Date: 28-Mar-23

**Refresh ANIMATION (Travel)**

VERSION: 1.0.2

# **Overview**

The Refresh Animation (Travel) component is a user defined animated UI. The component can be used in Travel related applications. The refresh animation is used to fill the space when a widget is pulled down to refresh the contents in the form.

## **Use case:**

Consider a case that you want to develop a Travel app. In this app, you can add the Refresh Animation (Travel) component to fill the space when a list (or page) is pulled down to update the travel fares for the respective destination. Using the APIs, you can hide the Refresh Animation (Travel) component when the list refresh is complete.

The Refresh Animation (Travel) component is a ready-to-use component. You can import the Refresh Animation (Travel) component into your app (created in Volt MX Iris), and achieve the Refresh Animation (Travel) feature without developing it from scratch. The Refresh Animation (Travel) component also facilitates a set of properties and APIs that helps you customize the UI elements and the functionality.

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

80-90%.

# **Getting Started**

## **Prerequisites**

Before you start using the Refresh Animation (Travel) 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 Refresh Animation (Travel)** **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.

## 

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" \l "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 Refresh Animation (Travel) work in real time.

# **References**

## **Dynamic Usage**

You can also add **Refresh Animation (Travel)** 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 travelrefresh = new com.voltmxmp.travelrefresh({

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

"clipBounds": true,

"height": "50%",

"id": "travelrefresh",

"isVisible": true,

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

"left": "0%",

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

"skin": "sknFlx4785f4",

"top": "0%",

"width": "100%",

"zIndex": 1

},{},{});   
/\* Setting component's properties \*/

travelrefresh.sunImageSrc = "voltmxmp\_travelrefresh\_sun.png";

travelrefresh.planeImageSrc = "voltmxmp\_travelrefresh\_plane.png";

travelrefresh.bigCloudImageSrc = "voltmxmp\_travelrefresh\_clouds.png";

travelrefresh.skyImageSrc = "voltmxmp\_travelrefresh\_sky.png";

travelrefresh.backgroundSkin = "slFbox";

travelrefresh.planeDuration = "5";

travelrefresh.sunDuration = "5";

travelrefresh.skyDuration = "4";

travelrefresh.bigCloudDuration = "5";  
/\*Adding the Component to the Form\*/

this.view.add(travelrefresh);

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 allows you to customize the elements in the **Refresh Animation (travel)** 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 a JavaScript code.

**General Properties**

## **IMAGE SOURCE**

**1. Sun Image Source**

|  |  |
| --- | --- |
| **Description:** | Specifies the file name of the image to be displayed as the sun during the refresh animation. |
| **Syntax:** | sunImageSrc |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. travelrefresh. sunImageSrc = "voltmxmp\_travelrefresh\_sun.png"; |
| **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. |

**2. Plane Image Source**

|  |  |
| --- | --- |
| **Description:** | Specifies the file name of the image to be displayed as the plane during the refresh animation. |
| **Syntax:** | planeImageSrc |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. travelrefresh. planeImagesrc = "voltmxmp\_travelrefresh\_plane.png"; |
| **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. |

**3. Sky Image Source**

|  |  |
| --- | --- |
| **Description:** | Specifies the file name of the image to be displayed as the sky during the refresh animation. |
| **Syntax:** | skyImageSrc |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. travelrefresh. skyImageSrc = "voltmxmp\_travelrefresh\_sky.png"; |
| **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. |

**4. BigCloud Image Source**

|  |  |
| --- | --- |
| **Description:** | Specifies the file name of the image to be displayed as the big cloud during the refresh animation. |
| **Syntax:** | bigCloudImageSrc |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. travelrefresh.bigCloudImageSrc = "voltmxmp\_travelrefresh\_clouds.png"; |
| **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. |

## **B. Animation Duration**

**5****. Plane Duration**

|  |  |
| --- | --- |
| **Description:** | Specifies the duration of the plane animation. The duration value is specified in seconds. |
| **Syntax:** | planeDuration |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this. view. travelrefresh. planeDuration = “5”; |

**6. BigCloud Duration**

|  |  |
| --- | --- |
| **Description:** | Specifies the duration of the big cloud animation. The duration value is specified in seconds. |
| **Syntax:** | bigCloudDuration |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this. view. travelrefresh. bigCloudDuration= “5”; |

**7. Sun Duration**

|  |  |
| --- | --- |
| **Description:** | Specifies the duration of the sun animation. The duration value is specified in seconds. |
| **Syntax:** | sunDuration |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this. view. travelrefresh. sunDuration= “5”; |
| **Remarks:** | The property throws an exception if you provide an invalid value. You must handle the exception. |

**8. Sky Duration**

|  |  |
| --- | --- |
| **Description:** | Specifies the duration of the sky animation. The duration value is specified in seconds. |
| **Syntax:** | skuDuration |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this. view. travelrefresh. skyDuration = “4”; |
| **Remarks:** | The property throws an exception if you provide an invalid value. You must handle the exception. |

**C. 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.

**9. Background Skin**

|  |  |
| --- | --- |
| **Description:** | Specifies the skin to be set as the background of the component. |
| **Syntax:** | backgroundSkin |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this. view. travelrefresh.backgroundSkin= “skinid”; |
| **Remarks:** | Before you set the property, ensure that the skin ID that you specify already exists in your app project |

## **D. APIs**

The following APIs pertain to the Refresh Animation (Travel) component.

### **i. show**

|  |  |
| --- | --- |
| **Description:** | The API displays the refresh animation. |
| **Syntax:** | show() |
| **Parameter:** | None |
| **ReturnValue:** | None |
| **Example:** | this. view. travelrefresh.show(); |

### **ii. hide**

|  |  |
| --- | --- |
| **Description:** | The API hides the refresh animation. |
| **Syntax:** | hide() |
| **Parameter:** | None |
| **ReturnValue:** | None |
| **Example:** | this. view. travelrefresh.hide(); |

### **iii. isShown**

|  |  |
| --- | --- |
| **Description:** | The API notifies whether the refresh animation is displayed or not. |
| **Syntax:** | isShown() |
| **Parameter:** | None |
| **ReturnValue:** | Boolean |
| **Example:** | Var returnValue = this.view.travelrefresh.isShown() |

**Sample Usage of the Component**

1. **Component with the Segment Widget**

Consider a use case where user pulls down a list to update the list data. You can use the Refresh Animation (Travel) component to fill the space when the list (segment widget) is pulled down to refresh the data. You can achieve the use case by following the below procedure.

**To use the component in the Segment Widget, do the following:**

1. Import the Refresh Animation (Travel) component.
2. Add a Flex Container Widget to the form.
3. Set the **Layout Type** of the FlexContainer Widget to **Flow Vertical**.
4. Drag the Refresh Animation (Travel) component and the Segment Widget onto the FlexContainer Widget.
5. Set the left and top margin values of the Refresh Animation (Travel) component to Zero percent.
6. In the Project Explorer, on the Projects tab, click the context menu arrow of Controllers under the respective channel, and then click FormController. The FormController JavaScript file opens in the Code Editor.
7. Copy and paste the following code snippet in the Action editor:

define({

  onNavigate: function(){

     this.view.seg1.scrollingEvents ={

       onPull: this.PostShow.bind(this)

     };

  },

  PostShow : function(){

    this.view.travelrefresh.setVisibility(true);

    this.view.travelrefresh.show();

    this.view.seg1.top = "23.9%";

    this.view.flx1.forceLayout();

    voltmx.timer.schedule("timer4",this.hide,5, false);

  },

  hide:function(){

    this.view.travelrefresh.setVisibility(false);

    this.view.seg1.top = "0%";

    this.view.flx1.forceLayout();

  }

});

The above code displays the Refresh Animation (Travel) component on triggering the onPull event of the Segment Widget.

8. Save the file

### **Component with the Flex Container Widget**

To use the component in the FlexContainer Widget, do the following:

1. Import the Refresh Animation (Travel) component.
2. Drag the Refresh Animation (Travel) component and the FlexContainer Widget onto the Form.
3. Set the left and top margin values of the Refresh Animation (Travel) component to Zero percent.
4. Set the left and top margin values of the FlexContainer Widget to Zero percent.
5. Define the onTouchStart, onTouchMove, and onTouchEnd events of the Form as shown in following code snippet:

/\*Call the methods in the forms Action editor\*/  
/\*onTouchStart of the Form \*/

this.onStart(y);  
/\*onTouchMove of the Form \*/

this.onMove(y);  
/\*onTouchEnd of the Form \*/

this.onEnd();

1. In the Project Explorer, on the Projects tab, click the context menu arrow of Controllers under the respective channel, and then click FormController. The FormController JavaScript file opens in the Code Editor.
2. Copy and paste the following code snippet in the FormController JavaScript file:

/\* Define methods in the Form Controller \*/

define({

yCoordinates: "0",

\_top: "0",

\_started: 0,

onStart: function (yy) {

this.yCoordinates = yy;

if (this.\_started === 0) {

this.view.componentInstance.show();

this.\_started = 1;

}

},

onMove: function (y) {

this.\_top = y - this.yCoordinates;

if (this.\_top > 0) {

this.view.flexContainer.top = this.\_top;

} else {

this.view.flexContainer.top =   
parseInt(this.view. componentInstance.height) + this.\_top + "dp";

}

},

onEnd: function () {

if (this.\_top >= 0) {

this.view.flexContainer.top = this.view. componentInstance.height;

} else {

this.view.flexContainer.top = "0dp";

this.view.componentInstance.hide();

this.\_started = 0;

}

}

});

The above code displays the Refresh Animation (Travel) component on triggering the Form events.

1. Save the file.

# **Revision History**

App version 1.0.2