Record360 Android SDK integration

Table of Contents

Preparation

In order to utilize the SDK framework, an account with Record360 is required. Please contact sales@record360.com for details.

Integrating Record360

1. The Record360 SDK can be installed using gradle dependencies and maven; it simplifies the process of integrating 3rd-party libraries into your projects.

Add the Record360 SDK by adding the following code snippet to your project build.gradle file:

```
buildscript {
    repositories {
          jcenter()
          google()
    }
}

allprojects {
    repositories {
          jcenter() // default
          maven { url 'http://dl.bintray.com/record360/maven'}
          maven { url 'http://maven.microblink.com' }
          maven { url 'http://maven.google.com' }
    }
}
```

Add the dependency to the build.gradle of the module you would like to import the SDK in to.

```
dependencies {
    compile 'com.record360.sdk:android-sdk:1.5.1'
}
```

Press the gradle sync button to import the SDK dependencies.

2. Our SDK is only available on ARM devices. Add abiFilters to module build.gradle file.

```
android {
   defaultConfig {
      ndk {
         abiFilters "armeabi-v7a"
      }
   }
}
```

3. In order to properly build the application, MultiDex must be enabled in your android project.

Create a new java class in your package directory (e.g. app/java/com.example.sample/). Also, statically initialize compat vectors for resources.

```
public class SampleMultiDexApplication extends MultiDexApplication {
    static {
        AppCompatDelegate.setCompatVectorFromResourcesEnabled(true);
    }
}
```

Next add the newly created application class name to your AndroidManifest.xml in the application tag.

```
<application
   android:name=".SampleMultiDexApplication"
   ...
</application>
```

Finally, add the multidex flag to the defaultConfig in your app module build.gradle file.

```
Android {
   defaultConfig {
         multiDexEnabled true
         ...
   }
}
```

4. Initialize the Record360SDK. It can be initialized inside of the SampleMultidexApplication we created earlier or within an Activity.

```
@Override
public void onCreate() {
    Record360SDK.initialize(this, settings);
}
```

Note: The Record360SDK.Setting class is available in SDK and its usage is explained below.

5. Create an Activity that extends Record360Activity, this will give you access to commands needed to start a Record360 Session. Also, designate a Record360Interface that will handle transaction upload events. Note that the upload process is asynchronous, so the callbacks will not be called immediately after the process completes.

```
public class MainActivity extends Record360Activity implements
Record360Activity.Record360Interface
```

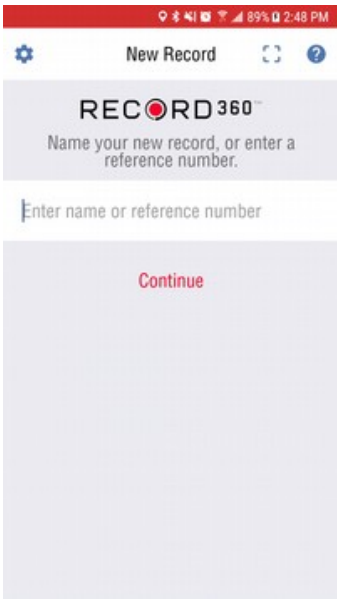
The session information will be sent from the SDK to the Record360Interface you supply as a parameter to the start functions.

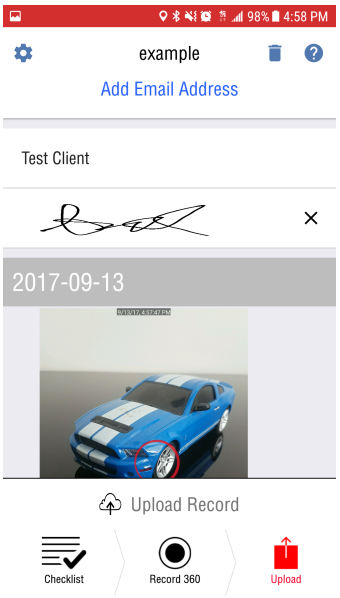
6. Pass in login credentials to this activity:

```
String username = "testuser@record360.com"
String password = "P@ssword!"

authenticatedStart(this, username, password, record360Interface);
```

7. Proceed through the Record360 application flow





8. After the transaction has finished or is canceled by the user, one of the Record360Interface callback methods will be called.

```
public void onTransactionComplete(String referenceNumber);
public void onTransactionCanceled(String referenceNumber);
```

9. When the Record360SDK has finished uploading a transaction, one of the Record360Interface methods will be called.

```
public void onTransactionUploaded(String referenceNumber);
public void onTransactionUploadFailed(String referenceNumber);

// You can also receive updates on upload progress
public void onTransactionUploadProgress(String refNum, long complete, long
total);
```

Build Time Optimization

Note: To improve and optimize build times it is recommended to create a gradle.properties file for your project and include the following.

```
# Project-wide Gradle settings.
# IDE (e.g. Android Studio) users:
# Gradle settings configured through the IDE *will override*
# any settings specified in this file.
# For more details on how to configure your build environment visit
# http://www.gradle.org/docs/current/userguide/build_environment.html
org.gradle.daemon=true

# Specifies the JVM arguments used for the daemon process.
# The setting is particularly useful for tweaking memory settings.
# Note: next command should be all on one line
org.gradle.jvmargs=-Xmx4608M -XX:MaxPermSize=512M -XX:
+HeapDumpOnOutOfMemoryError -Dfile.encoding=UTF-8
```

Record360Interface Class

Use these overriden callback methods to respond to various transaction upload events. The Record360Interface is passed in when starting a Record360 session.

```
void onUserAuthenticated(final String username, final String userId, final String
    token);

void onFailedToAuthenticate(final boolean credentialsAreValid, final String error);

// Deprecated after 1.3 use new functions
Map<String, String> getTransactionData(final String referenceNumber, Map<String,
String> transactionData);

// Replaces above function for versions 1.4+
Map<String, String> onReferenceNumberEnteredWithFieldData(final String referenceNumber,
Map<String, String> transactionData);

// Added in 1.4+, for mapping into PDF/Contract forms
Map<String, String> onContractFieldData(Map<String,String> contractFieldData);

void onTransactionComplete(final String referenceNumber);

void onTransactionCancelled(final String referenceNumber);

void onTransactionUploadProgress(final String referenceNumber, final long complete,
    final long total);

void onTransactionUploaded(final String referenceNumber);

void onTransactionUploadFailed(final String referenceNumber, final String error);
```

Transaction data will be passed to the following callback function. Here you can see the key-value pairs for the form and/or email data. Return the map object with any changes you would like to propagate back into the SDK.

```
@Override
public Map<String, String> onReferenceNumberEnteredWithFieldData(String
    referenceNumber, Map<String, String> transactionData) {

    // Replace the first parameter with form and control names from your
    // workflow you wish to replace. A list of fields is in data map keys.
    transactionData.put("Inspection Report.Customer Name:", "John Doe");
    transactionData.put("Inspection Report.Multi Line Text Example:",
    referenceNumber);

    // Example for overwriting email_sent_to_list, accepts a list of emails
    // (only valid emails will be added to transaction)
```

```
    transactionData.put("email_sent_to_list", "john@domain.com,
    test@domain.org", "invalidWontBeAdded");

    // return data
    return transactionData;
}

@Override
public Map<String, String> onContractData(Map<String, String> contract) {
    // You can replace contract data using the map passed into this functions
    contractData.put("Contract1.Customer Name:", "John Doe");
    return contractData;
}
```

Record360Activity Class

The Record360Activity class is meant to be extended in order to give you access to methods that allow you to start Record360 sessions.

If you would like to start a session using the Record360 authentication page, the function below will launch a new activity and ask the user for their username and password combination. This activity handles authentication errors and will notify the user if authentication fails.

```
public void start(Context context, Record360Interface)
public void start(Context context, String referenceNumber, Record360Interface
rec360Interface)
```

To create your own login activity, you can start a session using user credentials that you provide with the function below. This can also be useful for static credentials.

```
public void authenticateAndStart(Context context, final String username, final
   String password, Record360Interface record360Interface)
public void authenticateAndStart(Context context, final String username, final
   String password, String refNum, Record360Interface record360Interface)
```

If you already have a previous obtained user id and valid token you can use the function below to start the session.

```
public void authenticatedStart(Context context, final String token,
   @Nullable final String refNum, final String userId,
   Record360Interface record360Interface)
```

A reference number can also be specified to launch the user into created transaction with the specified reference number on all of the above start functions.

The Record360Activity can also be used to unregister the Record360Interface using the following function. Context object passed in should be the same one used to start the session.

```
public static void stop(Context context)
```

Record360Setting Class

Use one of the below constructors to create a Record360Setting that can be used to modify the workflow process. Use one of the SETTING_ constants as the first argument with the appropriate init method. The Example project contains various settings configurations.

```
public Setting(String setting)
public Setting(String setting, String defaultValue, boolean displayed)
public Setting(String setting, String label, String link)
private Setting(String setting, String label, String address, String subject, String
defaultValue, boolean displayed)
```

Use the Record360.Settings[] object to build an array of Settings that you can pass into the SDK initialization function.

```
Record360SDK.Setting[] sdkSettings = new Record360.Setting[]{
     new Record360SDK.Setting(SETTING_LOGOUT),
     new Record360SDK.Setting(SETTING_NATIVE_RESOLUTION, Boolean.toString(false), true),
     new Record360SDK.Setting(SETTING_LINKS, "Label", "https://www.domain.com"),
}
```

Here is a list of settings and their possible values.

```
// Setting for camera resolution
public static final String SETTING_RESOLUTION;
// Possible values
public static final String RESOLUTION_MEDIUM;
public static final String RESOLUTION_HIGH;
public static final String RESOLUTION_VERY_HIGH;

// Setting for Transaction Uploading Mode
public static final String SETTING_UPLOAD_MODE;
// Possible values
public static final String UPLOAD_MODE_ONLINE;
public static final String UPLOAD_MODE_WIFI_ONLY;
public static final String UPLOAD_MODE_OFFLINE;

// Setting for license scan region
public static final String SETTING_LICENSE_REGION;
// Possible values
public static final String REGION_UNITED_STATES;
public static final String REGION_CANADA;
public static final String REGION_AMERICAS;
public static final String REGION_AUSTRALIA;
public static final String REGION_ASIA
public static final String REGION_EUROPE;
public static final String REGION_AFRICA;
```

```
// Settings that take a value of true or false
// If added will be present as On/Off switch in settings
public static final String SETTING_NOTATIONS_ON_IMAGES;

// VIN Scan Mode
// enables/disables lock of reference number capture to vin scan mode
public static final String SETTING_VIN_SCAN;

// Native Resolution Mode
// enables the camera to take pictures at native resolution
public static final String SETTING_NATIVE_RESOLUTION;

// Timestamp Mode
// enables/disables timestamp on photos/videos
public static final String SETTING_TIMESTAMP_MODE;

// Other settings available that produce button in settings
public static final String SETTING_ACCOUNT;
public static final String SETTING_LOGOUT;
public static final String SETTING_VERSION;
public static final String SETTING_SEND_SUPPORT_LOG;
public static final String SETTING_RATE_RECORD360;
public static final String SETTING_LINKS;
```

Questions?
Justin Friberg – justin@record360.com
Alex Valencia – alex@record360.com
Visit us on the web at www.record360.com/business

Rest assured. We've got you covered.

RECORD 360™