Date: 1-Aug-24

PDF VIEWER

VERSION: 2.7.0

# **OVERVIEW**

PDF Viewer is a plug and play component to display PDF files in your app. It can render PDF files from the user's device or from a web URL. Under the hood, the component uses Native function APIs.

## **Use case**

### View PDF file

## **Features**

### User can view online PDF file

### User can view local PDF file on device

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

80-90%. (Data can be customizable and skins are not customized but can be changed manually)

# **GETTING STARTED**

## **Prerequisites**

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

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

## **Platforms Supported**

### Mobile

#### iOS

#### Android

### Tablets

#### iOS

#### Android

### PWA

## Notes: Should contain info on whether all platforms perform similar, or there are exceptions (say on iOS a feature behaves differently, for example, or certain API do not work in PWA etc. We can mention it here if the component has any such platform-specific deviations and link to a separate section)

## **Importing the app**

## Open your app project in Volt MX Iris.

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

Graphical user interface, text, application

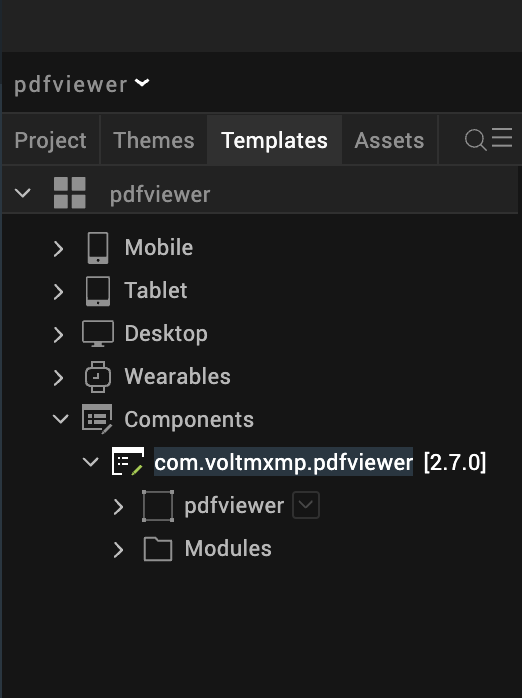
Description automatically generated

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.



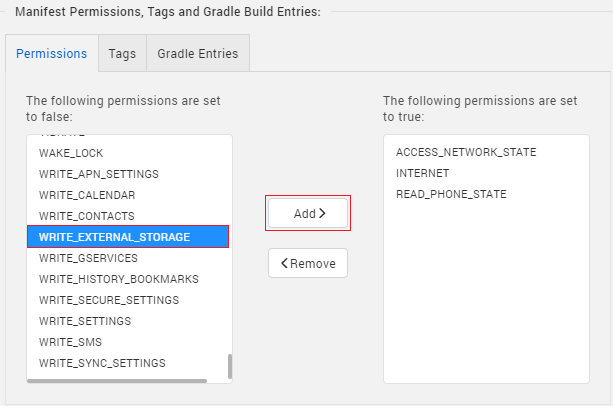
## Once you have imported a component to your project, you can easily add the component to a form. For more information, refer [Add a Component to a Form](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_user_guide/Content/C_UsingComponents.html#add-a-component-to-a-form).

## **Configuring Native Settings (Android)**

### **Run-time Permissions**

On the Android platform, the PDF Viewer component requires the **WRITE\_EXTERNAL\_STORAGE** 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** permission to **true**.  
   To set a permission to true, select the permission from the left panel and click **Add >**.  
   

### **Manifest Properties**

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://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_api_dev_guide/content/sharefilesandroid.html#specifying-the-sharable-directories-in-the-xml-file).

For Android, to ensure the stability of pdf online rendering, the pdf file will be downloaded into “files/” directory(voltmx.io.FileSystem.getDataDirectoryPath()) in your app’s internal storage. Please make sure this directory is specified in XML file.

<paths>

<files-path path="/" name="files"/>

</paths>

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.

To configure the Manifest properties, follow these steps:

1. From the Project Settings window, navigate to **Native** → **Android Mobile/Tablet**.
2. Scroll down to **Manifest Permissions, Tags and Gradle Build Entries** and switch to the **Tags** tab.
3. In the **Child tag entries under application tag** box, type the following code:

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

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

android:exported="false"

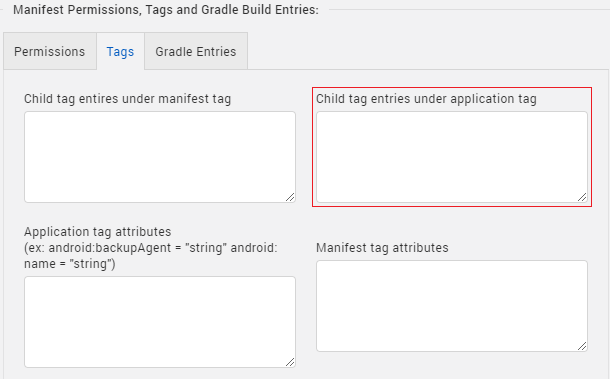
android:grantUriPermissions="true">

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

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

</provider>

**Note:** In the code snippet, make sure that you replace **filenames** with the name of your XML file. Make sure that you type the name of the file without the **.xml** extension.



## **Building and previewing the app**

## Refer to below links:

### [**BUILD & GENERATE NATIVE APPS**](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_user_guide/Content/Cloud_Build_in_VoltMX_Iris.html#cloud)

### [**PUBLISH APPS TO ENTERPRISE APP STORE**](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_user_guide/Content/EAS.html#accessing)

# **REFERENCES**

## **Dynamic Usage**

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 a component's Object \*/

var pdfviewer = new com.voltmxmp.pdfviewer(

{

"clipbounds": true,

"height": "80%",

"id": "pdfviewer",

"isVisible": true,

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

"left": "0dp",

"top": "100dp",

"width": "100%"

}, {}, {});

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

pdfviewer.url = "https://www.adobe.com/support/products/enterprise/knowledgecenter/media/c4611\_sample\_explain.pdf";

pdfviewer.pdfType = "Online";

pdfviewer.renderType = "Manual";

pdfviewer.setIphonePath = "sample";

pdfviewer.setAndroidPath = "/storage/emulated/0/Download/sample.pdf";

pdfviewer.setDesktopWeb = "desktopweb/web/localfiles/sample.pdf";

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

this.view.add(pdfviewer);

1. Save the file.

## **Properties**

#### **General**

1. **Type**

| **Description:** | Specifies the type of the PDF file that you want to render at runtime. |
| --- | --- |
| **Syntax:** | pdfType |
| **Type:** | * List Selector * String |
| **Read/Write:** | Write |
| **Values:** | * Online * Local File Path |
| **Default Value:** | Online |
| **Remarks:** | In Iris, you can set the value from a [drop down list](about:blank). |
| **Example:** | this.view.<componentID>.pdfType = "Online"; |

Open

1. **Render**

| **Description:** | Specifies whether you want to render the PDF file automatically. |
| --- | --- |
| **Syntax:** | renderType |
| **Type:** | * List Selector * String |
| **Read/Write:** | Write |
| **Values:** | * Automatic * Manual |
| **Default Value:** | Automatic |
| **Remarks:** | * If you set the value to **Manual**, you need to call the [render](https://docs.kony.com/marketplace/PDFViewer/Content/Reference.htm#Render) API to render the PDF at run time. * In Iris, you can set the value from a [drop down list](about:blank). |
| **Example:** | this.view.<componentID>.renderType = "Automatic"; |

#### **Online**

**1. URL**

| **Description:** | Specifies the URL of the PDF that you want to render at runtime. |
| --- | --- |
| **Syntax**: | url |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks:** | * Changes made to this property are reflected only at run time. iOS,does not support http protocol * For Android, make sure the “files/” directory(voltmx.io.FileSystem.getDataDirectoryPath()) in your app’s internal storage is specified in XML file. For more information, refer to [Specifying the Sharable Directories in the XML file](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_api_dev_guide/content/sharefilesandroid.html#specifying-the-sharable-directories-in-the-xml-file) |
| **Example:** | this.view.<componentID>.url = “https://www.adobe.com/support/products/enterprise/knowledgecenter/media/c4611\_sample\_explain.pdf” |

#### **Local File Path**

**1. Android**

| **Description:** | Specifies the local file that you want to render on Android at run time. |
| --- | --- |
| **Syntax:** | setAndroidPath |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks:** | * The device must have a PDF viewer installed to open the PDF file. Otherwise the component throws an exception. * If you want to view files that you include in your Iris project, make sure that you share the file path as a URI. For more information, refer to [Specifying the Sharable Directories in the XML file](https://opensource.hcltechsw.com/volt-mx-docs/docs/documentation/Iris/iris_api_dev_guide/content/sharefilesandroid.html#specifying-the-sharable-directories-in-the-xml-file). * Make sure that you place the PDF file in the specified directory before preview it on local. For example, put the sample.pdf to download folder of device. |
| **Example:** | this.view.<componentID>.setAndroidPath =  /storage/emulated/0/Download/sample.pdf"; |

[](about:blank)

**2. iOS**

| **Description:** | Specifies the file that you want to render on iOS at run time. |
| --- | --- |
| **Syntax:** | setIphonePath |
| **Type:** | String |
| **Read/Write:** | Write |
| **Remarks:** | Make sure that you place the PDF file in either of the following locations:   * The raw folder - *<workspace>*\*<project>*\*resources*\*mobile*\*native*\*iphone*\*raw* * Stored in the project level storage - You need to provide the file path. \*var*\*mobile*\*Containers*\*Data*\*Application*\*<Application-sandboxid>*\*Documents*\*sample.pdf* |
| **Example:** | this.view.<componentID>.setIphonePath = "sample"; |

**3. Desktop Web**

| **Description:** | Specifies the file that you want to render on Desktop Web at run time. |
| --- | --- |
| **Syntax:** | setDesktopWeb |
| **Type:** | String |
| **Read/Write:** | Write |
| **Example:** | this.view.<componentID>.setDesktopWeb = "desktopweb/web/localfiles/sample.pdf"; |

## **Events**

1. **onErrorCallback**

| **Description:** | Invoked when an error occurs within the component. |
| --- | --- |
| **Syntax:** | onErrorCallback |
| **Parameters:** | *error [JSON]* : Details about the error, such as **error code** and **error message**. |
| **Example:** | this.view.<componentID>.onErrorCallback = function(error)  {  alert("Error: "+JSON.stringify(error));  }.bind(this); |

## **API’s**

## **render**

| **Description:** | Renders the specified PDF at run time. |
| --- | --- |
| **Syntax:** | render() |
| **Parameters:** | None |
| **Return Value:** | None |
| **Remarks:** | * Use this API only when the render type is **manual**. * Make sure that you set the following properties before you call this API.   + pdfType   + URL (if the **pdfType** is **Online**)   + Android File Path (if the **pdfType** is **Local File Path**)   + iOS File Path (if the **pdfType** is **Local File Path**) |
| **Example:** | this.view.componentID.render(); |

1. **setURL**

| **Description:** | Set the URL of the PDF that you want to render at runtime. |
| --- | --- |
| **Syntax:** | setURL(url) |
| **Parameters:** | *url [String]* : The URL of the PDF that you want to render. |
| **Return Value:** | None |
| **Example:** | this.view.<componentID>.setURL(  " https://www.adobe.com/support/products/enterprise/knowledgecenter/media/c4611\_sample\_explain.pdf "); |

1. **setAndroidFilePath**

| **Description:** | Sets the file path of the PDF that you want to render on Android. |
| --- | --- |
| **Syntax:** | setAndroidFilePath(path) |
| **Parameters:** | *path [String]* : The file path of the PDF that you want to render on Android. |
| **Return Value:** | None |
| **Example:** | this.view.<componentID>.setAndroidFilePath(  "/storage/emulated/0/Download/sample.pdf"); |

1. **setIphoneFilePath**

| **Description:** | Sets the file path of the PDF that you want to render on iOS. |
| --- | --- |
| **Syntax:** | setIphoneFilePath(path) |
| **Parameters:** | *path [String]* : The file path of the PDF you want to render on iOS. |
| **Return Value:** | None |
| **Remarks:** | Make sure that you place the PDF file in either of the following locations:   * The raw folder - *<workspace>*\*<project>*\*resources*\*mobile*\*native*\*iphone*\*raw* * Stored in the project level storage - You need to provide the file path. \*var*\*mobile*\*Containers*\*Data*\*Application*\*<Application-sandboxid>*\*Documents*\*sample.pdf* |
| **Example:** | this.view.<componentID>.setIphoneFilePath("sample"); |

1. **setDesktopWeb**

| **Description:** | Sets the file path of the PDF that you want to render on Desktop Web. |
| --- | --- |
| **Syntax:** | setDesktopWeb(path) |
| **Parameters:** | *path [String]* : The file path of the PDF that you want to render on Desktop Web. |
| **Return Value:** | None |
| **Remark:** | Make sure that you place the PDF file in the following location:   * The local file folder - *<project>*\ web\localfiles\sample.pdf |
| **Example:** | this.view.<componentID>.setDesktopWebFilePath("desktopweb/web/localfiles/sample.pdf"); |

# **REVISION HISTORY**

App version 2.7.0:

## **Known Issues**

## No known issues

## **Limitations**

* To change the URL dynamically in the same form, you need to use the [setURL](https://docs.kony.com/marketplace/PDFViewer/Content/Reference.htm#setURL) API instead of setting the [URL](https://docs.kony.com/marketplace/PDFViewer/Content/Reference.htm#URL) property.
* To change the path dynamically in the same form, you need to use the [setIphoneFilePath](https://docs.kony.com/marketplace/PDFViewer/Content/Reference.htm#setIphoneFilePath) and [setAndroidFilePath](https://docs.kony.com/marketplace/PDFViewer/Content/Reference.htm#setAndroidFilePath) APIs instead of setting the [setAndroidPath](https://docs.kony.com/marketplace/PDFViewer/Content/Reference.htm#AndroidFile) and [setIphonePath](https://docs.kony.com/marketplace/PDFViewer/Content/Reference.htm#iOSFile) properties.
* To render a PDF on Desktop Web, you need to use the [setDesktopWeb](https://docs.kony.com/marketplace/PDFViewer/Content/Reference.htm#setDesktopWebFilePath) API.