Date : 23-Feb-22

FEED BACK PROMPT (2.0.2)

1. OVER VIEW

Feedback Prompt component is a custom widget, enables you to display a prompt that allow users to provide their feedback in the form of messages. The component comes with a sliding animation feature which provides a better UX for an app. The Feedback Prompt component contains following pages:

* **Rating:** Contains the logo and a message which prompts a user to provide the feedback.
* **Feedback:** Contains a text area where users can provide their reviews.
* **Thank You:** Contains a Thank you message. The Thank You section is displayed when a user provides a review.

## **Use case**

Consider a case that you are developing a food delivery app, which is used to order food from nearby restaurants. In the app, you can use the Feedback Prompt component that allows users to provide feedback about the food quality and delivery service.

The Feedback Prompt component is a ready-to-use component. You can import the Feedback Prompt component into your app (created in Volt mx Iris), and achieve the features of Feedback Prompt. The Feedback Prompt component also facilitates a set of properties and APIs that helps you customize the fields and the functionality.

## **Features**

### Prompts customers to provide feedback

### Sliding animation

### Facilitates customer to provide feedback by giving a message

### Displays thank you message after receiving the feedback.

### Opportunity to improve customer experience based on the received feedback.

## **Percentage of re-use**

70-80% (Data and skins are customized and can be changed manually)

# Getting Started

## Prerequisites

Before you start using the Feedback Prompt component, ensure you have the following:

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

## Platforms Supported

### Mobile

#### iOS

#### Android

### Tablets

### PWA

## Importing the Component

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

## **To import the Feedback Prompt component, do the following:**

## Open your app project in Volt MX Iris.

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

Graphical user interface, text, application

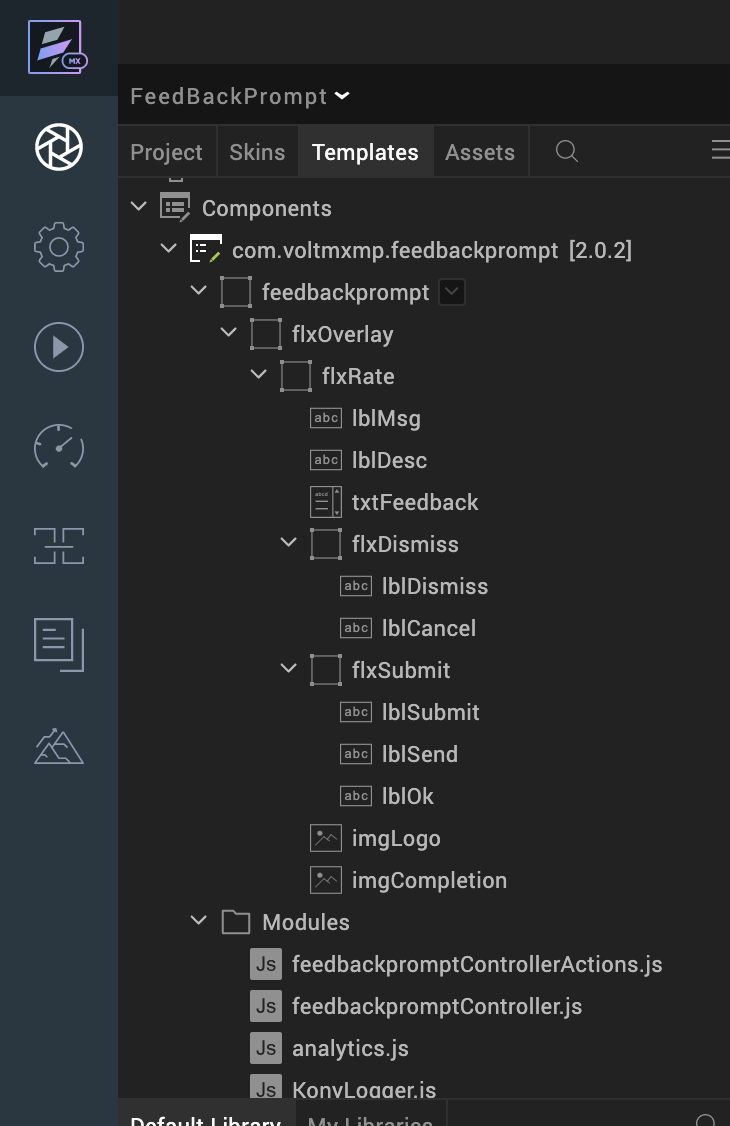
Description automatically generated

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

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 Volt MX Iris User Guide.

You can then run your app to see the Feedback Prompt work in real time.

# References

## Dynamic Usage

If you want to use the Feedback Prompt component dynamically, you will need to import the component into your project Templates. Follow the given steps to do so

## Download the component from HCL VoltMX Marketplace as a zip file.

## Go to the Templates tab in your project explorer.

## Right click on Components and select Import Component.

## Navigate to where you downloaded your zip file and import it into Iris.

After you import the component into your project templates, you can add it to your app dynamically. To do so, follow the given steps

## Access the FormController of the form you want to add the component into.

## Create a function called createComponent(); and write the code inside it to create and configure the component.

## You can refer to the given sample code for more information.

**Note:**

Add the invoke function i.e., **this.view.feedbackprompt.invokeFeedbackPrompt()** in the on click event of the button “open pop up” for static component creation

You can also add a Feedback Prompt 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 a component's Object \*/

var SlidingFeedback=new com.voltmxmp.feedbackprompt({

"clipBounds": true,

"height": "100%",

"id": "feedbackprompt",

"isVisible": true,

"left": "0dp",

"top": "0dp",

"width": "100%",

"zIndex":99

},{

},{});

SlidingFeedback.logoImageSrc = "voltmxmp\_sf\_icon\_wallet.png";

SlidingFeedback.btnDismissText = "Dismiss";

SlidingFeedback.btnSubmitText="Proceed";

SlidingFeedback.btnCancelText="Cancel";

SlidingFeedback.btnSendText="Submit";

SlidingFeedback.thankyouImageSrc="voltmxmp\_sf\_ico\_group.png";

SlidingFeedback.btnOkText="OK";

SlidingFeedback.feedbackTitle="Please enter your feedback";

SlidingFeedback.rateTitle="Please provide your feedback";

SlidingFeedback.thankyouTitle="Thank You!";

SlidingFeedback.thankyouMessage="Some text related to Thank You page with short description.";

SlidingFeedback.rateMessage="Some text with instructions and additional information.";

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

this.view.add(SlidingFeedback);

},

/\*Creating a button “openpopup” and adding this function to button “onclick” event\*/

onClickGiveFeedBack : function () {

this.view.feedbackprompt.invokeFeedbackPrompt();

},

1. **Save** the file.

**Note:** The component does not support landscape mode the component does not reflect on the templates tab.In the case of dynamic creation of the component in this case you must import the component directly into the templates tab**.**

## **Setting Properties**

The properties provided on the **Component** tab allow you to customize the UI elements and set constraints for the fields in the Feedback Prompt 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 a JavaScript.

### **General**

[[Open](javascript:void(0);)Invoke By Default](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Specifies if component should be invoked by default or not. By default the value of this property is set to true. |
| **Syntax:** | invokeByDefault |
| **Type:** | Boolean |
| **Read/Write:** | Read + Write |
| **Remarks** | If this property is set to false then you need to explicitly call the API [invokeFeedbackPrompt](https://docs.kony.com/marketplace/V8Marketplace/Content/Marketplace/slidingfeedback2.htm" \l "invokeFeedbackPrompt) to initialise the component. |
| **Example:** | this.view.componentID.invokeByDefault = true; |

### **Rating Screen**

[[Open](javascript:void(0);)Logo Image](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Specifies the name of the image to be set as the logo for the Feedback Prompt. |
| **Syntax:** | logoImageSrc |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks** | Before setting the property, do the following:   * 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.logoImageSrc = "logo.png"; |

[[Open](javascript:void(0);)Rate Prompt Title](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Specifies the title to be displayed in the rating page of the Feedback Prompt. |
| **Syntax:** | rateTitle |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks** | * The component does not display the text changes on Iris canvas because the default visibility is set to false. * The maximum characters limit is 35. You must handle the exceptions if you provide an invalid value |
| **Example:** | this.view.componentID.rateTitle = "Please Rate our App"; |

[[Open](javascript:void(0);)Rate Prompt Message](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Specifies the message to be displayed in the rating page of the Feedback Prompt. |
| **Syntax:** | rateMessage |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks** | * The component does not display the text changes on Iris canvas because the default visibility is set to false. * The maximum characters limit is 100. You must handle the exceptions if you provide an invalid value. |
| **Example:** | this.view.componentID.rateMessage = "Some text with instructions and additional information"; |

[[Open](javascript:void(0);)Dismiss Text](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Specifies the text to be displayed on the Dismiss button. |
| **Syntax:** | btnDismissText |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.btnDismissText= "DISMISS"; |

[[Open](javascript:void(0);)Submit Text](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Specifies the text to be displayed on the Submit button. |
| **Syntax:** | btnSubmitText |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Example:** | this.view.componentID.btnSubmitText= "SUBMIT"; |

### **Feedback Screen**

[[Open](javascript:void(0);)Feedback Screen Title](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Specifies the title to be displayed on the Feedback page of the Feedback Prompt. |
| **Syntax:** | feedbackTitle |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks** | * The component does not display the text changes on Iris canvas because the default visibility is set to false. * The maximum characters limit is 35. You must handle the exceptions if you provide an invalid value. |
| **Example:** | this.view.componentID.feedbackTitle= "Please submit your feedback"; |

[[Open](javascript:void(0);)Cancel Text](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Specifies the text to be displayed on the Cancel button. |
| **Syntax:** | btnCancelText |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks:** | * The component does not display the text changes on Iris canvas because the default visibility is set to false. |
| **Example:** | this.view.componentID.btnCancelText= "Cancel"; |

[[Open](javascript:void(0);)Send Text](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Specifies the text to be displayed on the Send button. |
| **Syntax:** | btnSendText |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks:** | * The component does not display the text changes on Iris canvas because the default visibility is set to false. |
| **Example:** | this.view.componentID.btnSendText= "SEND"; |

### **Thankyou Screen**

[[Open](javascript:void(0);)Thankyou Image](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Specifies the file name of the image to be set as the Thank You image. |
| **Syntax:** | thankyouImageSrc |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks** | Before setting the property, do the following:   * 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.thankyouImageSrc= "thankyou.png"; |

[[Open](javascript:void(0);)Thankyou Screen Title](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Specifies the title to be displayed in the Thank you page of the Feedback Prompt. |
| **Syntax:** | thankyouTitle |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks** | * The component does not display the text changes on Iris canvas because the default visibility is set to false. * The maximum characters limit is 35. You must handle the exceptions if you provide an invalid value. |
| **Example:** | this.view.componentID.thankyouTitle= "Thank You!"; |

[[Open](javascript:void(0);)Thankyou Message](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Specifies the message to be displayed on the Thank You page of the Feedback Prompt. |
| **Syntax:** | thankyouMessage |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks** | * The component does not display the text changes on Iris canvas because the default visibility is set to false. * The maximum characters limit is 100. You must handle the exceptions if you provide an invalid value. |
| **Example:** | this.view.componentID.thankyouMessage = "Some text related to Thank You page with short description"; |

[[Open](javascript:void(0);)OK Text](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | Specifies the text to be displayed on the OK button. |
| **Syntax:** | btnOkText |
| **Type:** | String |
| **Read/Write:** | Read + Write |
| **Remarks** | * The component does not display the text changes on Iris canvas because the default visibility is set to false. |
| **Example:** | this.view.componentID.btnOkText= "OK"; |

## **Defining 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 **Action** tab, click **Edit** against each event. For more information, refer [Add Actions](http://docs.kony.com/konylibrary/visualizer/visualizer_user_guide/Content/working_with_Action_Editor.htm).

[[Open](javascript:void(0);)onDismiss Event](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | The event is invoked when a user clicks **Dismiss**button of the Feedback Prompt component. |
| **Remarks:** | You must handle the functionality if you override the event. |
| **Syntax:** | onDismiss() |

[[Open](javascript:void(0);)onCancel Event](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | The event is invoked when a user clicks **Cancel** button of the Feedback Prompt component. |
| **Remarks:** | You must handle the functionality if you override the event |
| **Syntax:** | onCancel() |

[[Open](javascript:void(0);)onCompletion Event](javascript:void(0);)

|  |  |
| --- | --- |
| **Description:** | The event is invoked when a user clicks **OK** button in the Thank You page of the Feedback Prompt. |
| **Remarks:** | You must handle the functionality if you override the event |
| **Syntax:** | onCompletion() |

## **APIs**

The following APIs pertain to the Feedback Prompt component:

[[Open](javascript:void(0);)getFeedbackText](javascript:void(0);)

The API returns the text entered by a user in the Feedback section of the Feedback Prompt component.

**Syntax**

getFeedbackText()

**Parameters**

None

**Return Value**

*Text [String]:*  
Returns the text entered by a user in the Feedback section.

Remarks

* You must invoke the Feedback Prompt and enable the feedbackPopup before calling the API
* You must take care of the text validation.

**Example**

this.view.componentID.getFeedbackText();

[Open](javascript:void(0);)[invokeFeedbackPrompt](javascript:void(0);)

The API invokes the Feedback Prompt component.

**Syntax**

invokeFeedbackPrompt()

**Parameters**

**None**

**Return Value**

None

**Remarks**

Use the API to enable the Feedback Prompt in your app.

**Example**

this.view.componentID.invokeFeedbackPrompt();

# Revision History

* 1. App version 2.0.2

## Known Issues

## No issues

## Limitations

## Landscape mode is not supported