Date: 18-April-2024

GOOGLE MLKIT TEXT RECOGNITION

version: 1.0.0

# **OVERVIEW**

The MLKit Text recognition component can recognize text from the camera and images. Under the hood, the component contains NFIs that use the Google MLKit Text Recognition API.

## **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 translate into required language.

**B . Percentage of re-use:**

Approximate 90 % of reuse.

**C. Features**

1. The component utilizes Google MLKit Text Recognition SDK for Recognizing the text.
2. 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)

**2. GETTING STARTED**

## **A. Prerequisites**

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

• Volt MX Iris

## **Platforms Supported**

### Mobile

#### iOS

#### Android

## **Importing the component**

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

## **To import the Google MLKit Text Recognition component, do the following:**

## Open your app project in Volt MX Iris.

2. 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)

## **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**Google MLKit 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.

/\* Creating Google MLKit Text Recognition component instance \*/

var MKTextRecognition = new com.technohub.MKTextRecognition({

"height": "50%",

"id": "MKTextRecognitio",

"isVisible": true,

"left": "0dp",

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

"isModalContainer": false,

"skin": "slFbox",

"top": "0dp",

"width": "100%",

"zIndex": 1,

"overrides": {

" MKTextRecognition ": {

"right": "viz.val\_cleared",

"bottom": "viz.val\_cleared",

"minWidth": "viz.val\_cleared",

"minHeight": "viz.val\_cleared",

"maxWidth": "viz.val\_cleared",

"maxHeight": "viz.val\_cleared",

"centerX": "viz.val\_cleared",

"centerY": "viz.val\_cleared"

}

}

}, {

"overrides": {}

}, {

"overrides": {}

});

/\*Adding the Google MLKit Text Recognition component to a Form\*/

this.view.add(MKTextRecognition);

},

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.

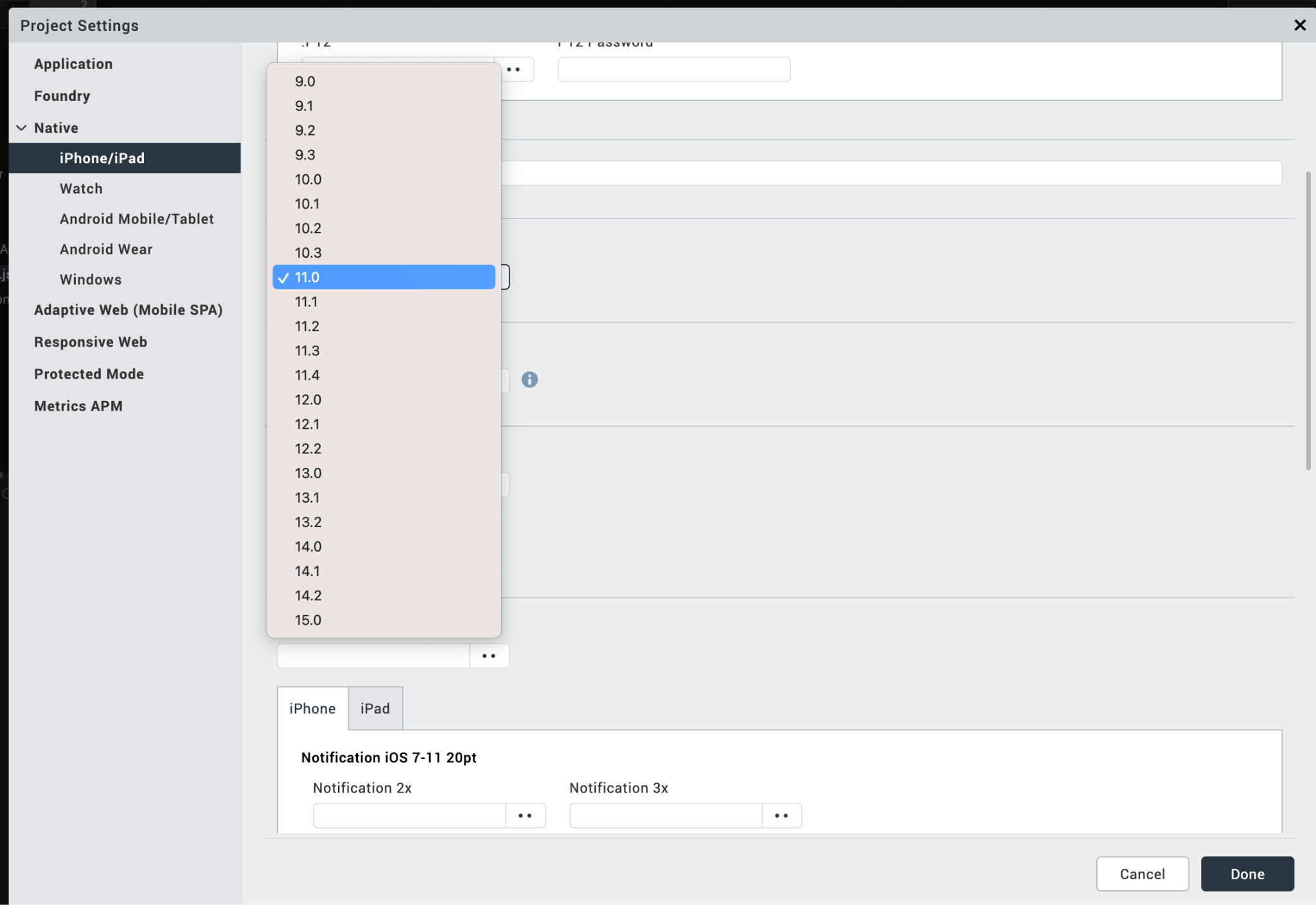
"NSCameraUsageDescription" : "Your Description"

Text

Description automatically generated with medium confidence

5. Save the file.

**Configuring Deployment Target**

1. From the left navigation menu, click **Project Settings**.
2. In the **Project Settings** window, go to **Native** → **iPhone/iPad**.
3. Under **Target Versions**, from the **iOS Version** list, select **11.0** or higher.  
   

After you configure the native settings, you can Build and Run your app to see the Google MLKit Text Recognition component work in real time for iOS.

**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 **CAMERA** permission to **true**.  
To set a permission to **true**, select the permission from the left panel, and then click **Add >**.  
Graphical user interface, application

Description automatically generated

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 {

// CameraX

implementation "androidx.camera:camera-camera2:1.0.0-beta11"

implementation "androidx.camera:camera-lifecycle:1.0.0-beta11"

implementation "androidx.camera:camera-view:1.0.0-alpha18"

// MLKit Text recognitions

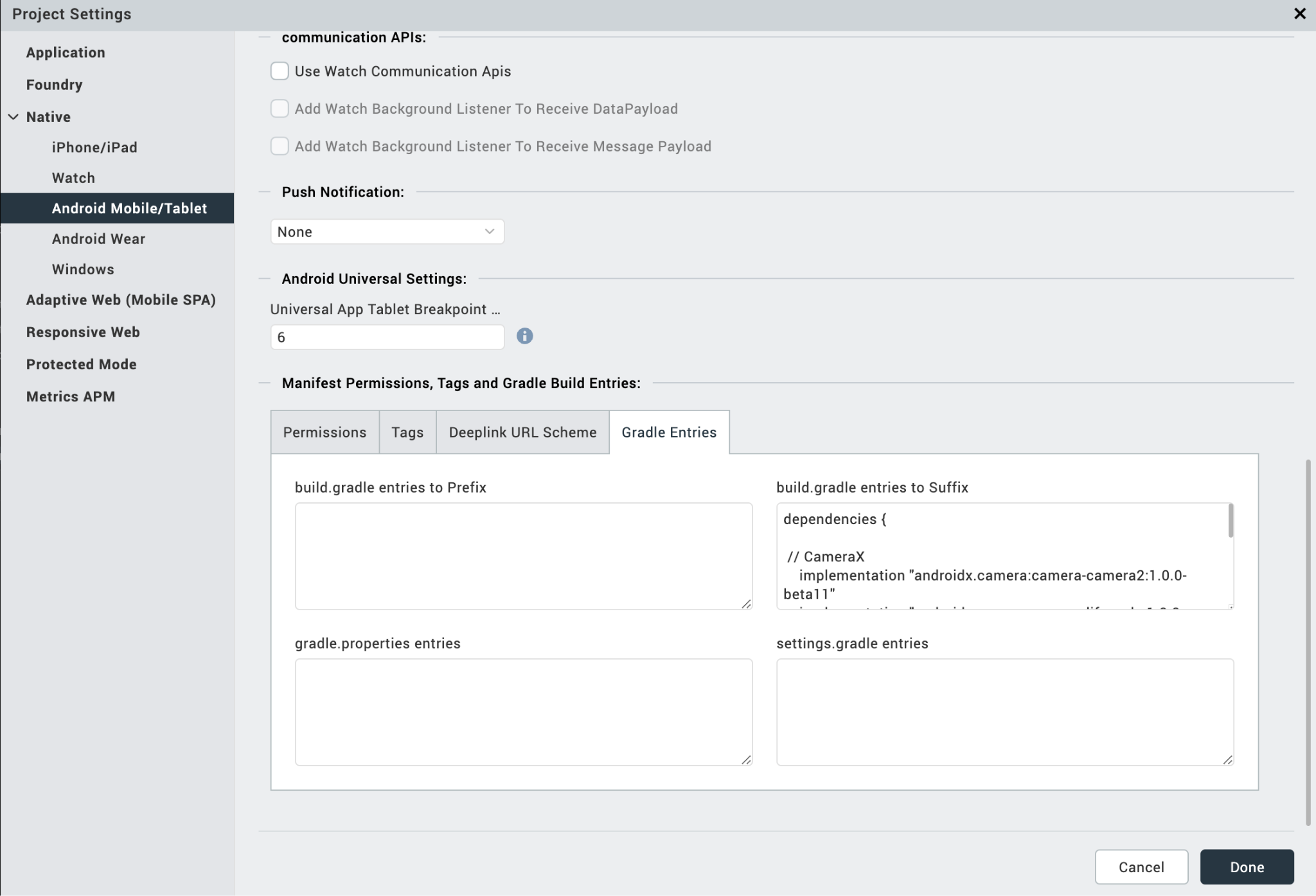
implementation 'com.google.android.gms:play-services-mlkit-text-recognition:18.0.0'

implementation 'org.jetbrains:annotations:15.0'

}

// used for 32 bit support

android { defaultConfig { renderscriptSupportModeEnabled false}}

**  
  
NOTE :**

* If you do not add the Gradle entries to your project, the app crashes.
* You can update the **Play Services** version in the later builds of your app.

After you configure the native settings, you can Build and Run your app to see the Barcode/QR Code Scanner component work in real time.

## **API’s**

### **i. startScaning**

|  |  |
| --- | --- |
| **Description:** | Starts the scanning. |
| **Syntax:** | startScaning () |
| **Parameters:** | None |
| **Return Value:** | None |
| **Remarks:** | Call this API first time to start the Camera Session. |
| **Example:** | this.view.MKTextRecognition.startScaning (); |

### **ii.** **stopScaning**

|  |  |
| --- | --- |
| **Description:** | Stops the scanning. |
| **Syntax:** | stopScaning () |
| **Parameters:** | None |
| **Return Value:** | None |
| **Remarks:** | Call this API after successful scan to avoid the continues scan.  And call this API when app moves to background. |
| **Example:** | this.view.MKTextRecognition.stopScaning (); |

### **iii.** **resultFromImageURL**

|  |  |
| --- | --- |
| **Description**:  Recognise the text from the URL contains image. | Turns on/off the device Flash light. |
| **Syntax:** resultFromImageURL() | enableFlash () |
| **Parameters:** String, Result callback | None |
| **Return Value:** None | None |
| **Example:** this.view. MKTextRecognition. resultFromImageURL (“URL”); | this.view.MLKitBCScanner.enableFlash(true/false); |
| **iv.** **resultFromImage**  |  |  | | --- | --- | | **Description:** | Recognise the text from the native image. | | **Syntax:** | resultFromImage () | | **Parameters:** | UIImage/Image, Result callback | | **Return Value:** | None | | **Remarks:** | Call this API to recognize the text from the native UIImage/Image object | | **Example:** | this.view.MKTextRecognition.resultFromImage (); | |  |
| **v.** **resultFromBase64**  |  |  | | --- | --- | | **Description:** | Recognise the text from the base 64 string. | | **Syntax:** | resultFromBase64 () | | **Parameters:** | String, Result Callback | | **Return Value:** | None | | **Remarks:** | Call this API to recognize the text from the native UIImage/Image object | | **Example:** | this.view.MKTextRecognition.resultFromBase64(); | | | |  |

### **C. Events**

### **i.** **setScanResultCallback**

|  |  |
| --- | --- |
| **Description:** | Invoked after the component recognises the text. This setScanResultCallback should be invoked in onNavigate function. |
| **Syntax:** | setScanResultCallback |
| **Parameters**: | Callback, And in callback, data[JSON] : The data that is recognised |
| **Remarks:** | If you want the camera to recognise text continuously, do not call the stopSession() in setScanResultCallback. |
| **Example:** | onNavigate: function(){  this.view.MLKitBCScanner. setScanResultCallback(callback);  }, |

### **ii.** **addInitCompleteCallback**

|  |  |
| --- | --- |
| **Description:** | Invoked after the component completes the initialization. |
| **Syntax:** | addInitCompleteCallback |
| **Parameters**: | function(): a call back function |
| **Remarks:** | startSession() invocation must be in this callback only. |
| **Example:** | this.view.MLKitBCScanner. addInitCompleteCallback(callback); |

1. **REVISION HISTORY**

App version 1.0.0:

## **A. Limitations**

1. Do not have support for Landscape mode.

2. This component supports only 64 bit devices.

**B. Known Issue**

NA