Date: 06-Apr-23

INTERSTITIAL SCREEN (RIPPLES)

VERSION: 1.0.2

1. **OVERVIEW**

The Interstitial screen (Ripples) can be used to fill natural transition points in the flow of an app. E.g., when new content is being loaded or whenever a major change in state occurs that a clean break is required. It is primarily a loading screen whenever a delay is encountered, you can use this screen to keep the user engaged. For instance, use it to tell the user that his/her request has been received and the app is working on it.

**A. Use case:**

## If a form takes more time to load, you can add the Interstitial Screen (Ripples) component to notify the user that the form is in the process of loading. The Interstitial Screen disappears when the form loads.

## When new content is being loaded or whenever a major change in state occurs that a clean break is required. It is primarily a loading screen.

###  **B. Features**

1. Ready-to-use the component.
2. Keep user engaged while content is being loaded- use the Interstitial screen (Ripples) to tell the user what is being loaded and how brief the delay will be.
3. Easy to customize.

## **C. Percentage of re-use**

80-90% (Data can be customizable)

# **Getting Started**

## **Prerequisites**

Before you start using the Interstitial Screen (Ripples) component, ensure you have the following:

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

## **Platforms Supported**

### Mobile

* 1. iOS
	2. Android

## **Importing the Interstitial Screen (Ripples) Component**

Before you start importing the component to Volt MX Iris, you must download the component from the HCL Volt MX Iris website.

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

**To import the Interstitial Screen (Ripples) component, do the following:**

1. Open your app project in Volt MX Iris.
2. On the **File** menu, point to **Import**, and click **Component**. The **Open** dialog appears.
3. Navigate to the location where you downloaded the component (zip file) on your computer, select the component, and click **Open**. The **Import Forge Item** dialog appears.
4. In the **Library Name** box, type a name if you want to create a new library. Otherwise, you can select the existing library name.
5. In the **Collection Name** box, type a name if you want to create a new collection. Otherwise, you can select the existing collection name.
6. In the **Component Name** box, the **com.voltmxmp.ripples** name is displayed by default.

When you import a component, the component is imported to Iris workspace, but not directly into your app. Thus, after you import a component, you must add the component to your app.

Adding an Interstitial Screen Component to your App:

**To add an Interstitial Screen (Ripples) component, do the following:**

1. Open your app, and then open the form to which you want to add the component.
2. On the **My Libraries** tab, select the defined Forge library from the drop-down list in which the component exists.
3. From the **Collections** sub-tab, drag the component onto the form on the Iris canvas. The component is added to the form. You can also see a new element, **com.voltmxmp.ripples** in the **Components** section on the **Templates** tab.

**Note:**Ensure that the component is set to occupy 100 percent width of the device. You can achieve this by setting the width of the parent widget to 100 percent.

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** and **Text** properties.

**Note:**Changes made to the **General** properties are reflected only during run time, but not during design time (Iris Canvas).

The component can be used only in the **Portrait** mode.

## **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](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_user_guide/Content/Cloud_Build_in_VoltMX_Iris.html#cloud) section of the Iris User Guide.

You can then run your app to see this component work in real time.

# **3. References**

## **A. Dynamic Usage**

You can also add an Interstitial Screen component dynamically. To do so,

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

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

var ripplescreen = new com.voltmxmp.ripplescreen(

 {

 "clipBounds": true,

 "height": "100%",

 "id": "ripplescreen",

 "isVisible": true,

 "left": "0dp",

 "top": "0dp",

 "width": "100%",

 "zIndex": 1

 }, {}, {});

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

 ripplescreen.backgroundSrc = "voltmxmp\_ripple\_bg.png";

 ripplescreen.title = "INTERSTITIAL SCREEN TITLE";

 ripplescreen.description = "Subtitle short description";

 ripplescreen.pulsationSrc = "voltmxmp\_ripple\_circle.png";

 ripplescreen.objIsVisible = true;

 ripplescreen.imgObjectSrc = "voltmxmp\_ripple\_icon.png";

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

 this.view.add(ripplescreen);

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

1. Save the file.

**B. Properties**

The properties provided on the **Component** tab allow you to customize the UI elements in the Interstitial Screen component. You can set the properties directly on the **Component** tab or by writing a JavaScript. This section provides information on how to set properties by writing a JavaScript.

* **General**

**1. Background Image**

|  |  |
| --- | --- |
| **Description:** | Specifies the file name of the image to be set as the background for the interstitial screen. |
| **Syntax:** | backgroundSrc |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks:** | * Ensure that the image file exists in the *workspace\resources\common* directory.
* 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.
 |
| **Example:** | this.view.componentID.backgroundSrc = "voltmxmp\_bouncing\_bg.png"; |

* **Text**

### **2. Title**

|  |  |
| --- | --- |
| **Description:** | Specifies the title to be displayed on the Interstitial Screen. |
| **Syntax:** | title |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.title = "INTERSTITIAL SCREEN TITLE"; |

### **3. Description**

|  |  |
| --- | --- |
| **Description:** | Specifies the description to be displayed on the interstitial screen. |
| **Syntax:** | description |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.description= "Subtitle short description"; |

### **4. Pulsation Image**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |

|  |  |
| --- | --- |
| **Description:** | Specifies the file name of the image to be displayed as ripples around the icon (magnifier). |
| **Syntax:** | pulsationSrc |
| **Type:** | String (Hex Color code) |
| **Read/Write:** | Read + Write |
| **Remarks:** | Before setting the property:* Ensure that the image file exists in the *workspace\resources\common* directory.
* 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.
 |
| **Example:** | this.view.componentID.pulsationSrc = voltmxp\_ripple\_circle.png |

 |

### **Object (Magnifier)**

### **5. Image**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|

|  |  |
| --- | --- |
| **Description:** | Specifies the file name of the image to be set as the icon in the center of the ripple (magnifier). |
| **Syntax:** | imgObjectSrc |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks:** | Before setting the property:* Ensure that the image file exists in the workspace\resources\common directory.
* 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.
 |
| **Example:** | this.view.componentID.imgObjectSrc = "voltmxmp\_ripple\_icon.png"; |

 |  |
|  |  |

###  **6. Visibility**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|

|  |  |
| --- | --- |
| **Description:** | Controls whether or not to display the icon in the center of the ripple (magnifier). |
| **Syntax:** | objIsVisible |
| **Type:** | Boolean |
| **Read/Write:** | Read + Write |
| **Remarks:** | The default value of the property is **true**. |
| **Example:** | this.view.componentID.objIsVisible = true; |

 |  |

### **Skin**

You can select skins from the **Exposed Skins** drop-down list on the **Skin** tab. This section provides information on how to set Skin by writing a JavaScript.

**7. Title**

|  |  |
| --- | --- |
| **Description:** | Specifies the skin of the title. |
| **Syntax:** | titleSkin |
| **Read/Write:** | Read + Write |
| **Remarks:** | Before you set the property, ensure that the skin ID that you specify already exists in your app project. |
| **Example:** | this.view.componentID.titleSkin= "Skin Name" |

**8. Description**

|  |  |
| --- | --- |
| **Description:** | Specifies the skin of the Description. |
| **Syntax:** | descriptionSkin |
| **Read/Write:** | Read + Write |
| **Remarks:** | Before you set the property, ensure that the skin ID that you specify already exists in your app project. |
| **Example:** | this.view.componentID.descriptionSkin= "Skin Name “; |

**C. Events**

 -- None of the events are exposed.

**D. APIS**

 The following API pertains to the Interstitial Screen component:

**1. show**

|  |  |
| --- | --- |
| **Description:** | The API displays the Interstitial Screen. |
| **Syntax:** | show() |
| **parameters:** | None |
| **Return Value:** | None |
| **Example:** | this.view.componentID.show(); |

#### **2. hide**

|  |  |
| --- | --- |
| **Description:** | The API hides the Interstitial Screen. |
| **Syntax:** | hide() |
| **parameters:** | None |
| **Return Value:** | None |
| **Example:** | this.view.componentID.hide(); |

# **4. Revision History**

 App version 1.0.2

## **A. Limitations**

* The component does not support landscape mode.