**Android SDK**

Manual includesthe 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

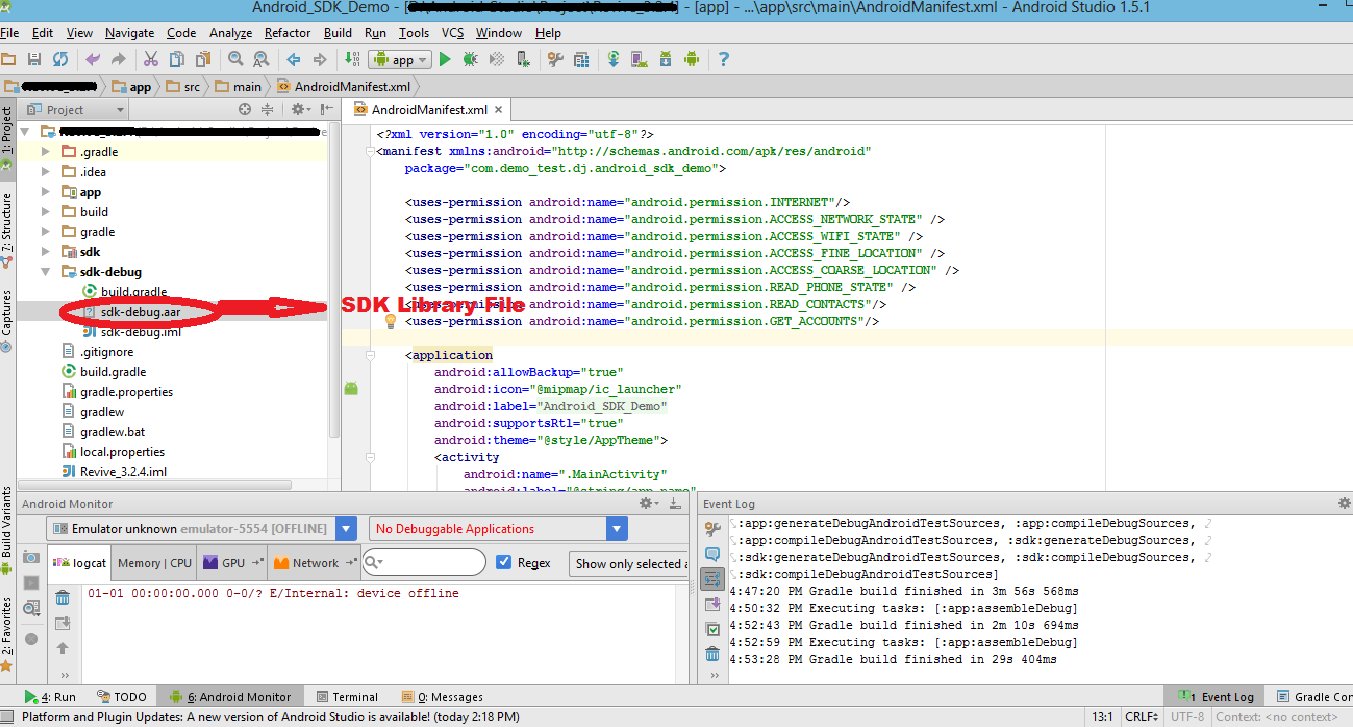


Fig: Android Studio With Library File

**Eclipse User:**

Step 1: Extract the zip file and use dJAX\_Lib library file for integration

Step 2: Import dJAX\_Lib library file into your workspace and use this library file foryour application

Step 3: Follow the below code structure for integration

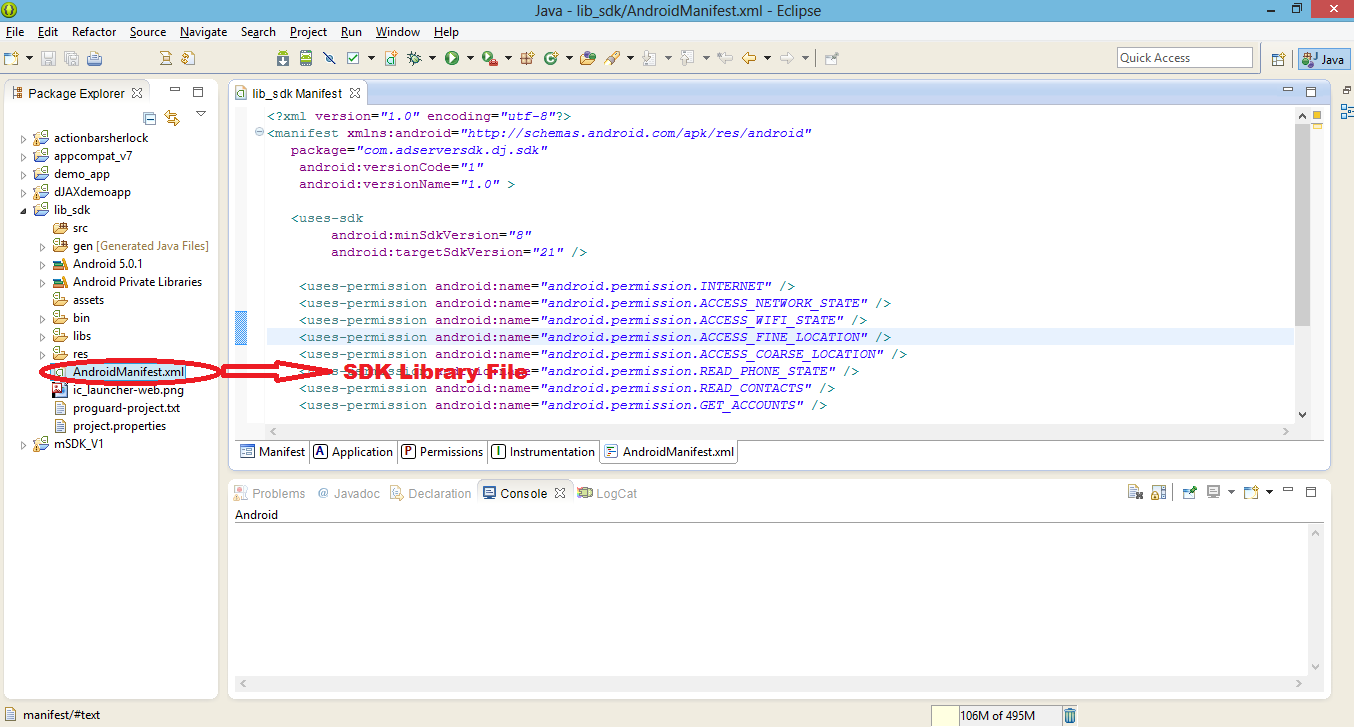


Fig: EclipseWith Library File

**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.WRITE\_EXTERNAL\_STORAGE"/>

<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />

<uses-permission android:name="android.permission.READ\_CONTACTS"/>

<uses-permission android:name="android.permission.GET\_ACCOUNTS"/>

* INTERNET (required) - Grants the SDK permission to access the internet.
* ACCESS\_NETWORK\_STATE (required) - Grants the SDK permission to check fora live internet connection.
* ACCESS\_WIFI\_STATE (required) - Grants the SDK permission to accessinformation about Wi-Fi networks.
* ACCESS\_FINE\_LOCATION (Optional) - Grants the SDK permission toaccess a more accurate location based on GPS.
* ACCESS\_COARSE\_LOCATION (Optional) - Grants the SDK permissionto access approximate location based on cell tower.
* READ\_PHONE\_STATE (Optional) - Grants the SDK permission to readonly access to phone state.
* READ\_CONTACTS (Optional) - Grants the SDK permission to read theuser's contacts data.
* GET\_ACCOUNTS (Optional) - Grants the SDK permission to the list ofaccounts in the Accounts Service.

**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.

**Banners**

You can configure your banner ad view using Java, XML, or a mixture of the two. The table below lists the XML and Java equivalents.

|  |  |  |  |
| --- | --- | --- | --- |
| **XML** | **Java Equivalent** | **Description** | **Example** |
|  |  |  |  |
| msdk:zone\_id | ad.setZoneid() | The zone ID associated with | **"436"** |
|  |  | your app's inventory. |  |
|  |  | **You must include a zone ID or an error will be thrown.** |  |
|  |  |  |  |
| msdk:auto\_refresh\_time | ad.set\_Auto\_refresh\_time() | The interval, in milliseconds, at which the AdView will | **"30000"** |
|  |  | 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. |  |

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"

**Sample code :**

Xml Format:



<LinearLayout

android:orientation="vertical"

android:layout\_width="match\_parent"

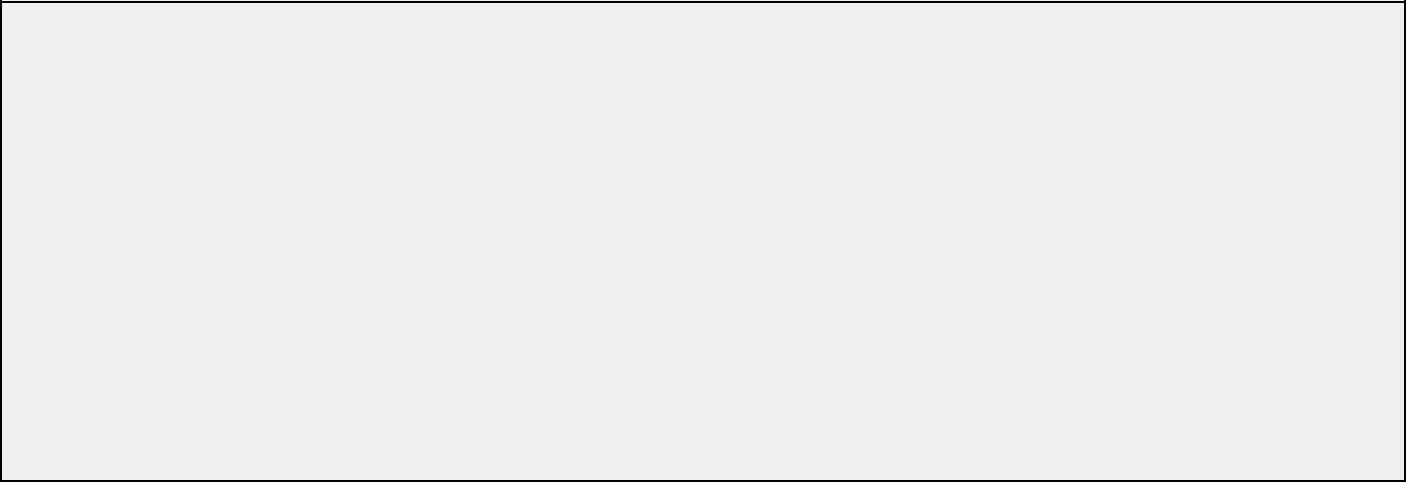
android:layout\_height="match\_parent"

android:layout\_centerHorizontal="true"

android:id="@+id/preview\_flag">

</LinearLayout>

Code Format:



LinearLayout.LayoutParamsparams = newLinearLayout.LayoutParams(LinearLayout. LayoutParams.MATCH\_PARENT,LinearLayout.LayoutParams.WRAP\_CONTENT);

LinearLayoutadFrame = (LinearLayout) findViewById(***R.id.preview\_flag***);

AdViewad = new AdView(this);

ad.setZoneid("436");//Place Your Zone id

ad.setAuto\_refresh\_time(30000);//Interval in Milliseconds

/\*Loads an Ad\*/

ad.LoadAd();

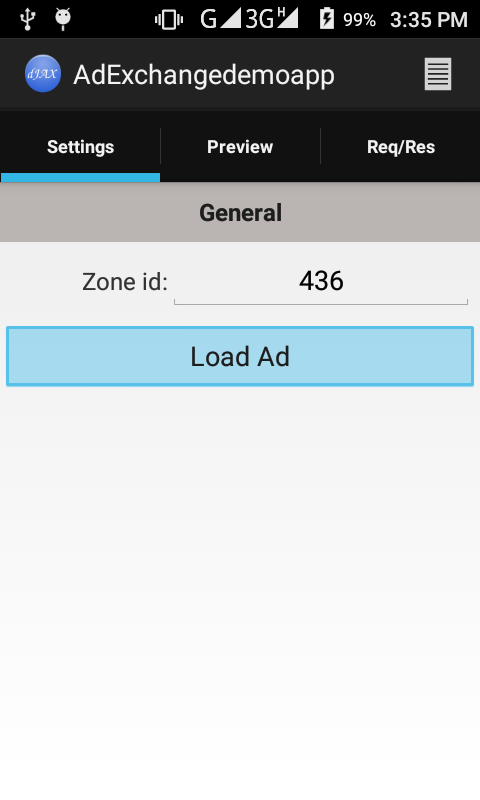
ad.setLayoutParams(params);

adFrame.addView(ad);

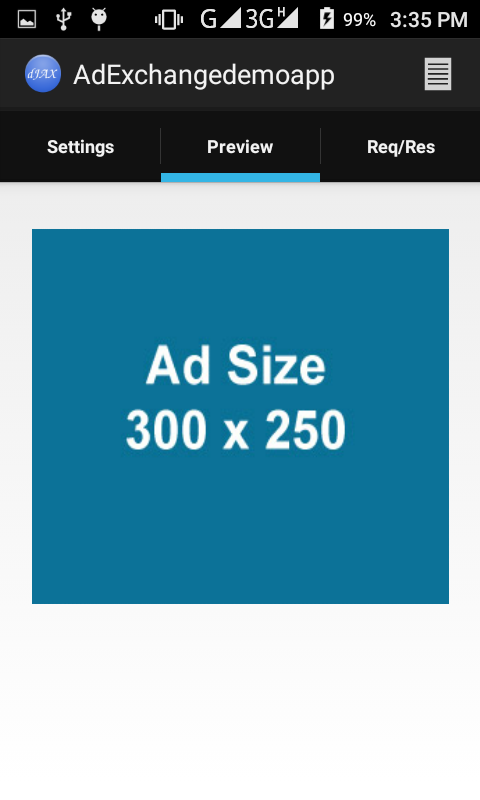
adFrame.bringToFront();

**dJAXdemoapp Procedure:**

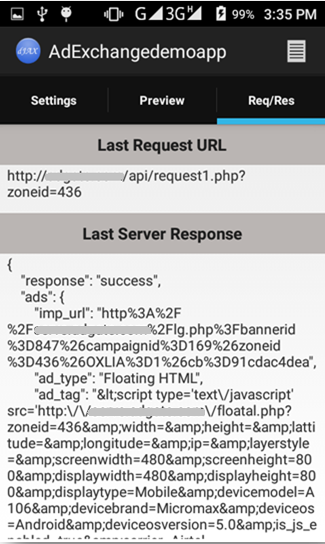
1. Enter the zone id then click load ad button.



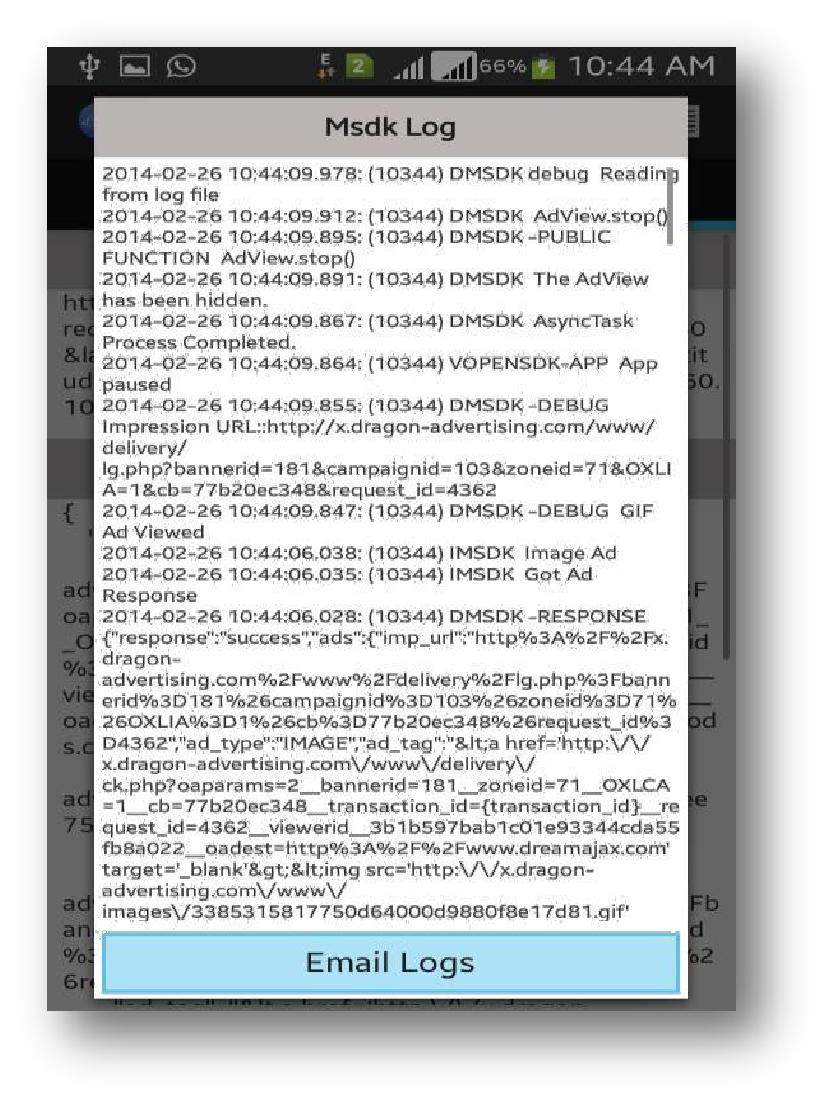
2. The requested banner is loaded in the preview tab as shown below.



3. You can view the request and response in the request tab.

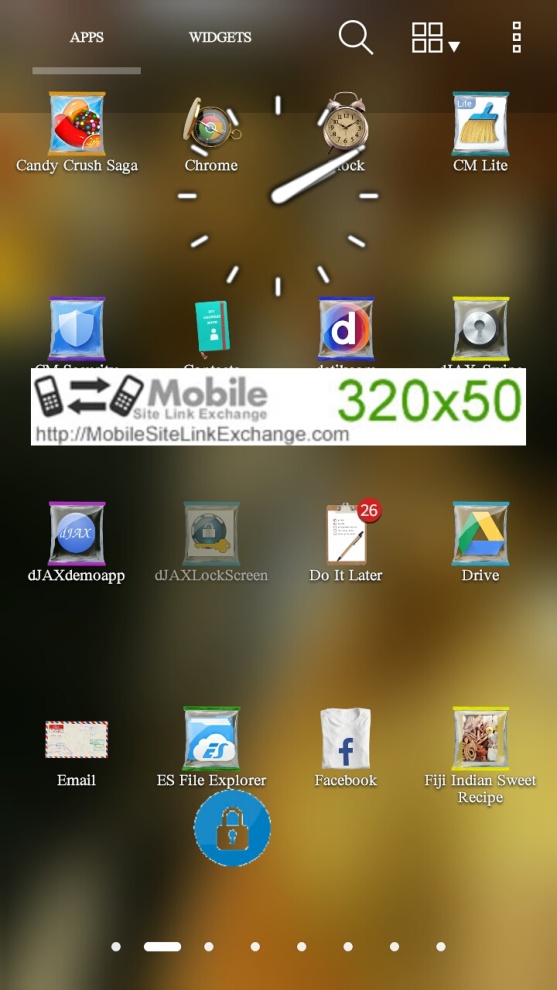


4. The Log of the app can be viewed by clicking on log icon in menu. 

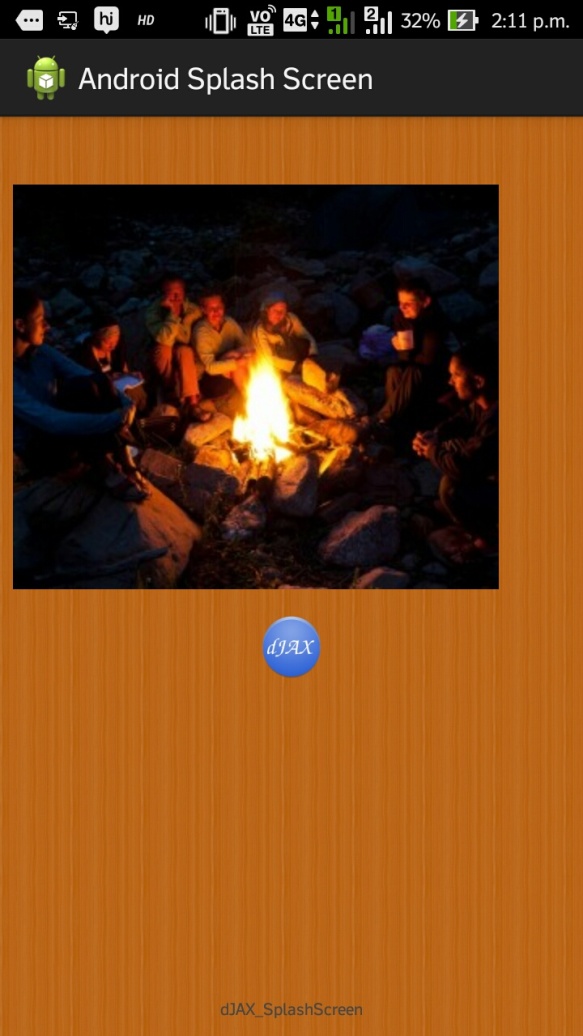


5. You can email the logs by clicking on the Email Logs button.

6. Delivering ad in Swipe Screen (Lock Screen), below is the appearance



7. Delivering ad in Splash Screen (Prestitial), below is the appearance



**Interstitial Video Ad Format**

**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

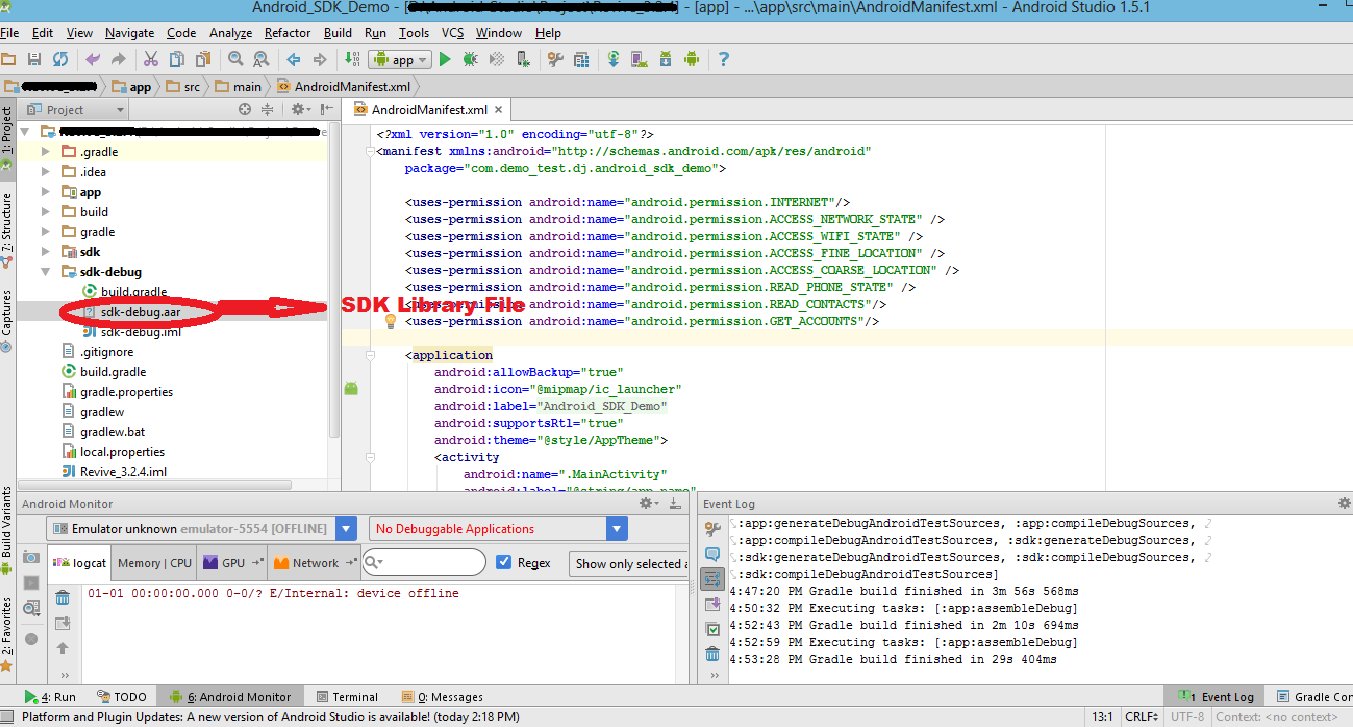


Fig: Android Studio With Library File

**Eclipse User:**

Step 1: Extract the zip file and use dJAX\_Lib library file for integration

Step 2: Import dJAX\_Lib library file into your workspace and use this library file for your application

Step 3: Follow the below code structure for integration

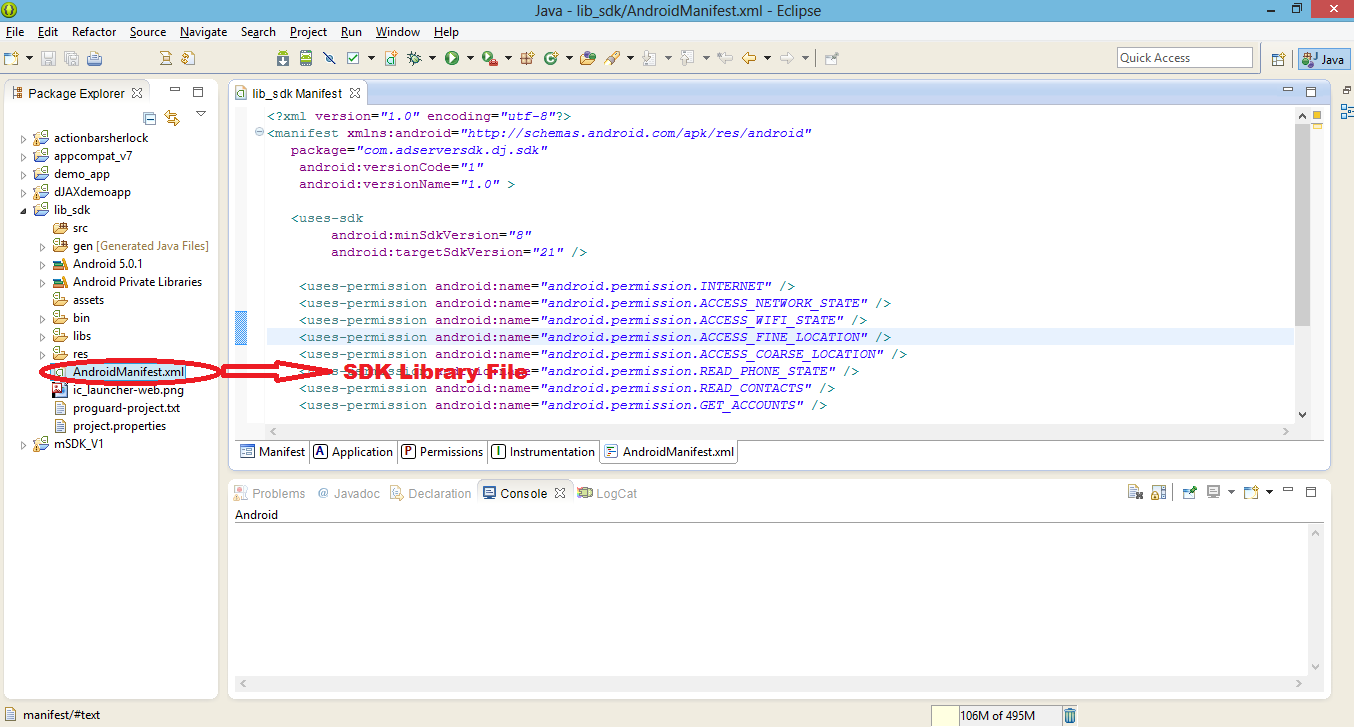


Fig: EclipseWith Library File

**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.WRITE\_EXTERNAL\_STORAGE"/>

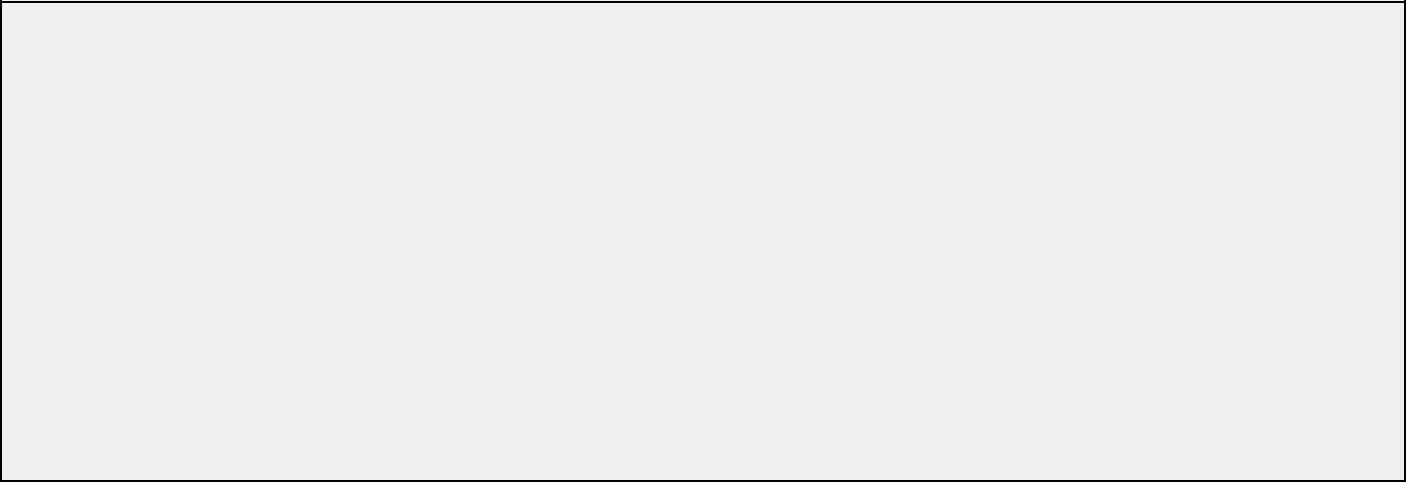
<uses-permission android:name="android.permission.ACCESS\_FINE\_LOCATION" />

<uses-permission android:name="android.permission.ACCESS\_COARSE\_LOCATION" />

<uses-permission android:name="android.permission.READ\_PHONE\_STATE" />

<uses-permission android:name="android.permission.READ\_CONTACTS"/>

<uses-permission android:name="android.permission.GET\_ACCOUNTS"/>



<application android:label="@string/app\_name"

android:icon="@drawable/ic\_launcher\_djax"

android:theme="@style/Theme.Example">

<activity android:name=".MainActivity" android:label="@string/app\_name">

<intent-filter>

<action android:name="android.intent.action.MAIN" />

<category android:name="android.intent.category.LAUNCHER" />

</intent-filter>

</activity>

**<activity android:name="** **com.adserversdk.dj.sdk.adserver.AdScreenLayout" >**

**</activity>**

</application>

</manifest>

* ACTIVITY (required) – To access the activity of interstitial video ad format
* ANDROID:NAME=”COM.ADSERVERSDK.DJ.SDK.ADSERVER.ADSCREENLAYOUT” (required) – This activity will invokes the interstitial ad file for app interstitial video ad function

**Show Ads**

This section describes some of the code you'll write in order to show ads. The code format which is given below must be included in your application MainActivity.java file.

The code must be used inside any Main Function () and the zone\_id is the keyword, you have to replace the zone\_id value instead “52” you can include as you needed. This ad will get displayed at application exit page

**Sample code:**

Code Format:



@Override

**public** **void** onBackPressed() {

Intent i = **new** Intent(getApplicationContext(),AdScreenLayout.**class**);

i.putExtra("zone\_id", "52");//Enter zone\_id value **replace "52"**

startActivity(i);

finish();

}

**After Installation:**

When you Press the interstitial view ad button of dJAXdemoapp or when you attempt to deliver your application you can view the interstitial video ad. If you need the background to be changed as black you can add it in ad\_interstitial file for the best appearance.

