

Android SDK

Manual includes the integration of SDK using eclipse and android studio.

Android Studio User:-

Step 1: Extract the zip file and use .aar file for integration

Step 2: Import .aar file into your workspace and add the dependency to your application

Step 3: Follow the below code structure for integration.

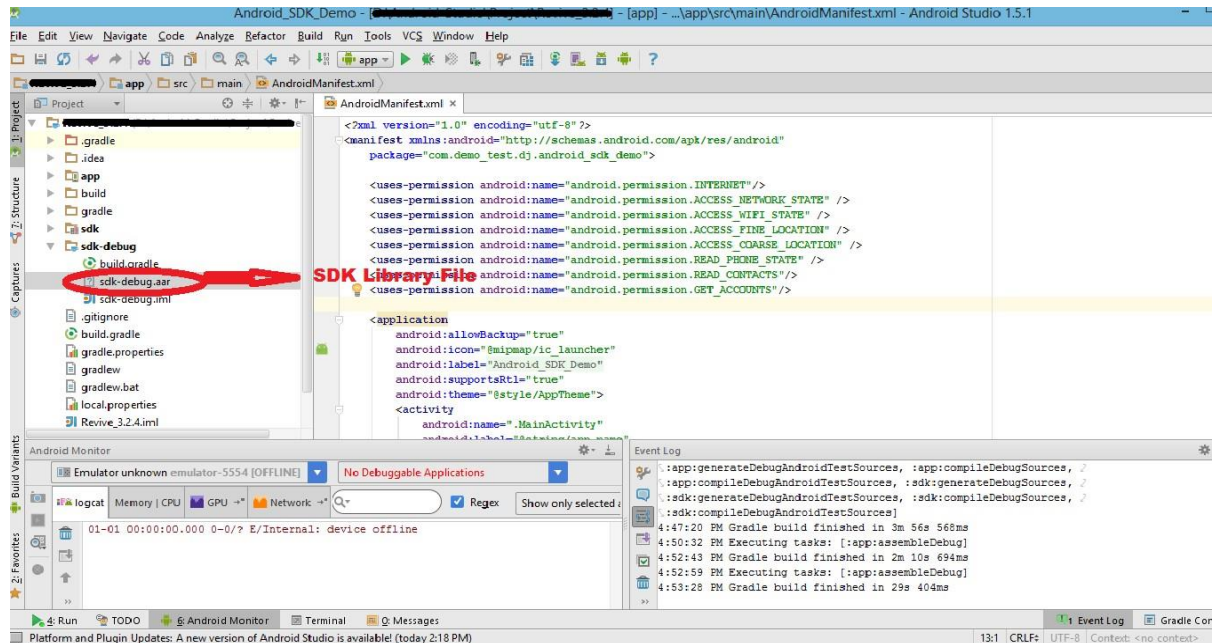


Fig: Android Studio With Library File

Code Structure:

Edit App Permissions

Edit your Android manifest to include the permissions needed by this app:

```
<uses-permission android:name="android.permission.INTERNET"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.GET_ACCOUNTS"/>
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

Permission Table:

S.NO	PERMISSION	IMPORTANTS	DESCRIPTIONS
1	INTERNET	Required	Grants the SDK permission to access the internet.
2	ACCESS_NETWORK_STATE	Required	Grants the SDK permission to check for a live internet connection. Allows applications to access information about networks
3	ACCESS_WIFI_STATE	Required	Grants the SDK permission to access information about Wi-Fi networks. Wifi state refers to the state of the phone's Wifi connection.
4	ACCESS_FINE_LOCATION	Optional	Grants the SDK permission to access a more accurate location based on GPS. For GEO ads targeting
5	ACCESS_COARSE_LOCATION	Optional	Grants the SDK permission to access approximate location based on cell tower.
6	GET_ACCOUNTS	Optional	Grants the SDK permission to the list of accounts in the Accounts Service.
7	READ_CONTACTS	Optional	Grants the SDK permission to the list of Email accounts in the Accounts Service.
8	WAKE_LOCK	Optional	Grants the SDK permission to use a wakelock for keeping the screen turned on
9	READ_PHONE_STATE	Optional	Grants the SDK permission to use this permission for get the user carries details.

In App side app developers to develop the apps above the Android 6.0 they need to write code for **runtime permission** in android. Below Android 6.0 no need write run time permission.

Dependency

We need to add below the dependency to **app level** gradle file.

```
implementation 'com.google.code.gson:gson:2.8.2'

implementation 'com.google.android.gms:play-services-ads-lite:11.8.0'
implementation 'com.google.android.gms:play-services-location:11.0.0'
implementation 'com.google.android.gms:play-services-maps:11.0.0'

implementation project(':sdk')
```

If you face any errors like **Execution failed for task ':app:checkDebugDuplicateClasses'**. after implement the above dependencies add the below code set in your app level gradle file(This error occurs selected development environment only not for all).

```
configurations.all {
    resolutionStrategy.eachDependency { DependencyResolveDetails details ->
        def requested = details.requested
        if (requested.group == 'com.android.support') {
            if (!requested.name.startsWith("multidex")) {
                details.useVersion '29.0.2'
            }
        } else if (requested.group == "com.google.android.gms") {
            details.useVersion '11.8.0'
        }
    }
}
```

Location permissions can help monetization

Although not technically required, the *LOCATION permissions make it possible for the SDK to send location-based data to advertisers. Sending better location data generally leads to better monetization. Please note that the SDK will never wake up the phone to request the location to be updated; this would take time and battery. Instead, it will use these permissions to access the last known location of the device.

Show Ads

This section describes some of the code you'll write in order to show ads.

This document refers to something called a "zone ID". A zone ID is just a numeric ID used by MSDK to identify a context within an app where advertisements can be shown. You'll need to obtain a zone ID from your MSDK representative or your ad network. Without it, you won't be able to fetch and display ads.

XML	Java Equivalent	Description	Example
msdk:zone_id	ad.setZoneid()	The zone ID associated with your app's inventory. You must include a zone ID or an error will be thrown. This method supports Multi zones. We can able to pass more than one zone IDs. Even it support 1 zone id.	"436,437,438,439"
msdk:auto_refresh_time	ad.set_Auto_refresh_time ()	The interval, in milliseconds, at which the AdView will request new ads, if autorefresh is enabled. The minimum period is 15 seconds. The default period is 30 seconds. Set this to 0 to disable autorefresh.	"30000"
msdk:setProfileInfo	ad.setProfileInfo ()	The profile target the user.if you need use this function.add any number of custom data	age=25&gender=Male&height=165&weight=60

msdk: setKeywords	ad.setKeywords()	Using this keywords targeting Sports,Politics,Art,Movies,Books we pass the set of keywords to target the users, based on the users interest like that.	
----------------------	----------------------------------	--	--

If you're using XML, you'll need to add the `xmlns:msdk` namespace attribute describing your application to your layout tag; for example this might be a RelativeLayout, LinearLayout, or FrameLayout and ScrollView.

`xmlns:msdk="http://schemas.android.com/apk/res-auto"`

1.Interstitial Video Ads :-

You can configure your Interstitial Video ad view using Java.

1.1.Code Format:

```
AdView ad = new AdView(this,this);
ad.setZoneid("200");
ad.setProfileInfo("age=25&gender=Male&height=165&weight=60");
ad.setKeywords("Sports,Politics,Art,Movies,Books");
ad.LoadAd();
```

1.2. Implement the Interface in your Class

```
public class YourActivity extends Activity implements
    InterstitialAdListener
```

1.3. Implement the override methods:

```
@Override
public void AdLoaded() {

}
@Override
public void AdFailed() {

}
@Override
public void Adclosed() {

}
```

```
@Override
public void Adclicked() {

}
@Override
public void Adshown() {

}
```

1.3. Explanation about the Integration steps:

```
final AdView adv1 = new AdView(this,this);
```

This is our entry point of our Android SDK whenever we call this line it will create the new object for the **AdView** class. Using this class we can access the methods of the **AdView** class. And also we can pass the **context** of the application and **listener**. **adv1** is an object for the class.

```
adv1.setZoneid("200");
```

Here **setZoneid()** function accept **String** type.

```
adv1.setProfileInfo("age=25&gender=Male&height=165&weight=60");
```

The **setProfile()** function used for to pass the customized variables like profile details which is given by App developers. Its support the string type. Using this function we can target the users based on the parameter values.

```
adv1.setKeywords("Sports,Politics,Art,Movies,Books");
```

The **setkeywords()** function used for to pass the keywords.Its support the string type. Using this function we can target the users based on the parameter values.

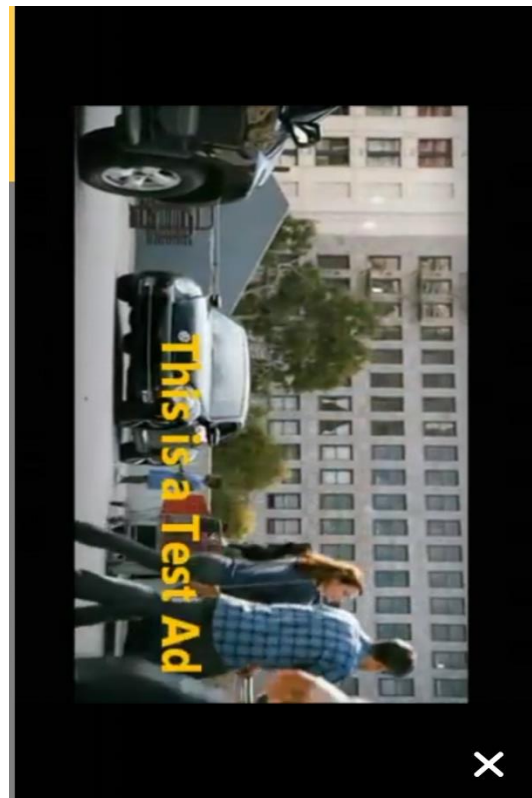
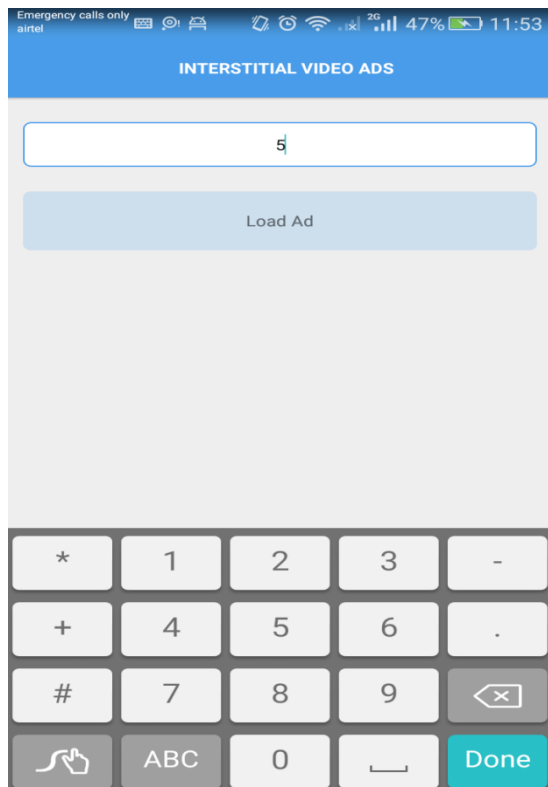
```
adv1.LoadAd();
```

The **LoadAd()** is main function in AdView class. This function is sent a request to AdServer and gets the results.

Implement the listener to our class once implement listener to the class we need to override the above methods. These methods are used to find the ads are load/show/click states.

1.4.Djax demoapp Procedure:

1. Enter the zone id then click load ad button.



Foot Note: Interstitial Video does not contains any UI/XML codes.

2.Interstitial Image Ads

You can configure your ad view using Java.

2.1.Code Format:

```
AdView adv1 = new AdView(this);
adv1.setZoneid("201");
adv1.setProfileInfo("age=25&gender=Male&height=165&weight=60");
adv1.setKeywords("Sports,Politics,Art,Movies,Books");
adv1.LoadAd();
```

2.2. Explanation about Integration steps:

```
final AdView adv1 = new AdView(this);
```

This is our entry point of our Android SDK whenever we call this line it will create the new object for the **AdView** class. Using this class we can access the methods of the **AdView** class. And also we can pass the context of the application.

adv1 is an object for the class.

```
adv1.setZoneid("201");
```

Here **setZoneid()** function accept the **String** type.

```
adv1.setProfileInfo("age=25&gender=Male&height=165&weight=60");
```

The **setProfile()** function used for to pass the customized variables like profile details which is given by App developers. Its support the string type. Using this function we can target the users based on the parameter values.

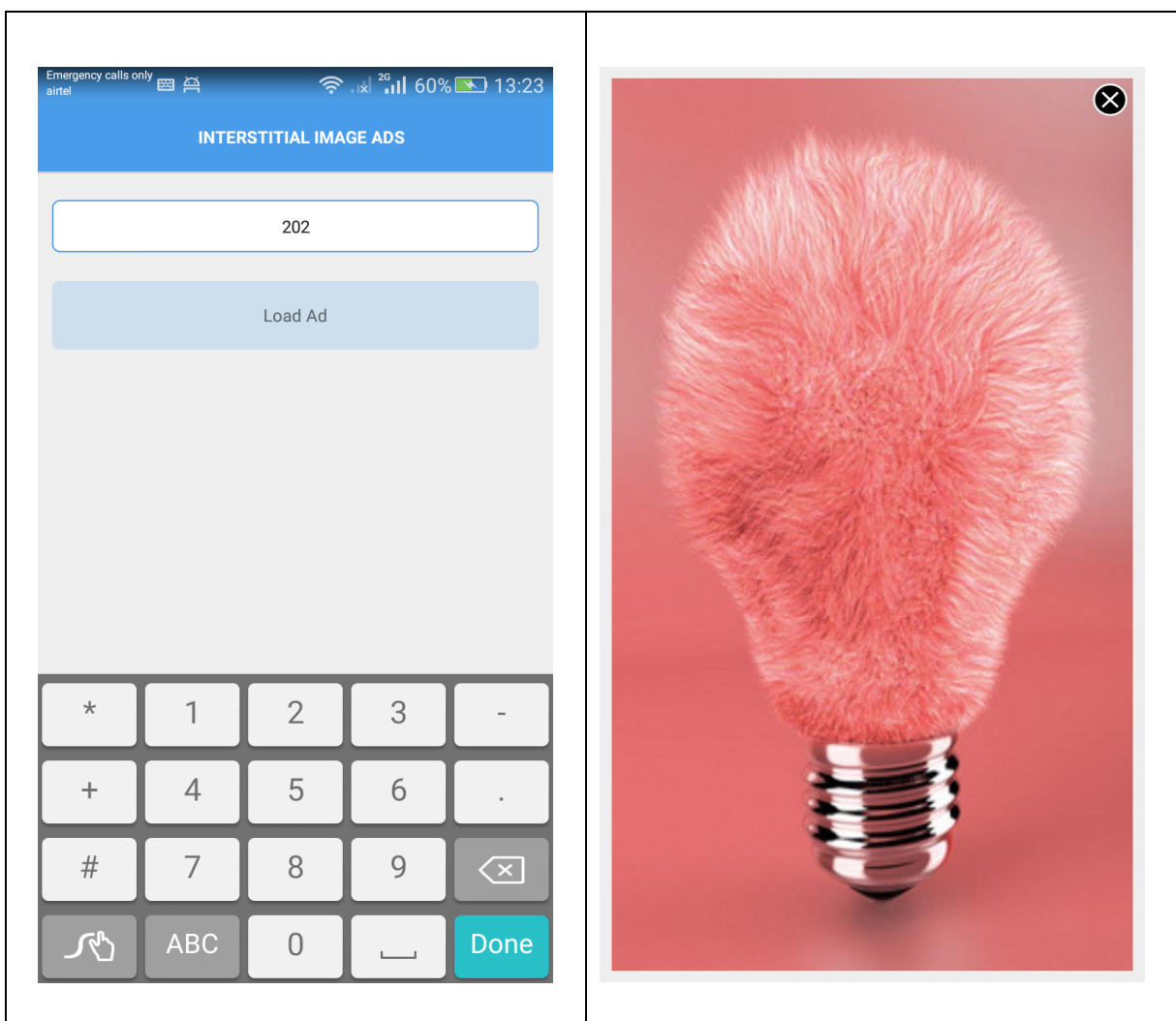
```
adv1.setKeywords("Sports,Politics,Art,Movies,Books");
```

The **setkeywords()** function used for to pass the keywords.Its support the string type. Using this function we can target the users based on the parameter values.

```
adv1.LoadAd();
```

The **LoadAd()** is main function in AdView class. This function is sent a request to AdServer and gets the results.

2.3.Djax demoapp Procedure:



Foot Note: Interstitial Image Ads does not contain any UI/XML codes.