Date : 21-Mar -24

AUDIO COMPONENT

VERSION : 2.1.0

1. **OVERVIEW**

Audio is a component that can record and play audio on the device. The component can also share the recorded file from the device. Under the hood, the Audio component uses the [Social Share Component](https://marketplace.hclvoltmx.com/items/social-share?search=social%20share" \t "_blank) and [Native Function APIs.](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_user_guide/Content/UsingNativeFunctionAPIsAndWidgets.html" \t "_blank)

## Use case

1. You can change the icon, add custom actions, and use the APIs to control the functionality of the component.
2. You can use the component in scenarios where you want to provide a feature to record and play audio in your app.

## B. Percentage of re-use:

80-90% (Audio icon and actions can be customizable and skins are not customized but can be changed manually)

# **GETTING STARTED**

## Prerequisites

Before you start using the Audio 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 Audio 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.

Graphical user interface, text, application

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](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_user_guide/Content/C_UsingComponents.html" \l "add-a-component-to-a-form).

1. **[Configuring Native Settings (Android)](http://javascript:void(0);/" \t "_blank)**
2. **Configuring the Manifest file (Application Tag)**

To access files that you add to your project from the device, you need to create an XML file that specifies the directories that you want to share. For more information, refer to [Specifying the Sharable Directories in the XML file](https://docs.kony.com/konylibrary/visualizer/viz_api_dev_guide/content/sharefilesandroid.htm#Specify2).

After you create the **XML** file, copy the file into the following folder of your project workspace: **resources/mobile/native/android/xml**. If the **xml** folder does not exist, you can create the folder.

After you copy the file to the Android folder, you need to configure the **Manifest properties** of the project.

Follow the given steps to configure the application tag.

1. Open your **Project Settings** from Volt MX Iris.
2. Go to the **Native** – **Android Mobile/Tablet**
3. Under **Android**, go to tab **Tags**.

Graphical user interface, text, application

Description automatically generated

1. Enter the given code snippet in **Child tag entries under <application> tag**.

<provider android:name="androidx.core.content.FileProvider"

android:authorities="${applicationId}.provider"

android:exported="false"

android:grantUriPermissions="true">

<meta-data android:name="android.support.FILE\_PROVIDER\_PATHS"

android:resource="@xml/filepaths" />

</provider>

1. **Run-time Permissions**

On the Android platform, the Audio component requires the **WRITE\_EXTERNAL\_STORAGE** and **RCORD\_AUDIO** permissions.

To enable the permissions, follow these steps:

1. From the left navigation bar, select **Project Settings**.
2. From the **Project Settings** window, go to **Native** → **Android Mobile/Tablet**.
3. Scroll down to **Manifest Permissions, Tags and Gradle Build Entries**.
4. Set the **WRITE\_EXTERNAL\_STORAGE** and **RCORD\_AUDIO** permissions to **true**.   
   To set a permission to true, select the permission from the left panel and click **Add >**.   
   Graphical user interface, application

   Description automatically generated

iii. **Note:** Need to enable "Use Google Pay Location Services" for android in the project settings. From **Project Settings** window, go to **Native → Android Mobile/Tablet** and enable **Use** **Google Pay Location Services**.

A screenshot of a computer

Description automatically generated

1. **Configuring Native Settings (iOS)**

Follow the given steps to enable the permissions.

1. From the **Project** explorer, navigate to the **Assets** tab.
2. Right click **Media** and select **Resource Location** to open the project resources folder.   
   Graphical user interface, application

   Description automatically generated
3. In the browser window that opens, navigate to the **common** folder.
4. Open the **infoplist\_configuration.json** file with a text or code editor.
5. Add   "NSMicrophoneUsageDescription" : "This is a dummy description"

Graphical user interface, text, application

Description automatically generated

1. Save the file.

# **3.REFERENCES**

## A. Dynamic Usage

You can also add an **Audio** 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.

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

/\* Creating the component's object \*/

var audio = new com.voltmxmp.audio(

{

"clipBounds": true,

"height": "66%",

"id": "audio",

"isVisible": true,

"left": "0dp",

"top": "0dp",

"width": "100%",

"zIndex": 1

}, {}, {});

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

audio.iconImage = "wave\_graphic\_active.png";

/\* Defining the component's events \*/

audio.recordSuccessCallback = function()

{

alert("Recording completed.");

};

audio.recordErrorCallback = function()

{

alert("Recording could not be completed.");

};

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

this.view.add(audio);

1. Save the file.

## B. Properties

The properties provided on the **Component** tab allow you to customize the UI elements in the **Audio** component. You can set the properties directly on the **Component** tab or by writing a JavaScript.

**[Open](javascript:void(0);)1. Icon Image**

|  |  |  |
| --- | --- | --- |
| **Category:** | Pass Through | |
| **Description:** | Specifies the source of the image that you want to use as the component icon. | |
| **Syntax:** | iconImage | |
| **Type:** | * Image * String | |
| **Read/Write:** | Read + Write | |
| **Default Value:** | "wave\_graphic\_active.png" | |
| **Remarks:** | Make sure that the image exists in the *resources*\*common* folder of your workspace. | |
| **Example:** |  | |
| this.view.componentID.iconImage = "audioIcon.png"; | |

* 1. **Events**

The component invokes events when its corresponding action is performed. You can configure any logic you want the component to perform whenever an event occurs.

**[Open](javascript:void(0);)1. recordSuccessCallback**

|  |  |  |
| --- | --- | --- |
| **Category:** | Custom | |
| **Description:** | Invoked when the component successfully records the audio. | |
| **Syntax:** | recordSuccessCallback | |
| **Parameters:** | None | |
| **Example:** | this.view.componentID.recordSuccessCallback = function()  {  alert("Recording completed.");  }.bind(this); | |
|  | | |

**[Open](javascript:void(0);)2. recordErrorCallback**

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Invoked when an error occurs while the component records the audio. |
| **Syntax:** | recordErrorCallback |
| **Parameters:** | None |
| **Example:** | this.view.componentID.recordErrorCallback = function()  {  alert("Recording could not be completed.");  }.bind(this); |

**D. APIs**

The following APIs pertain to the Audio component:

[Open](javascript:void(0);)1. startRecord

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Starts recording the audio. |
| **Syntax:** | startRecord() |
| **Parameters:** | None |
| **Return Value:** | None |
| **Example:** |  |
| this.view.componentID.startRecord(); | | |

[Open](javascript:void(0);)2. stopRecord

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Stops recording the audio. |
| **Syntax:** | stopRecord() |
| **Parameters:** | None |
| **Return Value:** | None |
| **Example:** |  |
| this.view.componentID.stopRecord(); | | |

[Open](javascript:void(0);)3. togglePlayPause

|  |  |  |
| --- | --- | --- |
| **Category:** | Custom | |
| **Description:** | Plays or pauses the audio playback. | |
| **Syntax:** | togglePlayPause() | |
| **Parameters:** | None | |
| **Return Value:** | None | |
| **Remarks:** | The component does not play the audio in the following scenarios:   * The component is recording the audio. * There is no recording available. | |
| **Example:** |  | |
| this.view.componentID.togglePlayPause(); | |

[Open](javascript:void(0);)4. isrecordedFileExist

|  |  |  |
| --- | --- | --- |
| **Category:** | Custom | |
| **Description:** | Checks whether a recorded file exists in the storage. | |
| **Syntax:** | isrecordedFileExist() | |
| **Parameters:** | None | |
| **Return Value:** | *doesFileExist [Boolean]*  The value is true if a recorded file exists. The value is false if a recorded file does not exist. | |
| **Example:** | var doesFileExist = this.view.componentID.isrecordedFileExist();  if(doesFileExist == true)  {  this.view.componentID.togglePlayPause();  } | |
|  | | |

[Open](javascript:void(0);)5. shareRecordFile

|  |  |
| --- | --- |
| **Category:** | Custom |
| **Description:** | Shares the recorded file using the name and base64 value. |
| **Syntax:** | shareRecordFile() |
| **Parameters:** | None |
| **Return Value:** | None |
| **Remarks:** | To share the recorded file, the API uses the [Social Share component](https://marketplace.hclvoltmx.com/items/social-share?search=social%20share) from Forge. |
| **Example:** |  |

this.view.componentID.shareRecordFile();

# **4. REVISION HISTORY**

App version 2.1.0

1. **Known Issues**

No known issues

1. **Limitations**

No limitations