i-nigma
Symbol Reader Developer Suite For Android

Version 4.02

Developer's Guide

Written by Adi Gadish, 3GVision
Last update: Apr 25, 2018

Contents
Symbol Reader Suite Developer's Guide

Symbol Reader Software Package

General

This document describes a software package ("**i-nigmaSDK**") that performs image processing and information decoding from images that contain encoded symbols, such as QR Code and Data Matrix 2D codes, as well as various 1D barcode types.

The package allows an application developer to incorporate Code Reading Data Acquisition into their mobile applications.
The current **i-nigma SDK** primarily targets Android Studio users, but can be used also in other development environments, and even in legacy environments like Eclipse.

The main components of the **i-nigmaSDK.aar** package are

- **classes.jar**, which is a combination of Java files belonging to the package **com.threegvision.products.inigma_sdk_pro.sdk_pro**
- the jni directory, that includes the native Android shared library **libi-nigma_core.so** compiled to various processors: **armeabi-v7a, x86, arm64-v8a** and **x86_64**

The **i-nigmaSDK** package handles & performs:

- Camera activation for capturing live streaming video and displaying it on screen.
- 2D/1D Symbol decoding by analyzing the visual information captured by the camera.
- Managing Over-the-Air (OTA) license acquisition to get and update a specific license for a single mobile unit (The connection is to the i-nigma server – "**i-nigma License Manager**")

The **i-nigma SDK** is packaged as part of a reference application, that shows how to integrate and link with **i-nigmaSDK.aar**.

Usage

In order to use the i-nigma SDK in Android Studio

- Open the project where you want integrate the **i-nigma SDK**
- Select the menu item **File | New | New Module…**
- In the **New Module** dialog that opens, select **Import .JAR/.AAR Package** and click **Next**
- In the File name field, enter the location of the **i-nigmaSDK.aar** file (you can locate it also using the "**…"** button). You can type any name in the

Subproject name field (In this example we will use **i-nigmaSDK**). Click **Finish**

- You will see a new subproject that includes **i-nigmaSDK.aar** as well as automatically generated **build.gradle** and **.iml** files
- Add the following lines to your project's **.gradle** file

```
dependencies {
    compile project(':i-nigmaSDK')
}
```

- In the Java files that need to use the **i-nigma SDK**, add the line

```
import com.threegvision.products.inigma_sdk_pro.sdk_pro.*;
```

and you can use the functionality described below

License Acquisition

The **i-nigma SDK** needs a valid license in order to operate. The license is kept as a file, and can be acquired Over-the-Air using the data link supplied by the device. The device should be able to connect to the Internet at least once in order for the **i-nigma SDK** to communicate with the i-nigma License Manager system. The license is then stored on the device to allow future operation with no need for re-acquisition.

The only information that is transferred to the server is:

- Device model
- Android ID

The **i-nigma SDK** provides an API for cases where an explicit re-acquisition of the license is needed, e.g. in cases where new features are provided by a new type of license. Other than that, the license acquisition process is done internally during the code reading process.

Please note that if a license is not acquired, is not valid, or has expired, the **i-nigma SDK** can be used for a temporarily period called "Grace Period".

Additions to the Manifest

The following permissions have to be added to your application permission (in your app manifest):

```
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-feature android:name="android.hardware.camera" />
<uses-feature android:name="android.hardware.camera.autofocus" />
```

Files, Classes and Methods

File name: "SDK.java"
Package: "com.threegvision.products.inigma_sdk_pro.sdk_pro"

1. Class name: SDK

This is the main class of the SDK. Your main Activity should create one instance of this class. As this class is performing asynchronous actions, your Activity should provide a reference for a listener that will be used by the SDK to report bar code decoding results or any error occurrences.

Methods:

- **public SDK(final Activity activity, Observer observer)**
  - Call this constructor to instantiate an **SDK** class.
  - ***Activity activity***: The originating Activity calling the SDK
  - ***Observer observer***: The object that will implement the "Observer" interface (see below for more details) and will receive all messages returning from the SDK.

- **public void Close()**
  - This method stops, closes and frees all SDK variables & memory. Call this function to completely destruct the SDK.

- **public boolean IsScanning()**
  - This method returns true if a scanning process is currently taking place.

- **public ViewGroup StartScanning (final Rect rect, final int DecodingFlags, final int FlashMode, final int TimeoutInSeconds)**
  - Call this method to initiate an asynchronous scanning session. Upon this call (if successful), a new ViewGroup containing the camera viewfinder will return. The camera will be started and processing will begin. The viewfinder will be voided automatically if bar code is detected or on timeout.
  - The application should display this ViewGroup in order to display the camera viewfinder. The application can modify or add its own views to custom the viewfinder look & feel according to its needs. As an example, see how the reference app shows aim on top of the camera viewfinder.
  - The camera viewfinder will use the rectangle values specified by the user for its position & size. Bear in mind that the device is only capable of showing its sensor aspect ratio; therefore in order

to prevent stretch, you must keep the device aspect ratio when specifying the rectangle.
    - Use (bitwise or) the following flags to determine which kind of barcodes are supposed to be detected (in complying with your license):
        - **Decode_EAN8** – EAN8.
        - **Decode_EAN13** – EAN13.
        - **Decode_EAN128** – EAN128.
        - **Decode_QR** – Quick Response (QR).
        - **Decode_DataMatrix** – Data Matrix.
        - **Decode_EAN39** – EAN39.
        - **Decode_PDF417** – PDF 417.
        - **Decode_NW7** – NW7 (CODABAR).
        - **Decode_I2OF5** – I 2 OF 5.
        - **Decode_GS1** – GS1.
        - **Decode_MICRO_QR** – Micro QR.
        - **Decode_BlackOnWhite** – Reverse printing.
    - Select the following values for the Flash mode:
        - **Flash_Off** – Do not use the device flash at all.
        - **Flash_On** – Turn on the device flash (Torch).
        - **Flash_Auto** – Allow the device to use its flash automatically.
    - Specify the scanning timeout in seconds to make the sdk stop if barcode was not found within that period.

- **public void Stop()**
    - This method closes camera and stops barcode decoding immediately. It also abort the license acquisition process (if was on going).

- **public void AutoFocus()**
    - Triggers an automatic focus cycle. Please note that an auto focus cycle is triggered periodically every 5 seconds.

- **public int GetMaxZoom()**
    - Returns the camera maximum zoom value, if applicable. Values are multiply by 10, thus a value of 30 means max zoom level of 3.

- **public void Zoom(int zoom)**
    - Sets the camera zoom level, if applicable. Zoom level sets to the value of the zoom parameter divided by 10, thus for zoom level of 2, you should call this method with a value of 20.

- **public void AcquireLicense()**
    - This method launch a license acquisition process. Application should call this function if no license is acquired.

- **public void RevokeLicense()**
  - This method causes the license to revoke. License would need to be renew before next scan.

- **public boolean IsLicenseValid()**
  - Returns true if license was acquired and is currently valid.

- **public boolean IsLicenseTemporarilyValid()**
  - Returns true if license was not acquired or not valid, but temporarily SDK can be used ("Grace Period").

- **public int Version()**
  - Returns the version number of the SDK.

- **public void SetParameters(DecodeParameters param, int value)**
  - Call this method in order to set specific decoding parameters. The first parameter to this function defines which decoding parameter is to be set, while the second one is it value.
    The following parameters are being supported:
    i. **TWO_OF_5_CHECKSUM_DIGIT** – Enable / disable the checksum test for the I2OF5 codes. Set '1' to enable, '0' to disable (default).
    ii. **TWO_OF_5_MIN_CODELENGTH** – Specify the minimum expected length of the I2OF5 code. For unlimited minimum length, specify 0 (default).
    iii. **TWO_OF_5_MAX_CODELENGTH** – Specify the maximum expected length of the I2OF5 code. For unlimited maximum length, specify 0 (default).

2. Interface name: SDK.Observer

This interface must be implemented by your activity class. Scanning results as well as error indications would be propagating by this interface.

Methods:

a. void OnDecode(final int type, final int mode, final byte[] content)

The method is called asynchronously on a successful bar code decoding. The type is reported by the type parameter, mode is indicated by the mode parameter, while content, as encoded into the bar code, is returned via the content string.
The following barcode types are defined:

i. **Type_Unknown** – Barcode of Unknown type.
ii. **Type_EAN8** – EAN8.
iii. **Type_EAN13** – EAN13.
iv. **Type_EAN128** – EAN128.
v. **Type_QR** – Quick Response (QR).

vi. **Type_DataMatrix** – Data Matrix.
vii. **Type_EAN39** – EAN39.
viii. **Type_PDF417** – PDF 417.
ix. **Type_NW7** – NW7 (CODABAR).
x. **Type_GS1** – GS1.
xi. **Type_MICRO_QR** – Micro QR.
xii. **Type_I2OF5** – I2OF5.

The following barcode modes are defined:

i. **Mode_NOFUNC** – Normal barcode mode.
ii. **Mode_FUNC1** – func1 mode.
iii. **Mode_FUNC2** – func2 mode.

b. void OnError(final int code)

The method is called asynchronously on any error occurred while scanning or after a timeout. The appropriate error code is indicated by the code integer.

The following error codes are defined:

i. **Error_NoError** – a special code returned as a result of calling the Test method, if server is ok.
ii. **Error_CameraCanNotBeOpened** – camera could not be operated, probably is used by another application, or some other error such as low battery etc.
iii. **Error_ScaningTimeouted** – scanning timed out w/o and decoding success
iv. **Error_LicenseNotAcquired** – no license found
v. **Error_SecurityError** – an error occurred while communicating to the sdk is most cases due to insufficient permissions of the Activity or user deny communication.
vi. **Error_GeneralError** – unspecified error.
vii. **Error_APIError** – error caused by invalid sequence or parameters specified to the SDK class.

The reference application

This zip file contains the reference application that demonstrates how to operate the **i-nigma SDK**, and can be used as a template for your application. Please note you have full control over the GUI (i.e. you can set the title by modifying the string "*app_name"* in string.xml, you can add views as you like, you can control the menu, the background, sounds, etc).