**Android SDK**

**Installation**

Here's how to install the SDK as a library in your app.

**Step 1. Get the code**

First, get the code from below:

**Step 2. Import our SDK as a library**

If you just want to use our SDK, import the sdk directory into your workspace as a library using the Android IDE.

**Step 3. Edit App Permissions**

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



<?xml version="1.0" encoding="utf-8"?>

<manifest xmlns:android="http://schemas.android.com/apk/res/android" package="com.example.djaxdemoapp"

android:versionCode="1" android:versionName="1.0"> <application android:icon="@drawable/ic\_launcher"

android:label="@string/app\_name" android:allowBackup="true"

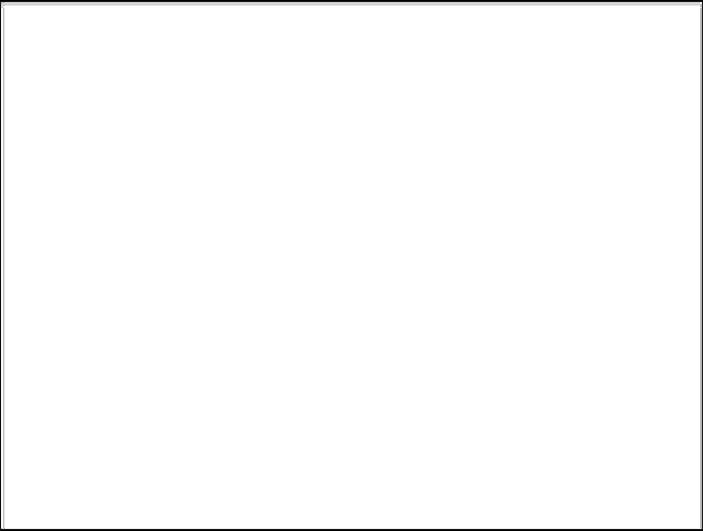
<activity android:label="@string/app\_name" android:name="com.example.djaxdemoapp.MainActivity" android:theme="@style/Theme.Sherlock">

<intent-filter>

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

</intent-filter> </activity>

</application>



**<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"/>**

**</manifest>**

* INTERNET (required) - Grants the SDK permission to access the internet.
* ACCESS\_NETWORK\_STATE (required) - Grants the SDK permission to check for a live internet connection.
* ACCESS\_WIFI\_STATE (required) - Grants the SDK permission to access information about Wi-Fi networks.
* WRITE\_EXTERNAL\_STORAGE (recommended) - Grants the SDK permission to read from external storage.
* ACCESS\_FINE\_LOCATION (recommended) - Grants the SDK permission to access a more accurate location based on GPS.
* ACCESS\_COARSE\_LOCATION (recommended) - Grants the SDK permission to access approximate location based on cell tower.
* READ\_PHONE\_STATE (recommended) - Grants the SDK permission to read only access to phone state.
* READ\_CONTACTS (recommended) - Grants the SDK permission to read the user's contacts data.
* GET\_ACCOUNTS (recommended) - Grants the SDK permission to the list of accounts 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 | **"1234"** |
|  |  | 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 | **"30000"** |
|  |  | 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. |  |
|  |  |  |  |
| msdk:ad\_width | ad.setAd\_width() | The width of the advertisement | **"320dp"** |
|  |  | to request from the server. If |  |
|  |  | this is not set, the SDK requests |  |
|  |  | an ad of at least |  |
|  |  | [android:layout\_widt](http://www.google.com/url?q=http%3A%2F%2Fandroidlayout_width&sa=D&sntz=1&usg=AFQjCNHvSHl2EGmeAntsLGxyNppGtkcy9g)h |  |
|  |  |  |  |
| msdk:ad\_height | ad.setAd\_height() | The height of the view. | **"50dp"** |
|  |  |  |  |
| msdk:layer\_style | ad.setLayer\_style() | layerstyle=simple or geocities | **"geocities"** |
|  |  | or cursor or floater |  |
|  |  |  |  |
| msdk:padding | ad.setPadding() | padding,ex:padding=2px | **"2px"** |
|  |  |  |  |
| msdk:align | ad.setAlign() | align=left or align=right or | **"left"** |
|  |  | align=center |  |
|  |  |  |  |

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"

**Get the Code**

First, you'll need to set up a AdView. The only required method is setZoneId, but it's usually a good idea to set the ad size. Note that the height and width you specify here must match the size of the ad zone. The code below shows a banner ad (assuming that your zone ID points at a 300x50 zone).

This simple example doesn't take advantage of all of the capabilities provided by the SDK – for example, you can also pass in the user's age and gender, as well as whether an ad click should open the device's native browser.

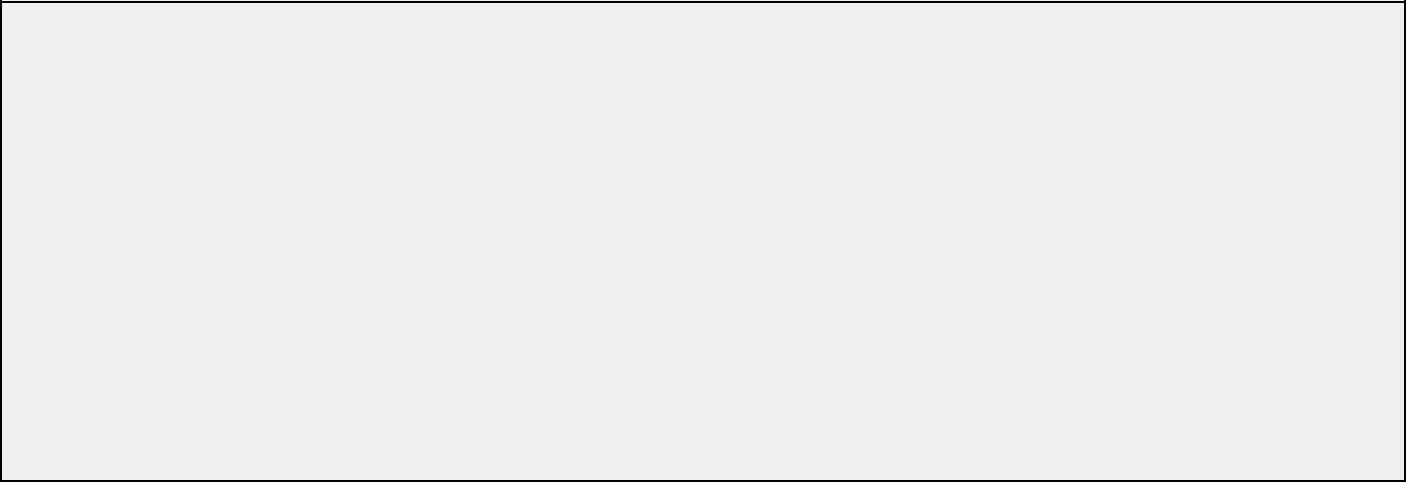
**Sample code :**

Xml Format:



<com.dragonmedia.adserver.AdView android:id="@+id/adView1" android:layout\_width="wrap\_content" android:layout\_height="wrap\_content" android:layout\_marginBottom="20dp" msdk:zone\_id="115" //Place Your Zone id msdk:ad\_width="468" //Place Your Ad width msdk:ad\_height="60" //Place Your Ad Height msdk:auto\_refresh\_time="30"> //Interval in Milliseconds </com.dragonmedia.adserver.AdView>

Code Format:



LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(LinearLayout.LayoutParams.MATCH\_PARENT,LinearLayout.Layout Params.WRAP\_CONTENT);

LinearLayout adFrame = (LinearLayout) findViewById(***R.id.preview***);

AdView ad = new AdView(this);

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

ad.setAd\_width("300"); //Place Your Ad width (Optional)

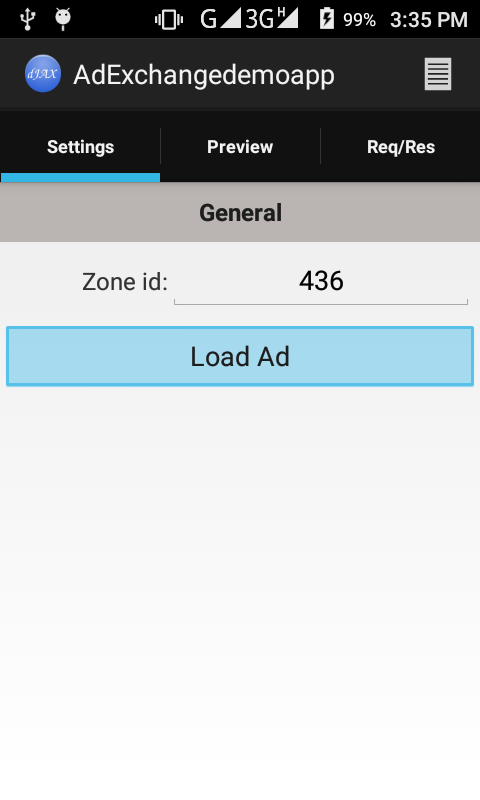
ad.setAd\_height("250"); //Place Your Ad Height (Optional)

ad.setAuto\_refresh\_time(45000); //Interval in Milliseconds

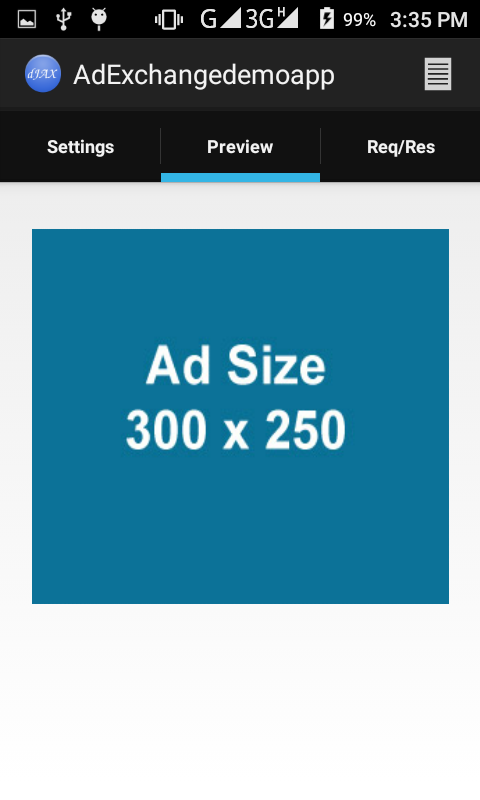
/\*Loads an Ad\*/ ad.LoadAd(); ad.setLayoutParams(params); adFrame.addView(ad); adFrame.bringToFront();

**After Installation:**

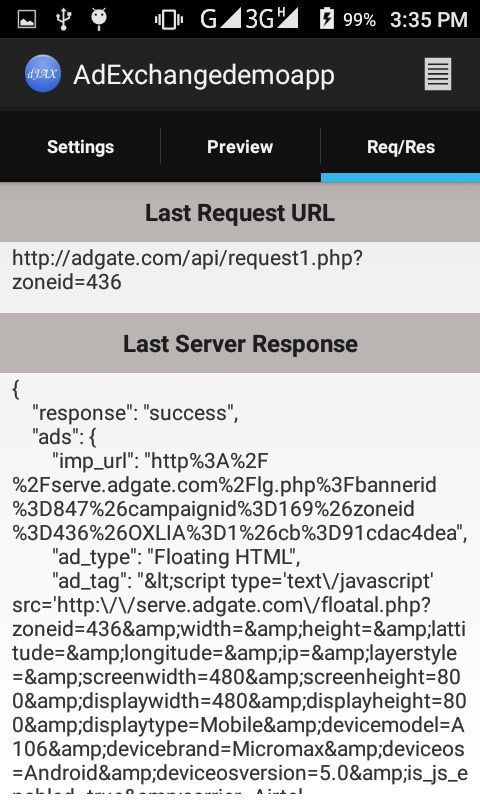
1. Enter the zone id then click load ad button.



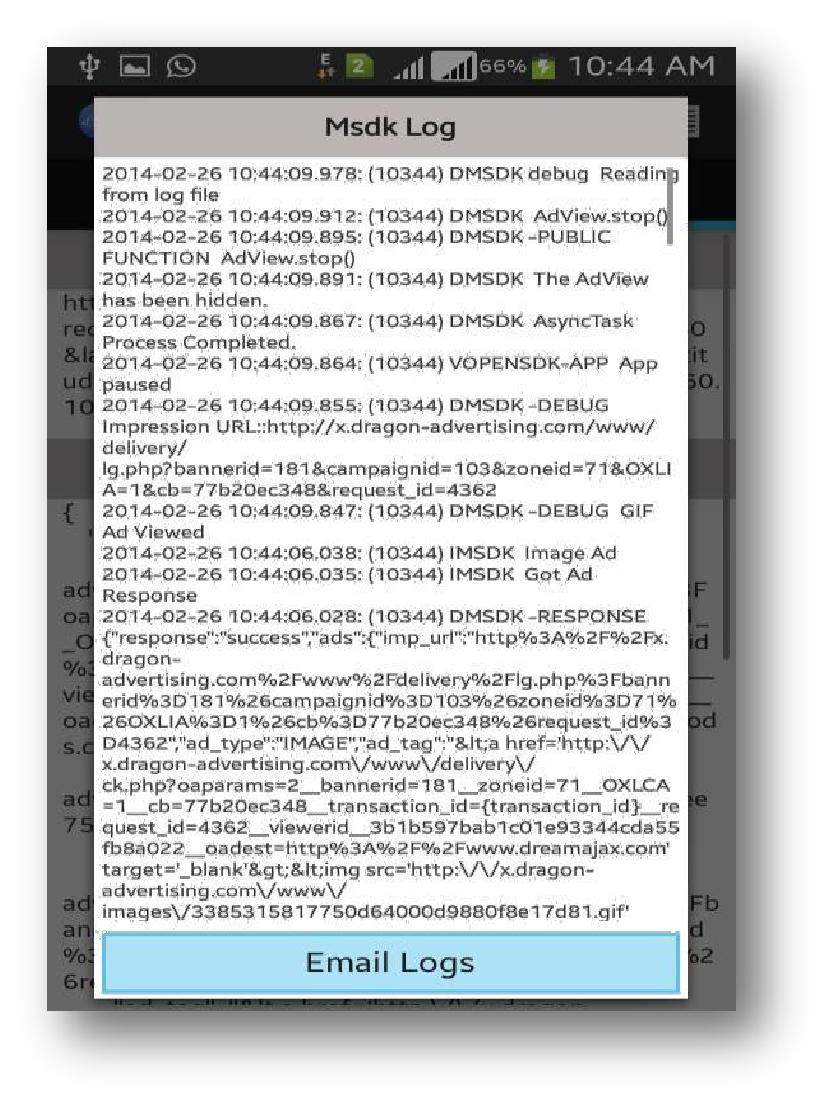
2. The requested banner is loaded in the preview tab as shown below.



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.