MRZScanner Guide for React Native

SDK v.1.9.0

`To download the SDK, follow up on the link:`

https://mrzscanner.com/download

Installation guide

Please follow these simple steps to integrate our SDK into your project:

1. Add the module with typing the following command in the terminal:

   $cd Project/Dir
   $yarn add react-native-mrzscannerlib

iOS

SDK v.1.9.0

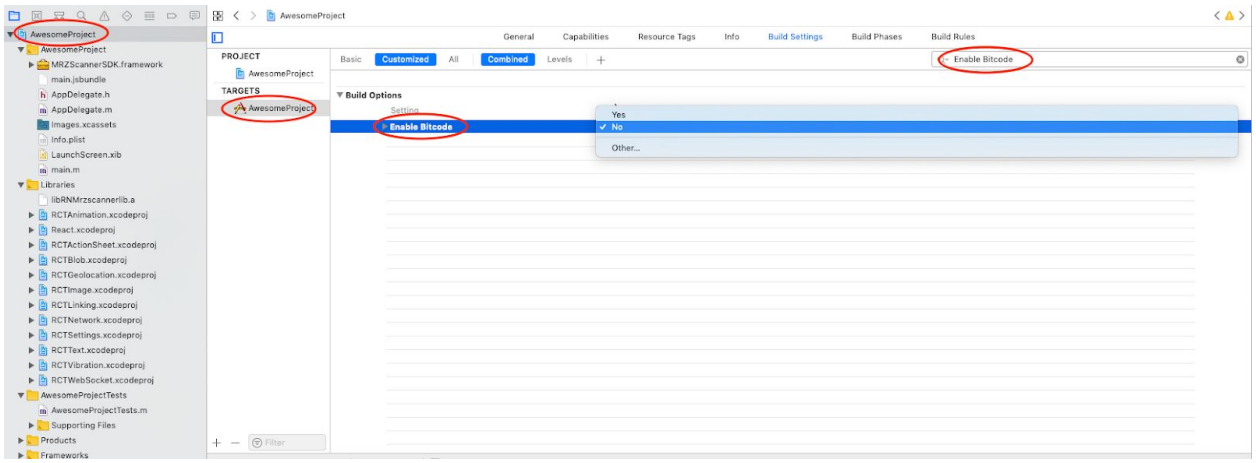
`To download the framework, follow up on the link:`

https://mrzscanner.com/download

Installation guide

---

1. Disable bitcode. Your app will immediately generate an error if you try and run with an Enable Bitcode.

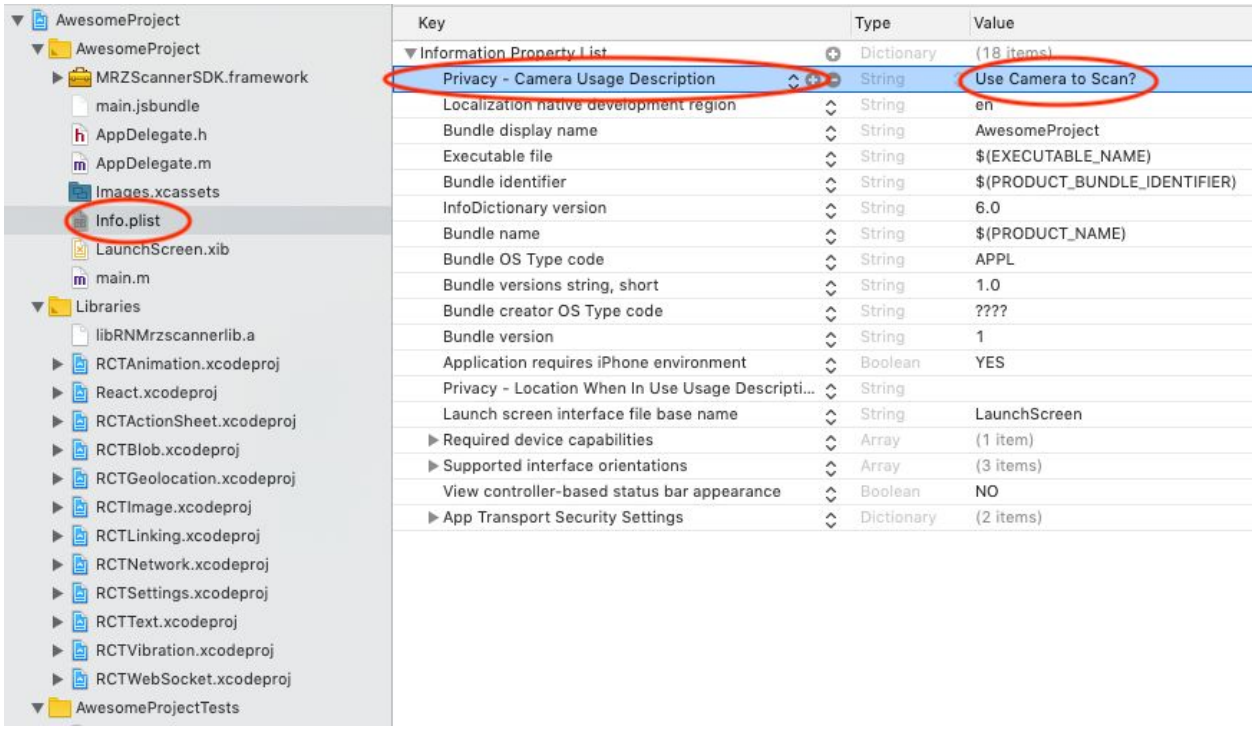


2. The scanner uses your phone camera, automatically requires a permission to access your camera.

Add the key/value pairs for camera usage description:

-key: NSCameraUsageDescription

-value: value: MRZ Scanner requires using camera



Implementation in React Native

1.In order to interact with module, first thing is to import it into your App.js file and create a reference to it:

```
import {NativeModules} from 'react-native' ;
const MrzApp= NativeModules.RNMrzScanner;
const EventEmitter = new NativeEventEmitter(MrzApp);
```

3. To receive results you need to embed an EventListener in the **componentDidMount()** life cycle method.

```
componentDidMount(){
    subscription=EventEmitter.addListener
    ('successfulScanEmittedEvent',
     (body) =>
      console.log("Results" + body.surname,
                        body.document_type_readable,
                        body.issuing_country,
                        body.surname,
                        body.document_number,
                        body.nationality,
                        body.dob_raw,
                        body.dob_readable,
                        body.sex, )
     );}

/* In this case we're using type SCANNER_TYPE_MRZ ScannerType(0),
```

the results for "body" will also include the following if found :
* - body.raw_result (raw String of the scanner MRZ)
* - body.portrait (base64 String)
* - body.signature (base64 String)
* - body.full_image (base64 String)
*/

Results are sent as an event by the native module with the following keys:

- 'successfulScanEmittedEvent' - for a type 1 (mrz) scan;
- 'successfulDocumentScanEmittedEvent' - for type 2 & 3 (document image) scan ;
- ‘scannerWasDismissedEmittedEvent’ - for dismissed scanner
- ’permissionsWereDeniedEmittedEvent’ - for denied permission
- ‘scanImageFailedEmittedEvent’- for a failed image scan

- JavaScript can then receive the registered event by “add ListenerOn” using the “Subscribe” mixin.

(Use of the eventEmitter can be found in the provided sample project ”Sample.js”.)

To get access to the new functionalities from JavaScript you have to pull them off of NativeModules each time :

```
const MrzScanner= NativeModules.RNMrzScanner;

MrzScanner.startScanner()
```

API Methods

```
*Activates the scanner, with a default SCANNER_TYPE_MRZ if not specified.
MrzScanner.startScanner()
```

```
* Specify which scanner type you want to use. There are two options: "MRZ Scanner" and "Document Image scanner".
* The "MRZ Scanner" option is used to scan for MRZ.
* The "Document image scanner" is used for capturing front and back image of the ID documents.
* @param int [1: SCANNER_TYPE_MRZ,  2: SCANNER_TYPE_DOC_IMAGE_ID, 3 : SCANNER_TYPE_DOC_IMAGE_PASSPORT]. Default value is SCANNER_TYPE_MRZ
MrzScanner.setScannerType(ScannerType  int)
```

```
* Resume scanning after the scanner has been paused/stopped. Usually after a successful scan.
MrzScanner.resumeScanning()
```

```
* @return the currently set date format in which the parsed dates are formatted.
* Set the date format in which the parsed dates are formatted.
* @param dateFormat the pattern describing the date format. Example: "dd.MM.yyyy"
MrzScanner.setDateFormat(String dateFormat)
```

```
* @return the current SDK Version.
MrzScanner.sdkVersion(): String
```

```
* Specify whether the scanner should detect and return result for IDs.
* @param isIDActive [true, false]. The default value is true.
MrzScanner.setIDActive(isIDActive: Boolean)
```

```
 * Specify whether the scanner should detect and return result for passports.
* @param isPassportActive [true, false]. The default value is true.
MrzScanner.setPassportActive(isPassportActive: Boolean)
```

```
* Specify whether the scanner should detect and return result for visas.
* @param isVisaActive [true, false]. Default value is true.
MrzScanner.setVisaActive(isVisaActive: Boolean)
```

```
* Specify the maximum number of CPU threads that the scanner can use during the
scanning process.
* @param maxThreads number of CPU threads. Default value is 2.
MrzScanner.setMaxThreads(maxThreads: Int)
```

```
* Register with the licence key provided to remove the asterisks (*) from the result.
* @param key     the provided licence key.
```

```
* @return 0 for success, -1 if registration failed.
MrzScanner.registerWithLicenseKey(key: String): Int
```

```
* Trigger an image picker.
MrzScanner.scanFromGallery()
```

```
*Costumizes the overlay of the finder in the scanner
*@param base64String takes a base64 type string
MrzScanner.startScannerWithCustomOverlay(base64String : String)
```

```
*@Trigger a flashlight
MrzScanner.toggleFlash(active : Boolean)
```

```
* Initiates the scanner view with continuous scanning enabled.
* After a successful scan, the scanner will not be dismissed or
paused.
* To close the scanner call closeScanner().
* See setIgnoreDuplicatesEnabled for further continuous scanning
behavior changes.
*
* @param boolean: activeed [true, false] Default value is false.
MrzScanner.setContinuousScanningEnabled(boolean : activated)
```

```
*Ignores duplicate is used to specify whether the scanner should
ignore the successful scan if the document number is identical to the
previous successful scan.
* @param boolean: activated [true, false] Default value is false.
MrzScanner.setIgnoreDuplicatesEnabled(boolean : activated)
```

```
* Enables vibrate on successful scan option.
* @param boolean: activated [true, false] Default value is false.
MrzScanner.setVibrateOnSuccessfulScan(boolean : activated)
```

```
* Set the scanning rectangle to limit the scanning area. The
parameters' values are representing percentages of the scanning
preview.

* @param x the top-left point of the scanning rectangle. [0,...,100]
Default value: 5.

* @param y the top-left point of the scanning rectangle. [0,...,100]
Default value: 20.

* @param x the top-left point of the scanning rectangle. [0,...,100]
Default value: 5.

* @param x the top-left point of the scanning rectangle. [0,...,100]
Default value: 5.
MrzScanner.startPartialViewScanner(int : x, int : y, int :
width, int : height)
```

Sample code is included with the download file.

If you have any questions or concerns please visit our site at www.mrzscanner.com where you can register and send us a support request ticket at our Developer Portal, or feel free to contact us at contact@mrzscanner.com.