



Android SDK v 2.0
SDK Integration Guide
Publishers



Android SDK v2.0

SDK Integration Guide

Publishers



Introduction

This document outlines the Android SDK v2.0 that allows displaying mobile ads and mobile editorials inside android applications.

The current version of the Android SDK is compatible with Android 2.3 and above.

Today users expect the same high-quality experience regardless of device and if they don't get it, they blame your brand. This is why we use **Akamai number 1 CDN world wide** to display your ads.

1. Download Android SDK v1.0

Download the Ad4game Android SDK for publishers. Decompress the zip file and extract the files to your development computer. The Android SDK is provided as a single java JAR file (**AndroidSDKPublisher.jar**), making it easy to include in your Android project.

2. Configure AndroidManifest.xml

The SDK requires the following permissions. Put these in before closing the manifest tag.

Internet Permission (Mandatory):

The Internet permission is required to connect to publishing servers.

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Wifi State Permission (Recommended):

This permission enable the SDK to access information about whether you are connected to a Wi-Fi network and obtain the device's MAC address.

```
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
```

Connection Type Permission (Recommended):

This permission enable the SDK to detect if you are connected from a Wi-Fi network or 3G network.

```
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```



Tasks Permission (Mandatory):

Allows the SDK to get information about the currently or recently running tasks to set the [refresh](#) value.

```
<uses-permission android:name="android.permission.GET_TASKS" />
```

Unique Device Identifier (Mandatory):

Allows the SDK to get a unique id per device. Put these lines before closing the application tag.

```
<service android:name="org.openudid.OpenUDID_service">
    <intent-filter>
        <action android:name="org.openudid.GETUDID" />
    </intent-filter>
</service>
```

3. Install the Android SDK

Add **AndroidSDKPublisher.jar** to your Android project's build path.

If you are using Eclipse, right click on your **project folder -> Properties -> Java Build Path -> Libraries -> Add JARs**. (As of ADT 17.0, you can simply place **AndroidSDKPublisher.jar** in a folder called "libs" in your project directory and

Eclipse will automatically include the jar as part of the build under the classpath container "Android Dependencies").

4. Display mobile ads

To display mobile ads inside your android application, you need to set the value of **"refresh"** variable to refresh banners. You'll then need to instantiate the A4GPublisher class to allow you to call the functions from the AndroidSDKPublisher.

Generally, the code is placed inside your main activity's onCreate(Bundle) method.



You will need to pass the layout ID to the **findViewById** method, and then call **loadZoneId("ZoneID")** method.

```
public class MyClass extends Activity {  
    public static int refresh = 30000; //30 seconds  
    @Override  
    protected void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        setContentView(R.layout.activity_a4g);  
        A4GPublisher widget = (A4GPublisher) findViewById(R.id.yourLayoutID);  
        widget.loadZoneId("ZoneID");  
    }  
}
```

In your **layout.xml** file, you need to include the properties of the layout where you want to display the banners as follows

```
<com.a4gpublisher.A4GPublisher  
    android:id="@+id/yourLayoutID"  
    android:layout_width="350dp"  
    android:layout_height="250dp"  
    ...  
</>
```

Note: You can add many zones as you want, just instantiate the A4GPublisher object to create many widgets.

Ex:

```
A4GPublisher widget1 = (A4GPublisher) findViewById(R.id.yourFirstLayoutID);  
widget1.loadZoneId("ZoneID");
```

```
A4GPublisher widget2 = (A4GPublisher) findViewById(R.id.yourSecondLayoutID);  
widget2.loadZoneId("ZoneID");
```

5. Display Android Editorials

To display Android Editorials inside your android application, you need to instantiate the A4GEditorial class to allow you to call the functions from the AndroidSDKPublisher.

you need to use one of the three methods (you can use all of them) bellow:

- 1- **showA4GEdito** : If you want to display the editorials once the activity's started.
- 2- **showA4GEditoWithStartTime** : If you want to display the editorials after certain time.
- 3- **showA4GEditoWithStartAndEndTime** : If you want to display the editorials after certain time and close them automatically after certain time.
- 4- **showA4GEditoOnClick** : If you want to display the editorials when the user click on a button.

Note: The three first methods mentioned above are valid only if your project target android **3.0** or above, **showA4GEditoOnClick** is valid from android **2.3** or above.

To use one of the three methods mentioned above, you need to define a View and assign it to the A4GEditorial object like shown bellow:

```
protected void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.activity_main);  
    A4GEditorial a4gEdito = new A4GEditorial(getApplicationContext());  
    a4gEdito.setZoneid("zoneID");  
  
    View v = findViewById(R.id.yourView); // a textView or any composant that you use inside your activity  
    a4gEdito.setVue(v);  
  
    a4gEdito.showA4GEdito(getApplicationContext());  
    a4gEdito.showA4GEditoWithStartTime(getApplicationContext(), 5000);  
    a4gEdito.showA4GEditoWithStartAndEndTime(getApplicationContext(), 3000, 10000);  
}
```

To use **showA4GEditoOnClick** function, you need to define a button and assign it to the A4GEditorial object like shown below:

```
Button myButton = (Button)findViewById(R.id.yourButton);  
a4gEdito.setButton(myButton);  
a4gEdito.showA4GEditoOnClick(getApplicationContext());
```