

Android Public LOUIE7 SDK DOCUMENTATION

Overview

The Louie7 SDK collects user data according to permissions given by the user.

Integrate Louie7 Android SDK into your Android app

To integrate Louie7 SDK into your app, you have to follow these steps:

1. Download the archive. Unarchive it into the preferable folder. It can be something like `/Users/user.name/Documents/repository/`. The path to the `*.aar` file should be something like `/Users/user.name/Documents/repository/com/louie7/louie-collectors/{current_sdk_version}`. Make sure, that the first folder in the repository is `com`, not the `louie-preview-sdk`.
2. Add the following line in your `build.gradle` file.

```
repositories {  
    google()  
    mavenCentral()  
    ...  
    maven {  
        url "path/to/your/repository"  
    }  
}
```

3. In your application level gradle file add the following code:

```
dependencies {  
    ...  
    implementation 'com.louie7:louie-collectors:{sdk_version}'  
    ...  
}
```

4. Sync your project.

Gradle files have changed since last project sync. A project sync may be necessary for the IDE to work properly.

[Sync Now](#) [Ignore these changes](#)


5. In your `Application` class in the `onCreate` method add the following code:

```


class Application : Application() {
    override fun onCreate() {
        super.onCreate()
        ...
        LouieSdk.Builder(this)
            .setAuth( "$X-API-KEY" )
            .build()
            .start()
    }
}

```

6. You successfully integrated Louie7 SDK into your app. Check [Implement gesture collection](#) guide to implement gesture collection.

 Make sure, that you have added your application class in the Android manifest.

 You can get your X-API-KEY from the dashboard.

 If you done everything right and there is internet connection on your test device, you will see the **Connection with Louie7 servers established** message.

Helpful information

If you are using proguard, make sure, that you have added these rules to the [proguard-rules.pro](#):

```

-keep class com.louie7.** {*; }
-keepclassmembers enum * {*; }


```

The SDK works in **background** using coroutine worker. **Only the phone's OS decides how much time background worker will be allowed to be active.** Also it restarts only if the OS allows it. If the application stays in memory or goes in foreground, the SDK's worker will always start, so basically the SDK's lifetime closely connected both to application usage time/frequency and phone's memory and battery state. You **can request ignore battery optimization**, if it is possible.

```

try {
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
        startActivity(
            Intent(
                Settings.ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS,
                Uri.parse("package:" + requireContext().packageName)
            )
        )
    }
} catch (e: Exception) {
}

```

 To get current user identifier, simply call `LouieSdk.getCurrentIdentifier(context)`

Implement gesture collection


Gesture collection is strictly dependent on android views. Louie7 need access to views to collect gestures in your app. So, there is some options to implement gesture collection into your app.

Method 1: use `GestureActivity` as your parent activity.

1. In **Android Studio** select your **main activity** class.
2. Replace `AppCompatActivity` in your class with the `GestureActivity`.

```
class MainActivity : GestureActivity() {  
  
    override fun onCreate(savedInstanceState: Bundle?) {  
        super.onCreate(savedInstanceState)  
        setContentView(R.layout.activity_main)  
    }  
  
}
```

3. Repeat step 2 for **every activity** in your project.

 Make sure, that you have not missed any activity in your project.

Method 2: use `GestureActivity` as a parent class for your custom activity.

1. In **Android Studio** select your **custom activity** class.
2. Replace `AppCompatActivity` in your custom activity class with the `GestureActivity`.
3. Use your custom activity for every not abstract activity class in your project.


```
abstract class MyCustomActivity : GestureActivity() {  
  
    fun internalFun1() {  
        ///  
    }  
  
    fun internalFun2() {  
        ///  
    }  
  
}
```

4. If you override `dispatchTouchEvent` method in your activity, make sure, that you have overridden it properly (both in custom and not abstract activities).


```

override fun dispatchTouchEvent(ev: MotionEvent?): Boolean {
    /// your code
    return super.dispatchTouchEvent(ev)
}

```

 Make sure, that you have not missed any activity in your project.

Method 3: use Louie7Gestures to implement gesture collection.

 Use this method if you avoid unnecessary inheritance or can't use GestureActivity.

1. In **Android Studio** select your **activity** class.
2. Create a lateinit variable of Louie7Gestures.

```

private lateinit var gestures : GesturesLouieFamily

```

3. Initialize the GesturesLouieFamily variable with the Louie7Gestures.createListener function in the onCreate method of your activity.

```

override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    //your code
    gestures = Louie7Gestures.createListener(this)
}

```


4. Override dispatchTouchEvent and call gestures.onTouchEvent(ev) (GesturesLouieFamily variable).

```

override fun dispatchTouchEvent(ev: MotionEvent?): Boolean {
    gestures.onTouchEvent(ev)
    return super.dispatchTouchEvent(ev)
}

```

5. Your activity is ready to collect gestures now.

 If you want to use this method, you should repeat it for each activity you have.

How to integrate with SDK's API

To integrate your client application you have authorize your device identifier. To authorize your device identifier send a POST request to SDK's API.

All API requests require the use of a received application key. To authenticate an API request, you should provide your application key in the X-API-Key header.

| Header Parameter | Type | Description |
|------------------|--------|----------------------|
| X-API-Key | string | Your application key |

Next , to authorize a device identifier

You have send a POST request to SDK's API using received X-API-Key key.

Authorize device identifier

```
POST /api/v1/devices
```

| Body Parameter | Type | Description |
|----------------|--------|-------------------|
| identifier | string | Device identifier |

| Header Parameter | Type | Description |
|------------------|--------|----------------------|
| X-API-Key | string | Your application key |

Authorize device identifier example

POSThttps://sdk-api.louie7ai.com/api/v1/devicesSend

ParamsAuthorizationHeaders (9)BodyPre-request ScriptTestsSettingsCookies

Headers8 hidden

| KEY | VALUE | DESCRIPTION |
|-----------|--------------------------------------|-------------|
| X-API-Key | cdfb92ed-1916-4a78-b442-b86ac4ad4b95 | |
| Key | Value | Description |

Response

POSThttps://sdk-api.louie7ai.com/api/v1/devicesSend

ParamsAuthorizationHeaders (9)BodyPre-request ScriptTestsSettingsCookies

Body

123

123

BodyCookiesHeaders (11)Test Results

PrettyRawPreviewVisualizeJSON

123

Status: 200 OKTime: 261 msSize: 578 BSave Response

Now , that you've successfully authorized your application's device identifier , you request application's SDK's config.

Next, to receive application config

You have send a GET request to SDK's API using received access (Authorization) header.

Receive application config

GET /api/v1/application/config

| Header Parameter | Type | Description |
|------------------|--------|-------------------------------|
| Authorization | string | Your device identifier access |

Receive application config example

GEThttps://sdk-api.louie7ai.com/api/v1/application/configSend

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettingsCookies

Headers6 hidden

| KEY | VALUE | DESCRIPTION |
|---|--|-------------|
| <input checked="" type="checkbox"/> Authorization | Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZGVudGmaWVjoleW9 | |
| <input checked="" type="checkbox"/> X-API-Key | cdfb92ed-1916-4a78-bf42-b86ac4ad4b95 | |
| Key | Value | Description |

GEThttps://sdk-api.louie7ai.com/api/v1/application/configSend

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettingsCookies

Headers6 hidden

| KEY | VALUE | DESCRIPTION |
|---|--|-------------|
| <input checked="" type="checkbox"/> Authorization | Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpZGVudGmaWVjoleW9 | |
| <input checked="" type="checkbox"/> X-API-Key | cdfb92ed-1916-4a78-bf42-b86ac4ad4b95 | |
| Key | Value | Description |

BodyCookiesHeaders (11)Test Results

PrettyRawPreviewVisualizeJSON

```
1 {
2   "apiKey": "cdfb92ed-1916-4a78-bf42-b86ac4ad4b95",
3   "lifetime": 30,
4   "bucketApi": "https://objectstorage.uk-london-1.oraclecloud.com/p/zCab8Qm6aaag0x6pRRNFtTPuASgd1nsa3g8SF1AMR0uW6Pw7hJMBT9_8s0gd0n/1zstouvvrg/h/ebs-integration/q/",
5   "configuration": {
6     "enablePowerManager": true,
7     "enableActionBattery": true,
8     "enableGestureManager": true,
9     "enableImageMediaStore": false,
10    "enableLocationManager": true,
11    "enableMusicMediaStore": true,
12    "enableCalendarContract": true,
13    "enableContactsContract": true,
14    "enableConnectivityManager": true
15  },
16   "wifiSampleRate-minutes": 25,
17   "batterySampleRate-minutes": 25,
18   "screenSampleRate-minutes": 250
19 }
```

Status: 200 OKTime: 200 msSize: 972 BSave Response

You can also download the application config as a file

Download application config

GET /api/v1/application/config/download