

# Slyce Android SDK

Version 4.5.9

Last updated: October 17,2017

## Contents

GETTING STARTED.....	3
Overview.....	3
Prerequisites.....	3
Setup.....	3
Event reporting.....	4
CODE INTEGRATION.....	5
Migrating from SDK 1.x to 2.x.....	13
Migrating from SDK 2.1 to 2.2.....	15
Migrating from SDK 2.2.x to 2.3.x.....	15
Migrating from SDK 2.3.x to 2.4.x.....	15
Migrating from SDK 2.5.x to 2.6.x.....	15
Migrating from SDK 2.6.x to 2.7.x.....	16

# GETTING STARTED

## Overview

The Slyce Android SDK enables Android developers to easily interact with the Slyce image recognition platform.

The SDK provides the methods required to submit images and receive results.

## Prerequisites

- Minimum Android OS versions 4.0 (API level 14) and higher.
- Android Studio development environment
- Slyce credentials:
  - Slyce client ID for premium users
  - APP KEY and APP SECRET pair for public users

## Setup

1. Create libs folder and place the Slyce AAR file inside.

2. Add at build.gradle:

```
repositories{  
    flatDir {  
        dirs 'libs'  
    }  
}
```

3. Add at build.gradle dependencies (in your application module)

```
compile(name:'slyce', ext:'aar')  
compile 'com.google.android.gms:play-services-vision:8.4.0'
```

4. It's important to initialize the Slyce object and call the `Slyce.open(---)` method in the extended Application class or in the main Activity of the application in order to sync the data as early as possible.

## Event reporting

Slyce provides a built-in event tracking system for collecting and aggregating points of interest. Such system allows adding context to in-app customers behaviour by connecting it to the backend e-commerce system. In order to send a custom event please use a method of an initialised SFSlyce Instance:

```
public void reportStatistics(final String eventName, final Map<String, String> additionalParams)
```

This method also allows you to send custom parameters in a form of NSDictionary which can contain only NSStrings as keys and values.

# CODE INTEGRATION

## SlyceRequest

**// Implement OnSlyceRequestListener:**

```
public class MainActivity extends Activity implements OnSlyceRequestListener {

//standard
    @Override
    public void onSlyceProgress(final long progress, final String message, String token) {
        // progress - progress percentage
        // message - progress message
        // requestToken - request unique id}

//standard
    @Override
    public void onProgressExt (final String progress) {
        // progress -String containing the progress extended info.
    }

//premium
    @Override
    public void onImageDetected(String productInfo) {
        // productInfo - representing a short info about the matched 2D products}

//premium
    @Override
    public void onImageInfoReceived(JSONArray products) {
        // products - representing the additional info}

//standard
    @Override
    public void onBarcodeDetected(SlyceBarcode barcode) {}

//standard
    @Override
    public void onResultsReceived(final JSONObject products) {
        // products - founds products (might be empty if no products found)
    }

//standard
    @Override
    public void onResultsReceivedExt(String result) {
        // result - extended result of the detected image
    }

//standard
    @Override
    public void onError(final String message) {
        // message - error message }

//premium
    @Override
    public void onSlyceRequestStage(SlyceRequestStage message) {
        // message - of type StageMessage (enum) indicates stage has been completed.
        // For example: this call back will be invoked after a bitmap has been uploaded to the}

//premium
    @Override
    public void onBarcodeInfoReceived(JSONObject products) {
```

```

        // }

//standard
@Override
public void onFinish() {}
}
...
}

// Create Slyce singleton object:
Slyce slyce = Slyce.getInstance(this);

// Initiate Slyce SDK with OnSlyceOpenListener

// For premium users:

slyce.open("YOUR_CLIENT_ID", new OnSlyceOpenListener() {

    @Override
    public void onSuccess() {}

    @Override
    public void onOpenFail(String message) {}

});

// For public users:

slyce.open("APP_KEY", "APP_SECRET", new OnSlyceOpenListener() {

    @Override
    public void onSuccess() {}

    @Override
    public void onOpenFail(String message) {}

});

// Create SlyceRequest object for searching products by image or by image url

SlyceRequest request = new SlyceRequest(slyce, this, new JSONObject());

// Searching products by image url
String imageUrl = "http://...";
request.getResults (imageUrl);

// Searching products by image (Bitmap)
Bitmap bitmap;
request. getResults (bitmap);

// Cancelling SlyceRequest
request.cancel();

```

## // SlyceCamera:

- \* Scanning products/barcodes/QR codes.
- \* Managing the camera and displaying its preview

## Create a CameraActivity and Implement OnSlyceCameraListener:

```
public class CameraActivity extends Activity implements OnSlyceCameraListener {
```

### //standard

```
@Override  
public void onCameraBarcodeDetected(SlyceBarcode barcode) {  
    // Called when barcode is found}
```

### //premium

```
@Override  
public void onCameraImageDetected(String productInfo) {  
    // Called when 2D products are found}
```

### //premium

```
@Override  
public void onCameraImageInfoReceived(JSONArray products) {  
    // Called when additional info for the previously recognized 2D product is found.}
```

### //standard

```
@Override  
public void onCameraSlyceProgress(long progress, String message, String token) {  
    // Reporting a numeric value and informative message.}
```

### //standard

```
@Override  
public void onProgressExt (final String progress) {  
    // progress -String containing the progress extended info.  
}
```

### //premium

```
@Override  
public void onCameraSlyceRequestStage(SlyceRequestStage message) {  
    // Reporting the stage currently being processed.}
```

### //standard

```
@Override  
public void onCameraResultsReceived(JSONObject products) {  
    // Called when 3D products are found}
```

### //standard

```
@Override  
public void onResultsReceivedExt(String result) {  
    // result - extended result of the detected image  
}
```

### //standard

```
@Override  
public void onSlyceCameraError(String message) {  
    // Called when an error occurred}
```

### //standard

```
@Override  
public void onTap(float x, float y) {  
    // Called when the camera was touched in a specific point.}
```

```
//standard
@Override
public void onSnap(Bitmap bitmap) {
    // Called when the snapped bitmap is ready after SlyceCamera.snap() was invoked}
```

```
//premium
@Override
public void onCameraBarcodeInfoReceived(JSONObject products) {
    // Called when additional info for the previously recognised barcode is found.}
```

```
//standard
@Override
public void onCameraPreviewMode(boolean front) {
    // Called when camera initiate or when calling SlyceCamera.flipCamera() method}
```

```
//standard
@Override
public void onCameraFinished() {
    // Called when Slyce search process ended}
```

**// Create and initiate Slyce single object as mentioned earlier**

**// UI:**

The SlyceCamera constructor expects an empty SurfaceView, it will take care of displaying the camera preview.

SurfaceView should be added to the Activity xml file.

Create activity\_camera.xml

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent" android:layout_height="match_parent">

    <SurfaceView
        android:id="@+id/preview"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</RelativeLayout>
```

Please add android:configChanges to the CameraActivity at your manifest.xml

```
<activity
    android:name=".CameraActivity"
    android:configChanges="orientation|screenSize"
</activity>
```

Now you can create the SlyceCamera object, it requires:

- \* Parent Activity
- \* Opened Slyce object
- \* The surface
- \* Options - optional
- \* OnSlyceCameraListener for notifying results
- \* Optional - customBarcodeFormat for a set of custom barcode formats



### // Create the SlyceCamera object

```
slyceCamera = new SlyceCamera(this, Slyce.getInstance(this), preview, null, this);
```

### // create SlyceCamera object with custom barcode format set

```
//int barcodeFormat = Barcode.EAN_13+Barcode.EAN_8;// customize barcode detection, the default is detection of all formats.
```

```
//slyceCamera = new SlyceCamera(this, Slyce.getInstance(this), preview, null, this,barcodeFormat);
```

### // Customize the next parameters if needed:

```
//slyceCamera.shouldPauseScanner(false);           //pause the detection after a successful scan, the default is true
```

```
//slyceCamera.setShouldPauseScannerDelayTime(5000); // set a custom time in milliseconds for resuming auto scanning after a successful scan,the default is 3000
```

```
//slyceCamera.setContinuousRecognition(false);      // disable/enable continuous recognition ,the default is true
```

```
//slyceCamera.setContinuousRecognition2D(false);    // disable/enable 2D continuous recognition the default is true
```

```
//slyceCamera.setContinuousRecognitionBarcodes(false); // disable/enable Barcode continuous recognition the default is true
```

You need to handle the life cycle of SlyceCamera:

```
@Override
protected void onResume() {
    super.onResume();
    slyceCamera.start();
}
```

```
@Override
protected void onPause() {
    super.onPause();
    slyceCamera.stop();
}
```

Now you can start scanning images/barcodes

## SlyceCameraFragment

Full UI implementation of SlyceCamera.

Create FullUIActivity and its xml file activity\_full\_ui.xml

Please add android:configChanges to the CameraActivity at your manifest.xml

```
<activity
...
    android:configChanges="orientation|screenSize"
...
</activity>
```

Please add a container for SlyceCameraFragment at activity\_full\_ui.xml

```
<FrameLayout
    android:id="@+id/slyce_camera_fragment_container"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</FrameLayout>
```

Adding SlyceCameraFragment after Slyce SDK successfully opened.

**// For premium users:**

```
Slyce slyce = Slyce.getInstance(activity);

slyce.open(clientID, new OnSlyceOpenListener() {
    @Override
    public void onOpenSuccess() {
        // Add SlyceCameraFragment to the FullUIActivity}

    @Override
    public void onOpenFail(String message) {

    }
});
```

**// For public users:**

```
slyce.open("APP_KEY","APP_SECRET", new OnSlyceOpenListener() {

    @Override
    public void onOpenSuccess() {
        // Add SlyceCameraFragment to the FullUIActivity}

    @Override
    public void onOpenFail(String message) {}

})
```

### Add SlyceCameraFragment to the FullUIActivity

SlyceCameraFragment.newInstance() expects 3 or 5 or 6 or 7 parameters in this order (3 factory methods in total):

1. JsonObject Options - optional (can be null)
2. boolean - enabling/disabling the scanner
3. boolean - pause/resume the automatic 2D image/barcode scanner after 2D image/barcode detection.
4. boolean - pause/resume the automatic 2D image scanner after 2D image detection.
5. int - set a custom delay time in milliseconds after each detection and resume automatic scanner (the default is 3000).

6. Int – set of custom barcode formats, the default is "0" – detection of all available formats (example – Barcode.EAN\_13+Barcode.EAN\_8)

```
// SlyceCameraFragment slyceFragment = SlyceCameraFragment.newInstance(null, true, true, false, false);  
// SlyceCameraFragment slyceFragment = SlyceCameraFragment.newInstance(null, true, true, false, false, 5000);  
// SlyceCameraFragment slyceFragment = SlyceCameraFragment.newInstance(null, true, true, false, false, 5000, 0);
```

#### Additional available customizations:

//All customized fragments must extend BaseDialogFragment class and implement onDismiss method (example provided in the demo application)

```
//customize the color of the circular progress  
//slyceFragment.setCircularProgressColor(Color.GREEN);  
  
//replace the default "Scanning Tips" screen with a custom fragment (BaseDialogFragment)  
//  
slyceFragment.setCustomHelpScreen(getCustomDialogScreen(CustomHelpScreen.SCAN_TIPS_DIALOG));  
  
//replace the default "not found" screen with a custom fragment (BaseDialogFragment)  
//  
slyceFragment.setCustomNotFoundScreen(getCustomDialogScreen(CustomHelpScreen.NOT_FOUND_DIALOG));  
  
//add custom buttons with custom fragments dialogs with a custom fragment (BaseDialogFragment), Position of the custom button are in percent, when 100% is of the screen(both for x and y). //  
slyceFragment.addCustomScreenWithButton(getCustomDialogScreen(CustomHelpScreen.GENERAL_DIALOG), 50.9f, 50f, R.drawable.slyce_flash, this);
```

```
FragmentTransaction transaction = getFragmentManager().beginTransaction();  
transaction.replace(R.id.slyce_fragment_container, slyceFragment);  
transaction.addToBackStack(null);  
transaction.commit();
```

It's important to add SlyceCameraFragment to your Activity BackStack

#### **OnSlyceCameraFragmentListener**

In order to receive results please implement OnSlyceCameraFragmentListener at your Activity. Please note its a must!

```
public class SlyceActivity extends Activity implements OnSlyceCameraFragmentListener {
```

```
//standard
```

```
@Override
```

```
public void onCameraFragmentBarcodeDetected(SlyceBarcode slyceBarcode) {  
    // Called when barcode is found}
```

```
//standard
```

```

@Override
public void onCameraFragmentImageDetected(String info) {
    // Called when 2D products are found}

//premium
@Override
public void onCameraFragmentImageInfoReceived(JSONArray products) {
    // Called when additional info for the previously recognised 2D product is found.}

//standard
@Override
public void onCameraFragmentResultsReceived(JSONObject results) {
    // Called when 3D products are found}

//standard
@Override
public void onResultsReceivedExt(String result) {
    // result - extended result of the detected image
}

//standard
@Override
public void onCameraFragmentError(String message) {
    // Called when an error occurred}

//premium
@Override
public void onCameraFragmentBarcodeInfoReceived(JSONObject products) {
    // Called when additional info for the previously recognised barcode is found. }

//standard
@Override
public void onCameraFragmentFinished() {
    // Called when Slyce search process ended}
}
...
}

```

- execute can be called only once per SlyceProductsRequest
- please note that any call to execute should be triggered after Slyce SDK was successfully opened (initialised).
- requestToken is a unique identifier per a request
- canceled request cannot be resumed

## Migrating from SDK 1.x to 2.x

### **SlyceRequest:**

**SlyceProductsRequest** changed to **SlyceRequest** and it has only one constructor now.

### **OnSlyceRequestListener methods**

- \* Changed:
  - \* on2DRecognition changed to onImageDetected
  - \* on3DRecognition changed to onResultsReceived
  - \* onStageLevelFinish changed to onSlyceRequestStage
- \* Added:
  - \* onBarcodeDetected
  - \* onImageInfoReceived
  - \* onFinished
  - \* onItemDescriptionReceived

### **New Methods:**

getProducts(Bitmap), getProducts(String) for getting a list of products.

getItemDescription(Bitmap), getItemDescription(String) for getting a keywords description of the given image (bitmap/url)

**Example code:** getting products with image url

```
SlyceRequest request = new SlyceRequest(slyce, this, new JSONObject());  
request.getProducts(image_url);
```

**Permissions** - no need to add app permissions at the AndroidManifest.xml

### **Slyce singleton:**

Slyce.getInstance(Context context) takes only one parameter now.

“ClientID” should be passed now to Slyce.open(...) method.

Example:

**// For premium users:**

```
Slyce slyce = Slyce.getInstance(this);

slyce.open(clientId, new OnSlyceOpenListener() {

    @Override
    public void onOpenSuccess() {

    }

    @Override
    public void onOpenFail(String message) {

    }

});
```

**// For public users:**

```
slyce.open("APP_KEY","APP_ID", new OnSlyceOpenListener() {

    @Override
    public void onOpenSuccess() {}

    @Override
    public void onOpenFail(String message) {}

});
```

## Migrating from SDK 2.1 to 2.2

### OnSlyceRequestListener methods

- \* Added:
  - \* onBarcodeInfoReceived

### OnSlyceCameraListener methods

- \* Added:
  - \* onCameraBarcodeInfoReceived
  - \* onCameraPreviewMode

### OnSlyceCameraFragmentListener methods

- \* Added:
  - \* onCameraFragmentBarcodeInfoReceived

## Migrating from SDK 2.2.x to 2.3.x

Add at build.gradle dependency to play-services-vision library(in your application module)

compile 'com.google.android.gms:play-services-vision:8.3.0'

## Migrating from SDK 2.3.x to 2.4.x

- **Methods added to SlyceCamera class**
  - setShouldPauseScannerDelayTime
  - setContinuousRecognition2D
  - setContinuousRecognitionBarcodes

## Migrating from SDK 2.5.x to 2.6.x

- CompileSdkVersion and targetSdkVersion changed to marshmallow (23)
- Gradle build tools should be changed to 2.0.0 and forward in the main Build.Gradle file
  - classpath 'com.android.tools.build:gradle:2.0.0'
- Marshmallow Permissions handling added, one should catch denied permissions in the Activity that implements the SlyceCamera view as in the example below:

```
• @Override
  public void onRequestPermissionsResult(int requestCode,
                                         String permissions[], int[] grantResults) {
    switch (requestCode) {
      case SlyceCamera.MY_PERMISSIONS_REQUEST_CAMERA: {
        // If request is cancelled, the result arrays are empty.
        if (grantResults.length > 0
            && grantResults[0] == PackageManager.PERMISSION_GRANTED) {

          // permission was granted, no need for action

        } else {

          // permission denied
          Log.e(TAG, "Camera permission denied");

          showMessageOKCancel("You need to allow access to Camera",
                              new DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog, int which)
```

```

{
    requestPermissions(new String[]
{Manifest.permission.CAMERA},
SlyceCamera.MY_PERMISSIONS_REQUEST_CAMERA);
    }
    });

    }
    return;
}

case SlyceCamera.MY_PERMISSIONS_REQUEST_WRITE_EXTERNAL_DATA: {
    // If request is cancelled, the result arrays are empty.
    if (grantResults.length > 0
        && grantResults[0] == PackageManager.PERMISSION_GRANTED) {

        // permission was granted, no need for action

    } else {

        // permission denied
        Log.e(TAG, "Camera permission denied");

        showMessageOKCancel("You need to allow access to Storage",
            new DialogInterface.OnClickListener() {
                @Override
                public void onClick(DialogInterface dialog, int which)
        {
            requestPermissions(new String[]
{Manifest.permission.WRITE_EXTERNAL_STORAGE},
SlyceCamera.MY_PERMISSIONS_REQUEST_WRITE_EXTERNAL_DATA);
        }
        });

    }
    return;
}

}

}

private void showMessageOKCancel(String message, DialogInterface.OnClickListener
okListener) {
    new AlertDialog.Builder(CameraActivity.this)
        .setMessage(message)
        .setPositiveButton("OK", okListener)
        .setNegativeButton("Cancel", null)
        .create()
        .show();
}
}

```

○

## Migrating from SDK 2.6.x to 2.7.x

Two callbacks added and needs to be implemented in the modes of the Slyce Requests: `onProgressExt(String)` and `onResultsReceivedExt(String)`.