LIPSECURE ANDROID SDK

Developer Guide

CONTENTS

Document Version:	1.0
LipSecure SDK Version:	Version 1.0.0
Date:	09 January 2019

1 OVERVIEW OF THE LIPSECURE ANDROID SDK

1.1 About LipSecure

The LipSecure service provides liveness checking for your Android applications, enabling application developers to robustly verify that a real user is present at key points during application operation. LipSecure prompts the user to speak a specific phrase or digit sequence and captures a short video of the user's lips during their response. The video is analysed by the LipSecure cloud platform and a liveness score returned to the client application.

1.2 About the LipSecure Android SDK

The LipSecure Android SDK offers two levels of operation depending upon the level of control needed by the developer in delivering their application. These two levels are:

- A packaged LipSecure *FragmentActivity* for direct incorporation into any existing application pathway. Several configurable options are available to customise the look and feel of the LipSecure component.
- Low-level, direct access to the LipSecure REST API to post and analyse captured video. This approach is helpful when full control is needed of the application user-interface or hardware components such as the camera and device accelerometer. When using the LipSecure API directly, the developer must ensure that the video captured is of the correct dimensions and format and that during video capture the user's face is properly positioned in the view and that the camera orientation is within acceptable bounds.

2 QUICK-START GUIDE

The quickest way to integrate LipSecure into your application is to use the *LipSecureApplication* singleton within a target activity. The LipSecure component is initialised as follows:

```
import uk.co.liopa.lipsecuresdk.LipSecureApplication;

public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        LipSecureApplication.INSTANCE.setupAPI(
            getResources().getString(R.string.base_url),
            getResources().getString(R.string.api_key)
        );
        LipSecureApplication.INSTANCE.setContext(this.getApplicationContext());
        LipSecureApplication.INSTANCE.setSavedInstanceState(savedInstanceState);
        LipSecureApplication.INSTANCE.setActivity(this);
        LipSecureApplication.INSTANCE.setupUI();
    }
}
```

When you register for LipSecure you will be provided with a unique API key and a URI base address. For example:

API Key: `38332b622aea11e8b103a860b60304d5` *(key for illustrative purposes only)*

LipSecure Base URI: `https://api.lipsecure.com`

Plug these into the setupAPI method, preferably by setting string values in your ***`strings.xml`*** *resource file:*

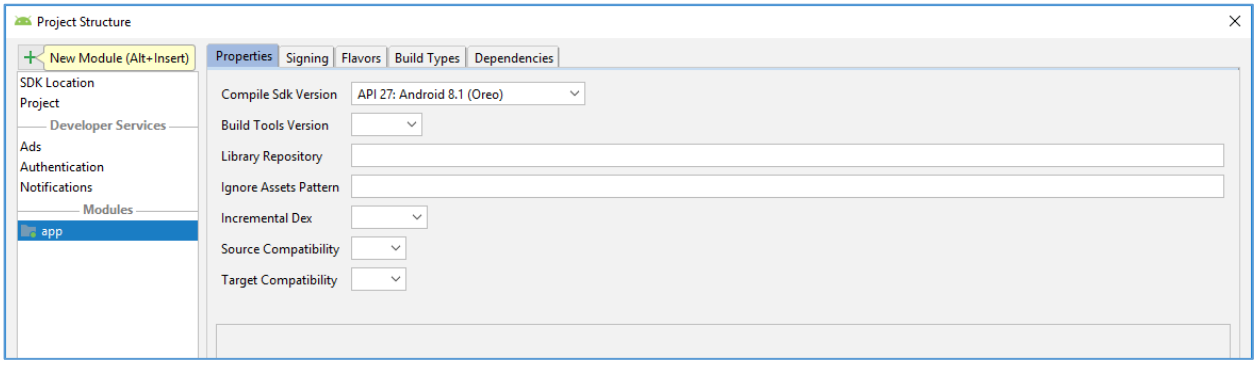
```
LipSecureApplication.INSTANCE.setupAPI(
            getResources().getString(R.string.base_url),
            getResources().getString(R.string.api_key)
);
```

Note that you will also receive a link to the ***LipSecureSDK*** *AAR module that you will need to download and import into your application.*

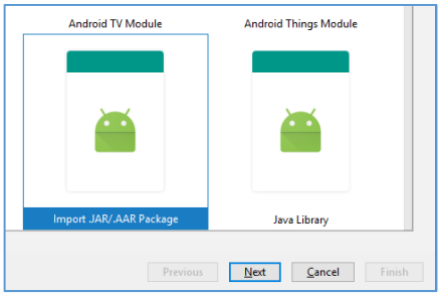
2.1 Importing the LipSecure SDK Module

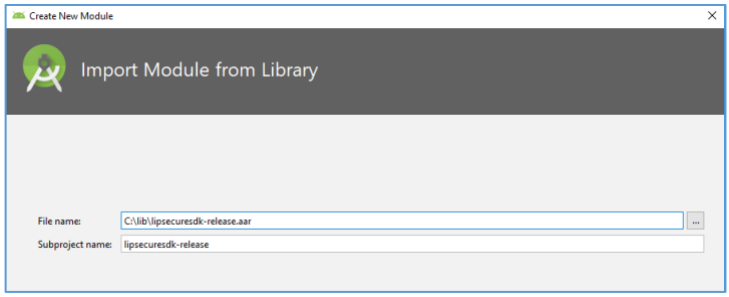
Import the LipSecureSDK module into your application using your preferred method for your development environment. The following procedure applies to Android Studio:

- Download the LipSecureSDK AAR file using the link provided during registration.
- Open the 'Module Settings' window on Android Studio (select you project and hit 'F4' or right-click > "Open Module Settings".

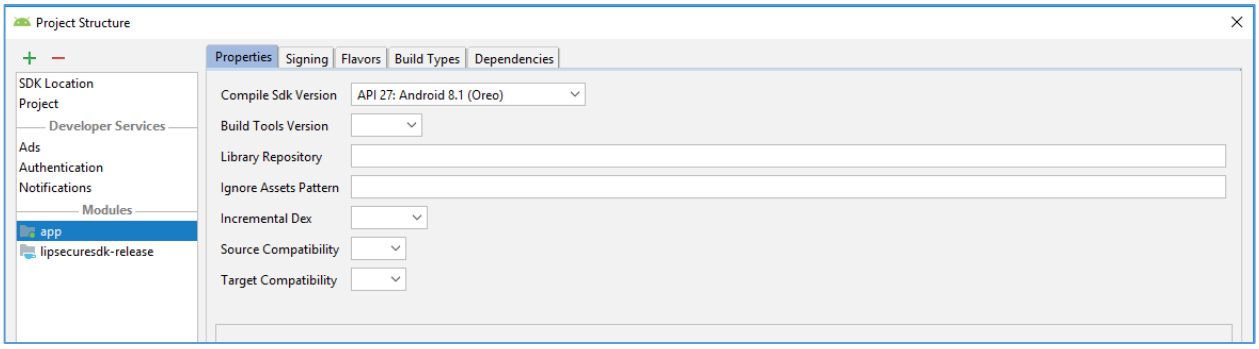


- Click the '+' icon to add a new module and select 'Import JAR/AAR Package':

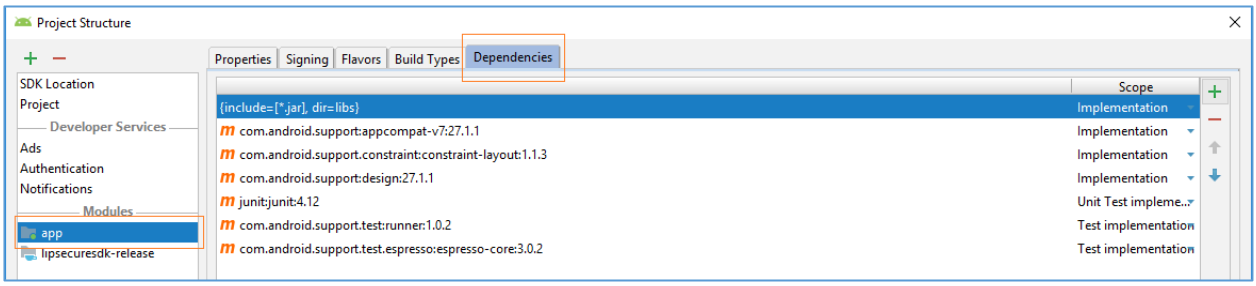




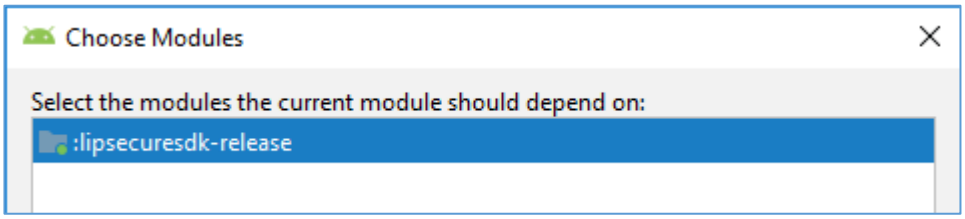
- Click 'OK' to import the LipSecureSDK AAR into your project:



- Next add a dependency to the imported SDK module from your app. This can be done through the same Module Settings UI by selecting your application in the Modules section on the left of the screen and then selecting the 'Dependencies' tab:



- Click the '+' icon (top right) to add a new dependency and select 'Module Dependency':



Select the *lipsecuresdk* module and click 'OK'. Finally click 'OK' to exit the 'Module Settings' window.

2.2 Additional Dependencies

The LipSecure SDK requires additional dependencies to run. To include these dependencies it's necessary to add the following lines to your build.gradle (app) file:

```
...

dependencies {
    ...
    implementation project(':lipsecuresdk-release')
    implementation 'com.android.support:support-v13:27.1.1'
    implementation 'com.daimajia.easing:library:2.0@aar'
    implementation 'com.daimajia.androidanimations:library:2.3@aar'
    implementation 'com.squareup.retrofit2:retrofit:2.4.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.4.0'
    implementation 'com.squareup.okhttp3:logging-interceptor:3.9.1'
}
...
```

2.3 Adding Device Permissions

The LipSecure SDK requires access to key device resources including the camera, the accelerometer, the file system and a few other system-level resources. To enable your app to access these resources it is necessary to add them to your AndroidManifest,xml file:

```
<manifest>

    <uses-permission android:name="android.permission.CAMERA" />
    <uses-feature android:name="android.hardware.camera" />
    <uses-permission android:name="android.permission.RECORD_AUDIO" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.RECORD_VIDEO" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

    <application>
    …
    </application>
</manifest>
```

2.4 Running Your Basic App

You should be able to execute the basic LipSecure user interface with a simple call to:

```
LipSecureApplication.INSTANCE.setupUI();
```

The default LipSecure UI is presented.

3 CUSTOMISATION OPTIONS

The LipSecure SDK permits several customization options for the look and feel of the user interface.

3.1 Setting View Colours

Enable/Disable the face guide

The face guide can be disabled or reenabled as follows:

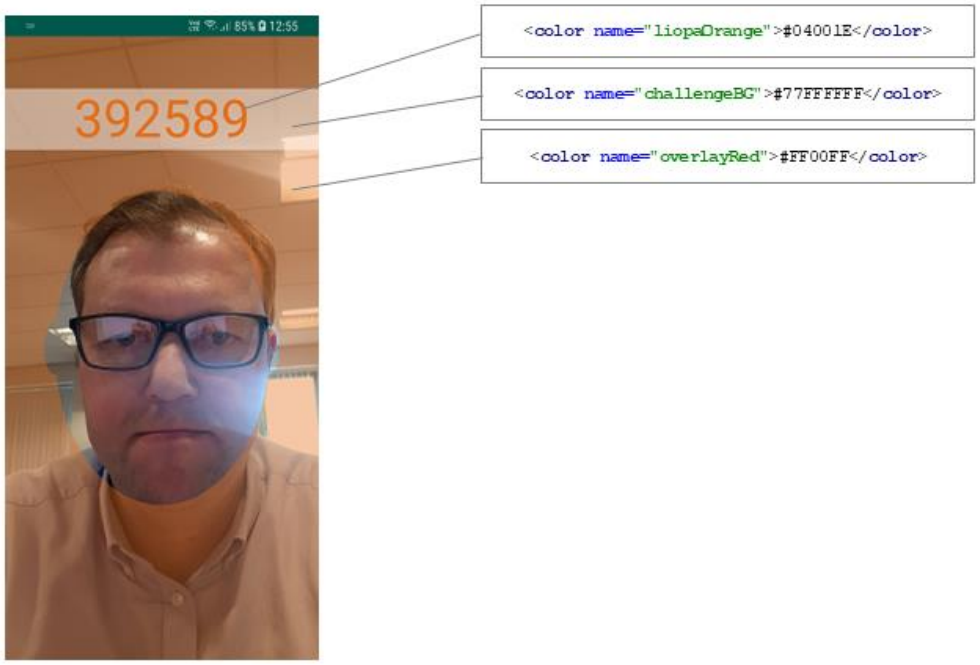
```
findViewById(R.id.headOverlay).setVisibility(View.GONE);

findViewById(R.id.headOverlay).setVisibility(View.VISIBLE);
```

Change component colours

It is possible to override the component colours used by the LipSecure library module within your app `colors.xml` file. The screenshot below illustrates the color names and usage within the default LipSecure view.

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="overlayGray">#550000</color>
    <color name="overlayRed">#FF00FF</color>
    <color name="liopaOrange">#04001E</color>
    <color name="challengeBG">#77FFFFFF</color>
</resources>
```



4 DIRECT API INVOCATION

The LipSecure SDK provides convenient access to the LipSecure API allowing client applications to upload video for testing against a known challenge phrase.

> *LipSecure API*
>
> The LipSecure API is documented here: https://liopa.github.io/ and developers can use their preferred approach to invoking REST endpoints. Alternatively, for convenience, developers can access the ready-made methods within the LipSecure SDK library to readily access the API.

To invoke a challenge request through the LipSecure SDK:

```
LipSecureApplication.INSTANCE.getAPI().challenge(new ChallengeRequest())
    .enqueue(new Callback<ChallengeResponse>() {
        @Override
        public void onResponse(Call<ChallengeResponse> call,
            Response<ChallengeResponse> response) {
            if (response.body()!=null) {
                ChallengeResponse challengeResponse = response.body();
                challenge = challengeResponse.getPhrase();
                transactionId = challengeResponse.getTransactionId();
                // Application specific code goes here
            }
        }

        @Override
        public void onFailure(Call<ChallengeResponse> call, Throwable t) {
            // Application specific code goes here
        }
    });
```

To invoke a verify request through the LipSecure SDK and retrieve a confidence value:

```
// file = handle to video file
RequestBody requestFile = RequestBody.create( MediaType.parse("video/*"), file );
MultipartBody.Part body = MultipartBody.Part.createFormData("file", file.getName(),
requestFile);

LipSecureApplication.INSTANCE.getAPI().verify(token, transactionId, body)
    .enqueue(new Callback<ResponseBody>() {
        @Override
        public void onResponse( Call<ResponseBody> call,
                                Response<ResponseBody> response) {
            if (response.isSuccessful()) {
                JSONObject json = new JSONObject(response.body().string());
                confidence = json.getString("confidence");
                // Application specific code goes here
            }
        }
    });
```