Date: 13-Nov-2024

TEXT RECOGNITION

VERSION: 1.0.0

# **OVERVIEW**

The Text Recognition component can recognize text from the camera and images. Under the hood, the component contains NFIs that use the Google MLKit on Android and Apple Vision API on iOS.

## **Use case:**

1. An event organizer app, where a user needs to read the text from the image through the Camera.
2. A text translator app that scans the text from the boards and translates into required language.

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

Approximate 90% of reuse.

**C. Features**

1. The component utilizes Google MLKit on Android and Apple Vision API on iOS for Recognizing the text.

**Reference links:**

* MLKit<https://developers.google.com/ml-kit/vision/text-recognition/v2/android>
* Apple Vision API <https://developer.apple.com/documentation/vision/recognizing-text-in-images>

1. It supports the following languages: [https://developers.google.com/ml-kit/vision/text-recognition/languages](https://apc01.safelinks.protection.outlook.com/?url=https%3A%2F%2Fdevelopers.google.com%2Fml-kit%2Fvision%2Ftext-recognition%2Flanguages&data=05%7C01%7Cvishnuvardhan.nasina%40hcl.com%7C1ebf2f047bc044ae6a9e08da7aa24d0b%7C189de737c93a4f5a8b686f4ca9941912%7C0%7C0%7C637957134758920367%7CUnknown%7CTWFpbGZsb3d8eyJWIjoiMC4wLjAwMDAiLCJQIjoiV2luMzIiLCJBTiI6Ik1haWwiLCJXVCI6Mn0%3D%7C3000%7C%7C%7C&sdata=Mr4PN%2F36LyGk2crJLOhg%2BoFQuAmEQ6lDQsumvkE4VC8%3D&reserved=0)

# **GETTING STARTED**

## **Prerequisites**

Before you start using the Text Recognition component, ensure the following

• Volt MX Iris

## **Platforms Supported**

1. Mobile
2. iOS
3. Android

## **Importing the component**

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

## **To import the Text Recognition 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

1. Right-click **Components**, and then select **Import Component**. The **Import Component** dialog box appears.

Graphical user interface, text, application, Teams

Description automatically generated

1. 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)

## **D. 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 User Guide.

You can then run your app to see the Text Recognition work in real time.

**3. REFERENCES**

## **Dynamic Usage**

You can also add a **Text Recognition** 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.

createComponent : function(){

/\* Creating TextRecognition component instance \*/

var textRecognition = new com.voltmxmp.textrecognition({

id: "textrecognition",

isVisible: true,

top:"0dp",

left:"0dp",

width:"100%",

height:"100%",

clipBounds: true,

autogrowMode: voltmx.flex.AUTOGROW\_NONE,

skin: "slFbox",

zIndex:1

},{},{});

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

this.view.add(textRecognition);

/\* setScanResultCallback API \*/

this.view.textrecognition.setScanResultCallback(function(result){

try{

alert("Scan Result : "+result);

}

catch(err){

alert("e= "+err);

}

});

}

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

**Configuring Native Settings (iOS)**

To configure the native settings for iOS, follow these steps:

1. From the **Project** explorer, go to **Assets** and expand **Media**.
2. Right-click **Common**, and then select **Resource Location**. Volt MX Iris opens the common resources folder in a file explorer.  
   Graphical user interface, text

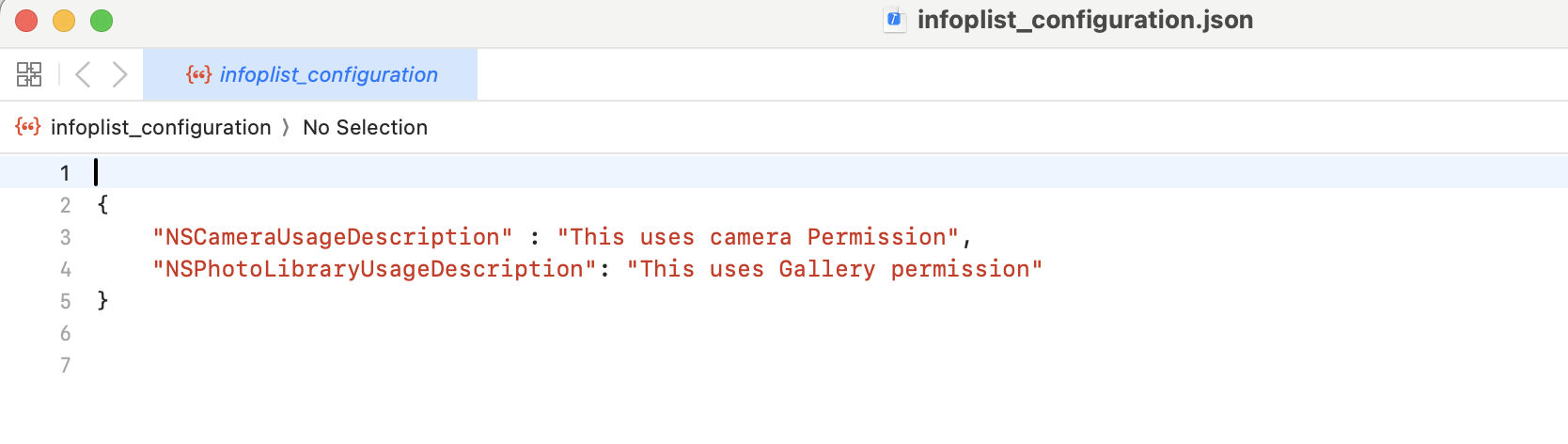
   Description automatically generated
3. Open the **infoplist\_configuration.json** file with a text or code editor.
4. At the end of the file, type the following code. You can change the description based on your preference.

{

"NSCameraUsageDescription" : "This uses camera Permission",

"NSPhotoLibraryUsageDescription": "This uses Gallery permission"

}



5. Save the file.

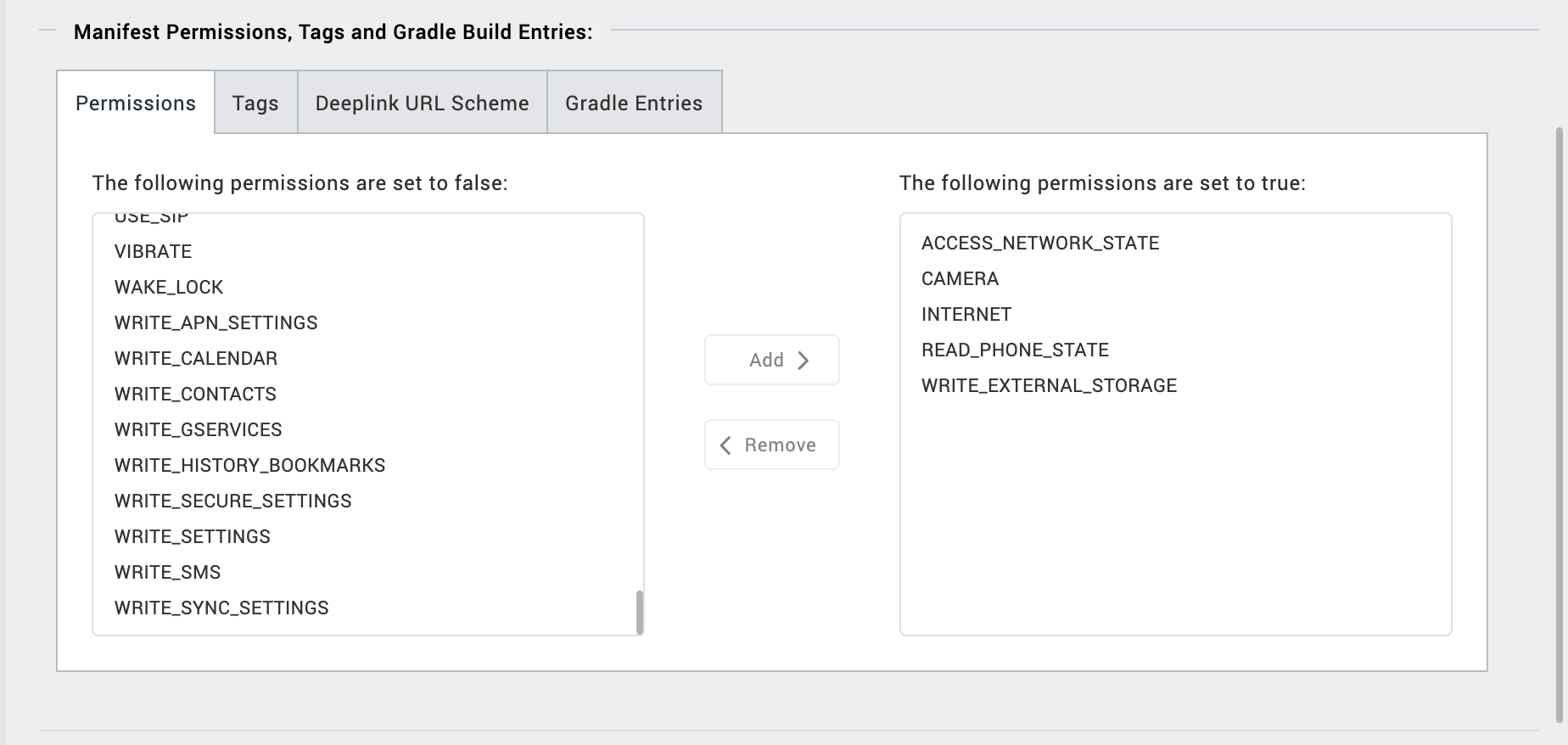
**Configuring Native Settings (Android)**

To configure the native settings for Android, follow these steps:

1. From the left navigation menu, click **Project Settings**.

2. In the Project Settings window, go to **Native** → **Android Mobile/Tablet**.

3. Set the **Required** permissions to **true**.

To set a permission to **true**, select the permission from the left panel, and then click **Add >**.  


4. Switch to the **Gradle Entries** tab.

5. In the **build.gradle entries to Suffix** box, type the given code based on the version of the component.

dependencies {

implementation 'com.google.mlkit:text-recognition:16.0.1'

implementation 'androidx.camera:camera-core:1.3.4'

implementation 'androidx.camera:camera-camera2:1.3.4'

implementation 'androidx.camera:camera-lifecycle:1.3.4'

implementation 'androidx.camera:camera-video:1.3.4'

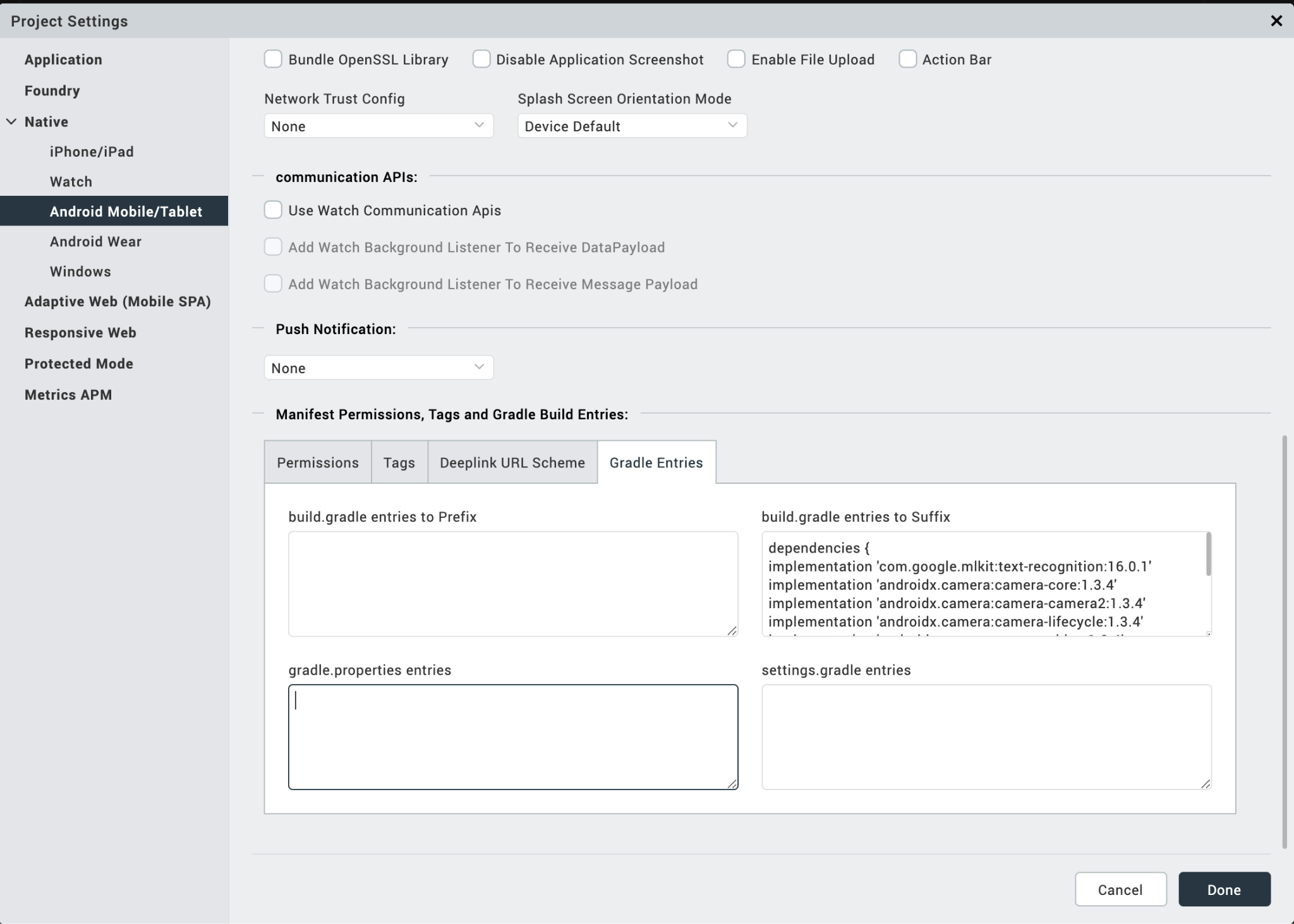
implementation 'androidx.camera:camera-view:1.3.4'

implementation 'androidx.camera:camera-extensions:1.3.4'

implementation(platform("org.jetbrains.kotlin:kotlin-bom:1.8.0"))

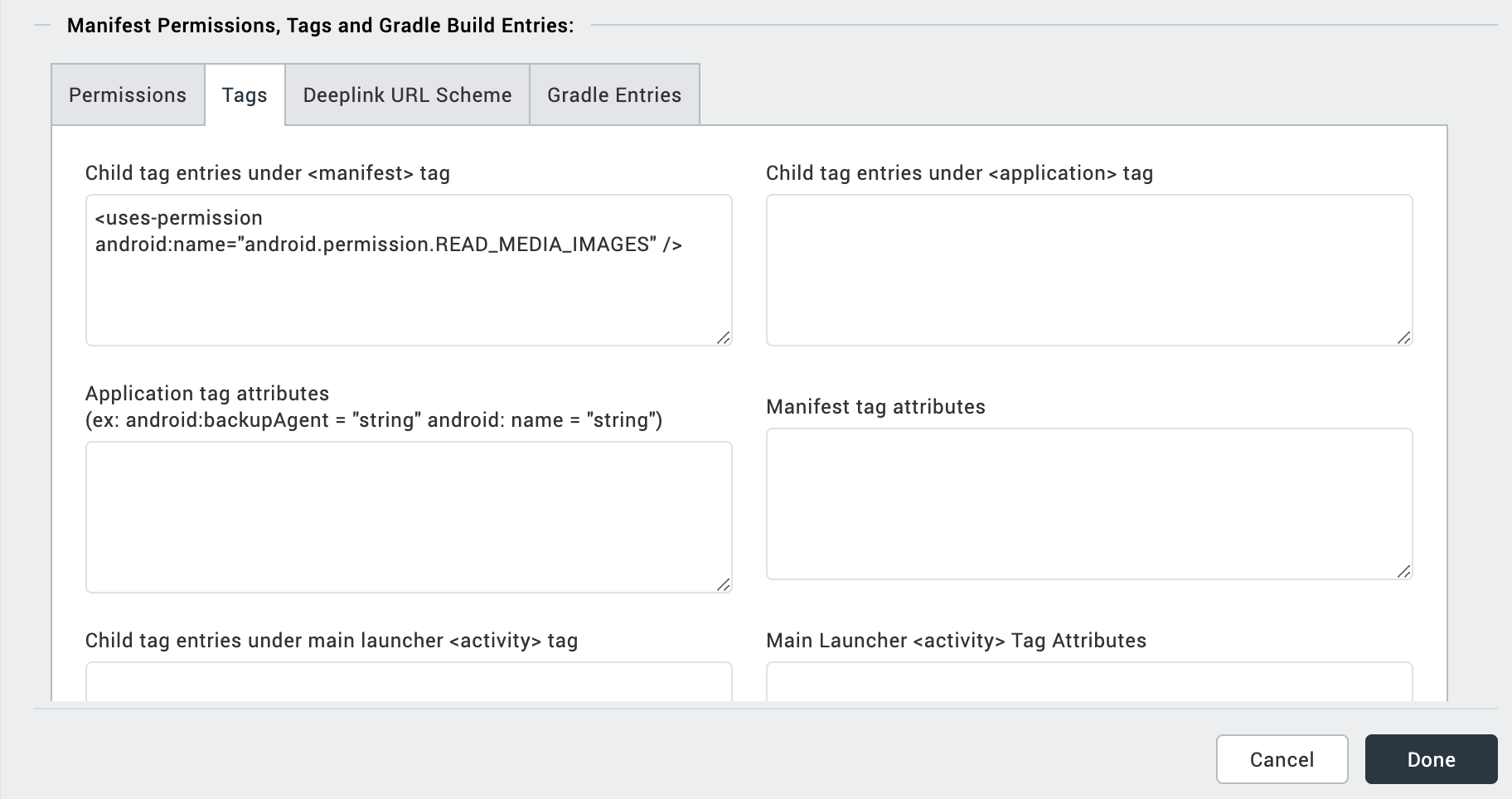
implementation 'androidx.lifecycle:lifecycle-livedata-ktx:2.3.1'

}

****

6. Enter the given snippet in Child tag entries under <manifest > tag

<uses-permission android:name="android.permission.READ\_MEDIA\_IMAGES" />



## **API’s**

### **i. setScanResultCallback**

### 

| **Description:** | Returns the scanned text. This setScanResultCallback should be invoked in the onNavigate function. |
| --- | --- |
| **Syntax:** | setScanResultCallback |
| **Parameters**: | Callback, And in callback, data : The data that is recognised |
| **Example:** | this.view.textrecognition.setScanResultCallback(callback); |

### **ii. getImgRawbytesFromGallery**

| **Description:** | Returns the rawbytes for the image selected from the gallery. |
| --- | --- |
| **Syntax:** | getImgRawbytesFromGallery |
| **Parameters**: | function(): a callback function |
| **Example:** | this.view.textrecognition.getImgRawbytesFromGallery(callback); |

1. **REVISION HISTORY**

App version 1.0.0:

## **A. Limitations**

* NA

**B. Known Issue**

* NA