15 Feb 2022

Legacy Barcode/QR Code Scanner (2.0.0)

# Overview

The Legacy Barcode/QR Code Scanner component can scan and decode barcodes. Under the hood, the component contains NFIs that use the [Google Mobile Vision: Barcode API](https://developers.google.com/vision/android/barcodes-overview). The component also supports continuous scanning and automatic rendering. It works with NFI 3.0 and Temenos Quantum 9.x.

## Use case

1. An event organizer app, where a user needs to continuously scan barcodes on passes and allow entry to the attendees.

2. A calendar app that adds events to a user's calendar using the QR Code of the event

## Percentage of re-use: 90%

## Features

* The component utilizes [Google Mobile Vision's Barcode API](https://developers.google.com/vision/android/barcodes-overview)for decoding barcodes and QR codes
* It automatically parses QR Codes, Data Matrix, PDF-417 and Aztec values for following supported formats:
  + URL
  + Contact information (VCARD etc.)
  + Calendar event
  + Email
  + Phone
  + SMS
  + ISBN
  + WIFI
  + Geo-location (latitude and longitude)
  + AAMVA driver license/ID
* Updated to NFI 3.0

# Getting Started

## Prerequisites

Before you start using the Barcode/QR Code Scanner component, ensure the following:

• Temenos Fabric

• Quantum Visualizer

## Platforms Supported

### Mobile

#### iOS

#### Android

### Tablets

## Importing the app

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

## **To import the Barcode/QR code Scanner component, do the following:**

## Open your app project in Quantum Visualizer.

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, 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 Description automatically generated

Once you have imported a component to your project, you can easily add the 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 Build and Run your app section of the Quantum Visualizer User Guide.

You can then run your app to see the Barcode/QR Code Scanner work in real time.

# References

## Dynamic Usage

You can also add **Barcode/QR Code Scanner** 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 Barcode/QR Code Scanner component instance \*/

var barcodeqrscanner = new com.konympl.barcodeqrscanner(

{

id: "barcodeqrscanner",

isVisible: true,

height: "45%"

top: "0dp",

left: "0dp",

width: "100%",

layoutType: kony.flex.FREE\_FORM,

autogrowMode: kony.flex.AUTOGROW\_NONE,

skin: "slFbox",

clipBounds:true,

zIndex:1

}, {}, {});

/\*Adding the Barcode/QR Code Scanner component to a Form\*/

this.view.add(barcodeqrscanner);

/\* Displaying result after scanning barcode/qrcode \*/

this.view.barcodeqrscanner.afterScanCallback = () =>{

alert(this.view.barcodeqrscanner.getScanResult());

};

In the code snippet, you can edit the properties of the component as per your requirement.

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**. Quantum Visualizer 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 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{

implementation 'com.tbruyelle.rxpermissions2:rxpermissions:0.9.4@aar'

implementation 'io.reactivex.rxjava2:rxjava:2.1.5'

implementation 'com.google.android.gms:play-services-base:11.6.0'

implementation 'com.google.android.gms:play-services-basement:11.6.0'

implementation 'com.google.android.gms:play-services-tasks:11.6.0'

implementation 'com.google.android.gms:play-services-vision:11.6.0'

}

NOTE : 1. If you do not add the gradle entries to your project , the app crashes.

2. you can update the Play Services version in the later builds of your app.

To add NFI 3.0 in android follow the steps:

* Copy the aar/jar from the NFI zip, Delete existing Android NFI zip.
* Paste that jar/aar that has been separated in step 1, at <project>/resources/customlibs/Android folder.

**NOTE**: If customlibs folder is not present then create it.

General Properties

1. Camera Facing (cameraFacing)

|  |  |
| --- | --- |
| **Description:** | Specifies the camera that the component should use to scan codes. |
| Syntax: | cameraFacing |
| **Type:** | List Selector, String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. barcodeqrscanner. cameraFacing = "Back"; |
| **Remarks:** | The default value for the property is “Back”. |

2. Render Automatic(renderAutomatic)

|  |  |
| --- | --- |
| **Description:** | Specifies whether you want to render the scanner by default. |
| Syntax: | renderAutomatic |
| **Type:** | Boolean |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. barcodeqrscanner. renderAutomatic = true; |
| **Remarks:** | The default value for the property is “true”. |

3. Enable Continuous Scanning (enableContinuousScanning)

|  |  |
| --- | --- |
| **Description:** | Specifies whether the camera should remain open after scanning a code. |
| Syntax: | enableContinuousScanning |
| **Type:** | Boolean |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. barcodeqrscanner. enableContinuousScanning = true; |
| **Remarks:** | * If you want the camera to scan codes continuously, set the Enable Continuous Scanning property to **true**, and then use the afterScanCallback event to call the resumeScan API. * If the Enable Continuous Scanning property is **false**, then the component closes the camera after scanning a code. * The default value for the property is “true”. |

4. Close Button Visibility (closeButtonVisibility)

|  |  |
| --- | --- |
| **Description:** | Toggles the visibility of the close button of the scanner. |
| Syntax: | closeButtonVisibility |
| **Type:** | Boolean |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. barcodeqrscanner. closeButtonVisibility = true; |
| **Remarks:** | The default value for the property is true. |

5. Close Button Image (closeButtonImage)

|  |  |
| --- | --- |
| **Description:** | Specifies the image that you want to use for the close button of the scanner. |
| Syntax: | closeButtonImage |
| **Type:** | Image, String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. barcodeqrscanner. closeButtonImage = “close.png”; |
| **Remarks:** | * If you want to use a local file, make sure that the file exists in the *workspace*\*resources*\*common* directory. * The default value for the property is “close.png”. |

6. Flash Button Visibility (flashButtonVisibility)

|  |  |
| --- | --- |
| **Description:** | Toggles the visibility of the flash button of the scanner. |
| Syntax: | flashButtonVisibility |
| **Type:** | Boolean |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. barcodeqrscanner. flashButtonVisibility = true; |
| **Remarks:** | The default value for the property is true. |

7. Flash Button Image (flashButtonImage)

|  |  |
| --- | --- |
| **Description:** | Specifies the image that you want to use for the flash button of the scanner. |
| Syntax: | flashButtonImage |
| **Type:** | Image, String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. barcodeqrscanner. flashButtonImage = “flash.png”; |
| **Remarks:** | * If you want to use a local file, make sure that the file exists in the *workspace*\*resources*\*common* directory. * The default value for the property is “flash.png” |

8. Camera Toggle Visibility(cameraToggleVisibility)

|  |  |
| --- | --- |
| **Description:** | Toggles the visibility of the camera toggle button of the scanner. |
| Syntax: | cameraToggleVisibility |
| **Type:** | Image, String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. barcodeqrscanner. cameraToggleVisibility= true; |
| **Remarks:** | The default value for the property is true. |

9. Camera Toggle Image (flashButtonImage)

|  |  |
| --- | --- |
| **Description:** | Specifies the image that you want to use for the camera toggle button of the scanner. |
| Syntax: | cameraToggleImage |
| **Type:** | Image, String |
| **Read/Write:** | Read + Write |
| **Example:** | this. view. barcodeqrscanner. cameraToggleImage = “switch\_cam.png”; |
| **Remarks:** | * If you want to use a local file, make sure that the file exists in the *workspace*\*resources*\*common* directory. * The default value for the property is “switch\_cam.png” |

## B. API

### i.renderScan

|  |  |
| --- | --- |
| **Description:** | Renders the barcode scanner on the device. |
| Syntax: | renderScan () |
| **Parameters:** | None |
| **Return Value:** | None |
| **Remarks:** | Use the API to manually render the scanner when the Render Automatic property is **false**. |
| **Example:** | this. view. barcodeqrscanner . renderScan (); |

### ii.resumeScan

|  |  |
| --- | --- |
| **Description:** | Resumes the scanning process of the component. |
| Syntax: | resumeScan () |
| **Parameters:** | None |
| **Return Value:** | None |
| **Remarks:** | If you want the camera to scan codes continuously, set the Enable Continuous Scanning property to **true**, and then use the afterScanCallback event to call the resumeScan API. |
| **Example:** | this. view. barcodeqrscanner . resumeScan (); |

### iii.stopScan

|  |  |
| --- | --- |
| **Description:** | Stops the scanning process of the component. |
| Syntax: | stopScan () |
| **Parameters:** | None |
| **Return Value:** | None |
| **Example:** | this. view. barcodeqrscanner . stopScan (); |

### iv.flashControl

|  |  |
| --- | --- |
| **Description:** | Toggles the flashlight of the camera during the scanning process. |
| Syntax: | flashControl () |
| **Parameters:** | None |
| **Return Value:** | None |
| **Example:** | this. view. barcodeqrscanner . flashControl (); |

### v.getScanResult

|  |  |
| --- | --- |
| **Description:** | Used to get the value of the scanned barcode/Qr code. |
| Syntax: | getScanResult() |
| **Parameters:** | None |
| **Return Value:** | String or Object |
| **Example:** | this. view. barcodeqrscanner . getScanResult(); |

### C. Events

### 1. afterScanCallback

|  |  |
| --- | --- |
| **Description:** | Invoked after the component scans and decodes a barcode or a QR code. |
| Syntax: | afterScanCallback |
| **Parameters**: | None |
| **Remarks:** | If you want the camera to scan codes continuously, set the Enable Continuous Scanning property to **true**, and then use the afterScanCallback event to call the resumeScan API. |
| **Example:** | this.view.barcodeqrscanner.afterScanCallback = function(){  alert("Result : " + this.view.barcodeqrscanner.getScanResult());  }; |

### 2. onClickClose

|  |  |
| --- | --- |
| **Description:** | Invoked when the user taps the close button of the scanner. |
| Syntax: | onClickClose |
| **Parameters**: | None |
| **Example:** | this.view.barcodeqrscanner.onClickClose= function () {  alert(“Closing the scanner”);  }; |

### 3. errorCallback

|  |  |
| --- | --- |
| **Description:** | Invoked when an error occurs in the component. |
| Syntax: | errorCallback |
| **Parameters**: | error [JSON] : Information about the error, such as the error code and error message. |
| **Example:** | this.view.barcodeqrscanner. errorCallback = function (error) {  alert(“Error”+ error);  }; |

# Revision History

|  |  |
| --- | --- |
| Asset Version | Modifications |
| Version 2.0.0 | * Updated the component with **NFI 3.0** and included NFI code for checking flashlight in android and IOS. * Works for Temenos Quantum 9.x |
| Version 1.0 | * Initial release of asset and documentation. * Works for **NFI 3.0** in Temenos Visualizer 8.x |

App version 2.0.0:

## Limitations

1. Landscape mode is not supported.