

];

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

* **label:** The key accepts string values, so define the value within the quotation marks. You can specify up to 04 characters as a column name. For example, "lbl1". Specifying more than 04 characters results distortion in the component UI.
* **stackValuei:** The key accepts integer values.

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

var stackDetails =

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

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

];

In the above format, **legendName**, **color** key is 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 07 key-value pairs results distortion in the component UI.

The component supports up to 5 stack values.

**Return Value**

None

**Example**

var chartData =

[

 {

 "label": "d1Code",

 "stackValue1": "5",

 "stackValue2": "2",

 "stackValue3": "1",

 "stackValue4": "3",

 "stackValue5": "1"

 },

 {

 "label": "d2",

 "stackValue1": "1",

 "stackValue2": "4",

 "stackValue3": "2",

 "stackValue4": "3",

 "stackValue5": "2"

 },

 {

 "label": "d3",

 "stackValue1": "4",

 "stackValue2": "5",

 "stackValue3": "4",

 "stackValue4": "2",

 "stackValue5": "3"

 },

 {

 "label": "d4",

 "stackValue1": "3",

 "stackValue2": "3",

 "stackValue3": "6",

 "stackValue4": "1",

 "stackValue5": "4"

 }];

var stackDetails = [

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

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

 {"color": "#f2b54b", "legendName": "updatedYellow"},

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

 {"color": "#987654", "legendName": "blabla"}

 ];

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

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

 this.view.componentID.createChart(chartData, stackDetails);

**Note:**

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

# **Revision History**

App version 1.0.2

## **Known Issues**

Following are the limitations in the Stacked Vertical Bar Chart 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 in the axis names.
* The maximum length of the data that can be passed to the chart is **seven**. Exceeding the limit leads to UI distortions.

## **Limitations**

Following are the known issues in the Stacked Vertical Bar 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.