Date : 01-Feb-2024

Single Slider

VERSION: 1.0.1

# **Overview**

Sliders have been around for a long time and have become the de facto standard for selecting and filtering apps. This component helps you achieve that by letting you visually select a value within a larger data set. This component incorporates the latest material design UX principles and cool animations that gives an enhanced User experience for the users.

The Single Slider component contains a slider and a handle. The component allows users to move the handle to select a specific value from a given range.

## **Use case:**

## Users can use the Single Slider component to control volume, to zoom in and out of an image, and so on.

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

 80-90%.

## **Features:**

1. Slider with a single handle to select a specific value from a given range.
2. Easy to achieve the single slider feature without developing it from scratch.
3. Ability to define intervals on the slider.
4. Facility to customize the UI as you require.

# **Getting Started**

## **Prerequisites**

Before you start using the Single Slider component, ensure the following:

• [HCL Foundry](https://manage.hclvoltmx.com/)

• Volt MX Iris

## **Platforms Supported**

### Mobile

#### iOS

#### Android

## **Importing the app**

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

##  **To import the Single Slider 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

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

## Graphical user interface, text, application, Teams  Description automatically generated

## 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.

## A screenshot of a computer  Description automatically generated

## 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.

## **Building and previewing the app**

## After performing all the above steps, you can build your app and run it on your device. For more information, you can refer to the Building and Viewing an Application section of the Volt MX User Guide.

## You can then run your app to see the Single Slider work in real time.

# **References**

## **Dynamic Usage**

You can also add **Single Slider** component dynamically. To do so,

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

/\* creating a component's Object \*/

 var slider = new com.voltmxmp.singleslider({

 "clipBounds": true,

 "height": "15%",

 "id": "slider",

 "isVisible": true,

 "left": "0dp",

 "top": "0dp",

 "width": "100%",

 "zIndex": 1

 }, {}, {});

/\* Setting the component's properties \*/

 slider.minValue=5;

 slider.maxValue=15;

 slider.intervalDisplay = true;

 slider.graduations = 10;

 slider.divisions = 10;

 slider.endIndex = 10;

/\*Adding the Single Slider component to a Form\*/

 this.view.add(slider);

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

2. Save the file.

## **Properties**

The properties provided on the **Component** tab allow you to customize the elements in the **Single Slider** component. These elements can be UI elements, service parameters, and so on. You can set the properties from the Volt MX Iris Properties panel on the right-hand side. You can also configure these properties using JavaScript code.

**General Properties**

**1. Minimum Value**

|  |  |
| --- | --- |
| **Description:** | Specifies the minimum value of the slider. The minimum value is the start index of the slider. |
| **Syntax:** | minValue |
| Type: | Number |
| **Read/Write:** | Read + Write |
| **Example:** | this. view.ComponentID.minValue=5; |

**2. Maximum Value**

|  |  |
| --- | --- |
| **Description:** | Specifies the maximum value of the slider. The maximum value is the end index of the slider. |
| **Syntax:** | maxValue |
| Type: | Number |
| **Read/Write:** | Read + Write |
| **Example:** | this. view.ComponentID.maxValue=15; |

**3. Display Interval**

|  |  |
| --- | --- |
| **Description:** | Controls whether the interval labels are visible on the slider. |
| **Syntax:** | intervalDisplay |
| **Type:** | Boolean |
| **Read/Write:** | Write |
| **Example:** | this. view. componentID.intervalDisplay = true; |
| **Remarks:** | The default value of the property is true. |

**4. Steps**

|  |  |
| --- | --- |
| **Description:** | Specifies the number of intervals on the slider. |
| **Syntax:** | graduations |
| **Type:** | Number |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. componentID.graduations = 10; |
| **Remarks:** | The values of interval points on the slider are determined based on the specified minimum and maximum values of the slider. The values are displayed in the pop-up when you move the handle on the slider. |

**5****. Interval Count**

|  |  |
| --- | --- |
| **Description:** | Specifies the number of interval labels to be generated and displayed on the slider. |
| **Syntax:** | divisions |
| **Type:** | Constant |
| **Read/Write:** | Write |
| **Example:** | this. view. componentID.divisions = 10; |

**6. Default Selected Value**

|  |  |
| --- | --- |
| **Description:** | Specifies which value on the slider must be selected by default. |
| **Syntax:** | defaultIndex |
| **Type:** | Number |
| **Read/Write:** | Write |
| **Example:** | this. view. componentID.defaultIndex = 10; |

**Image Options**

**7. Slider Source**

|  |  |
| --- | --- |
| **Description:** | Specifies the file name of the image to be displayed as the slider icon. |
| **Syntax:** | srcSlider |
| **Type:** | Image |
| **Read/Write:** | Read + Write |
| **Example:** |  this. view. componentID. sliderSource= “singleslider\_point.png”; |
| **Remarks:** | * Do not add any uppercase characters in the file name of the image.
* Ensure that the image is in PNG format.
* Specify the file name of the image along with the extension
 |

**8. Pin Source**

|  |  |
| --- | --- |
| **Description:** | Specifies the file name of the image to be displayed as the pin indicator. |
| **Syntax:** | srcPin |
| **Type:** | Image |
| **Read/Write:** | Read + Write |
| **Example:** |  this.view.componentID.pinSource=“singleslider\_currentpin.png”; |
| **Remarks:**  | * Do not add any uppercase characters in the file name of the image.
* Ensure that the image is in PNG format.
* Specify the file name of the image along with the extension
 |

**9. Bullet Icon Source**

|  |  |
| --- | --- |
| **Description:** | Specifies the file name of the image to be displayed as bullet icons. |
| **Syntax:** | srcBullet |
| **Type:** | Image |
| **Read/Write:** | Read + Write |
| **Example:** |  this. view. componentID. srcLeftBullet= “point\_blue.png”;this. view. componentID. srcRightBullet= “point\_grey.png”; |
| **Remarks:** | * Do not add any uppercase characters in the file name of the image.
* Ensure that the image is in PNG format.
* Specify the file name of the image along with the extension
 |

**Skin Section**

**10. Selected range Skin**

|  |  |
| --- | --- |
| **Description:** | Specifies the skin for the line which is displayed between the initial position and the slider pin. |
| **Syntax:** | sknRange |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. componentID.sknRange= “voltmxmplblSkinRange”; |
| **Remarks:** | Before you set the property, ensure that the skin ID that you specify already exists in your app project. |

**11. Active Index Skin**

|  |  |
| --- | --- |
| **Description:** | Specifies the skin for the number displayed in the selected region. |
| **Syntax:** | sknActiveIndex |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** |  this.view.componentID.sknActiveIndex=“voltmxmpnumberBlue”; |
| **Remarks:** | Before you set the property, ensure that the skin ID that you specify already exists in your app project. |

**12. Inactive Index Skin**

|  |  |
| --- | --- |
| **Description:** | Specifies the skin for the number which is not in the range of the single slider. |
| **Syntax:** | sknInactiveIndex |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** |  this.view.componentID.sknInActiveIndex= “voltmxmpnumberLineSkn”; |
| **Remarks:** | Before you set the property, ensure that the skin ID that you specify already exists in your app project. |

**13. Selected Index Skin**

|  |  |
| --- | --- |
| **Description:** | Specifies the skin that is applied to the header main text. |
| **Syntax:** | sknSelectedIndex |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** |  this.view.componentID.sknSelectedIndex=“CopysliderNumber”; |
| **Remarks:** | Before you set the property, ensure that the skin ID that you specify already exists in your app project. |

**14. Background Skin**

|  |  |
| --- | --- |
| **Description:** | Specifies the skin for the background of the single slider component. |
| **Syntax:** | sknBackground |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** |  this.view.componentID.sknBackground= “voltmxmpflxback”; |
| **Remarks:** | Before you set the property, ensure that the skin ID that you specify already exists in your app project. |

## **APIs**

## The following APIs pertain to the Single Slider component.

 **1. getSelectedValues**

**Description:** The API fetches the selected values.

**Syntax:** getSelectedValues()

**Parameters:** None

**Return Value:** *Integer*
Returns the selected value on the Single Slider.

**Example:**

var selectedValues= this.view.componentID.getSelectedValues();

var index = selectedValues;

## **D. Events**

You can define events to be executed when an action is performed. You can configure the events directly on the Actions tab or by writing a JavaScript. To configure the events on the Actions tab, click Edit against each event. For more information, refer to Add Actions.

This section provides details about each event that help you define the actions by writing a JavaScript.

**1. onChangeEnd**

**Description:** The event is invoked when the user-touch is released from the

 Slider.

**Syntax:** onChangeEnd(values)

**Remarks:** The event returns the index value of the slider where the user-touch is released at the time of event is invoked.

**2. onValueChange**

|  |  |
| --- | --- |
| **Description:** | The event is invoked when the index is changed from one point to another. |
| **Syntax:** | onValueChange(values) |
| **Remarks:** | The event returns the changed index value. |

# **Revision History**

App version 1.0.1

## **Limitations**

* The numbers overlap if the number of divisions are many.
* The component renders as expected only during the run time.