admob

AdMob Android SDK

Installation Instructions

For upgrade instructions, see final page

AdMob Android SDK Installation Instructions

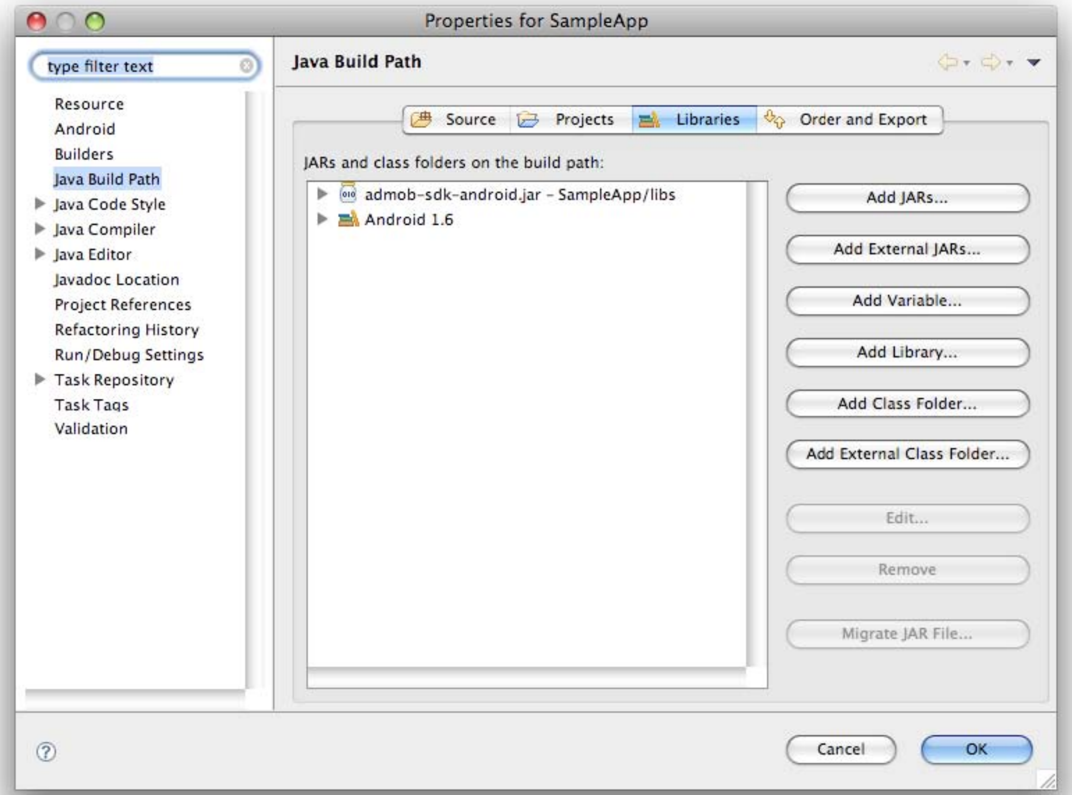
The AdMob Android SDK contains the code necessary to install AdMob ads in your application. This PDF will guide you through a simple XML implementation. For more technical information, please see the Javadocs that are included with the SDK download. If you're upgrading from an older AdMob SDK, please refer to the last page of this document for upgrade instructions.

Step 1 – Adding the JAR

In your project's root directory, create a subdirectory called `libs`. This will already be done for you if you used Android's `activitycreator` tool. Copy the AdMob JAR file (`admob-sdk-android.jar`) into that `libs` directory.

For Eclipse projects:

- Right-click on your project from the Package Explorer tab and select “Properties”
- Select "Java Build Path" from the left panel
- Select "Libraries" tab from the main window
- Click on "Add JARs..."
- Select the JAR that's been copied to the `libs` directory
- Click "OK" to add the SDK to your Android project



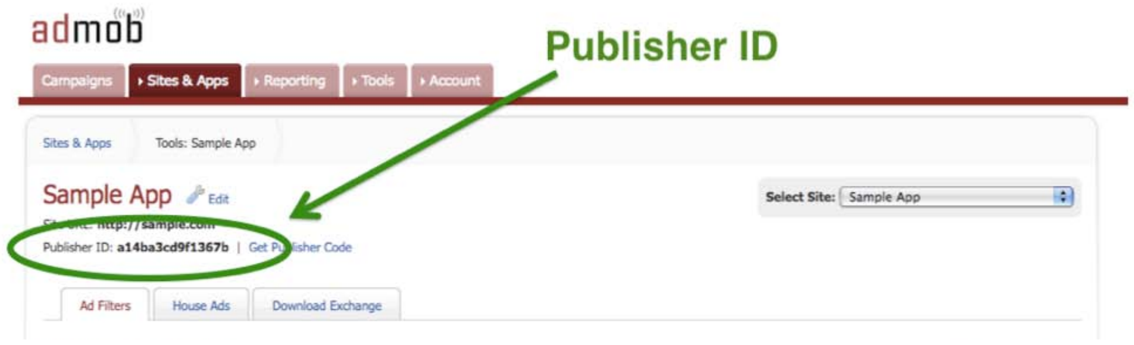
JARs and class folders on the build path:

admob-sdk-android.jar – SampleApp/libs

Android 1.6

Step 2 – Editing Your Manifest File

First you will need to note your AdMob publisher ID. It was given to you when registering your Android application on www.admob.com. It is a 15-character code like *a1496ced2842262* and can be found by locating your app in the Sites & Apps tab of your AdMob account and clicking "Manage Settings":



Just before the closing `</application>` tag of your AndroidManifest.xml file, you will need to add three things:

1. Your publisher ID from admob.com
2. Install tracking receiver
3. AdMobActivity for in-app landing pages

Copy and paste the following XML just before the closing `</application>` tag to add these three elements, replacing `YOUR_ID_HERE` with your 15-character publisher ID:

```
<!-- The application's publisher ID assigned by AdMob -->
<meta-data android:value="YOUR_ID_HERE" android:name="ADMOB_PUBLISHER_ID" />

<!-- AdMobActivity definition -->
<activity android:name="com.admob.android.ads.AdMobActivity"
android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
android:configChanges="orientation|keyboard|keyboardHidden" />

<!-- Track Market installs -->
<receiver android:name="com.admob.android.ads.analytics.InstallReceiver"
android:exported="true">
 <intent-filter>
   <action android:name="com.android.vending.INSTALL_REFERRER" />
 </intent-filter>
</receiver>

</application>
```

Note: If you are using Google Analytics, use the code above, and do not separately list the Google Analytics receiver in your Manifest (package name: com.google.android.apps.analytics.AnalyticsReceiver). AnalyticsReceiver will be called by InstallReceiver in the code you've just added here.

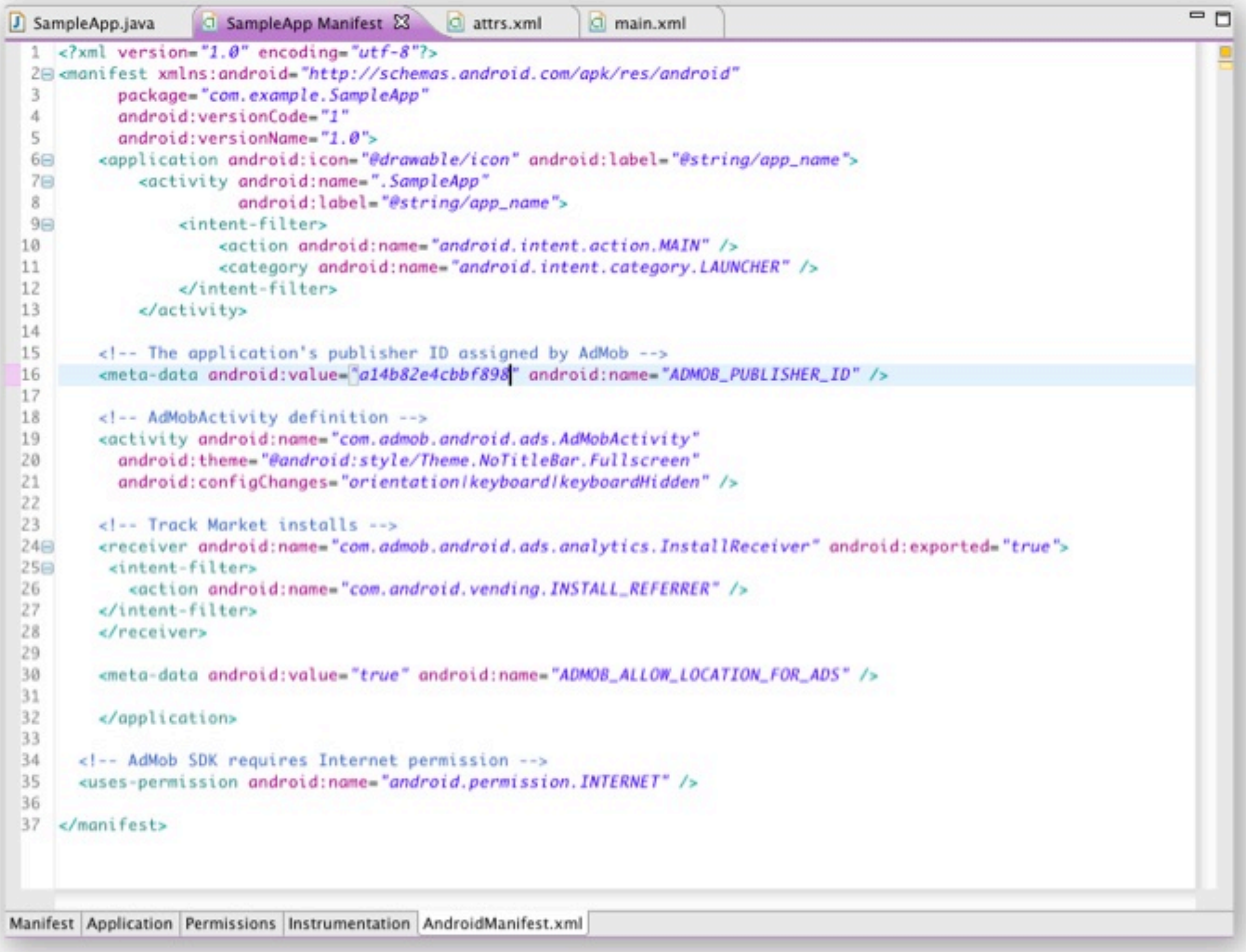
Make sure Internet permission is included. If it isn't already included, paste this XML after the `</application>` tag and before the closing `</manifest>` tag:

```
</application>

<!-- AdMob SDK requires Internet permission -->
  <uses-permission android:name="android.permission.INTERNET" />

</manifest>
```

If your application has `ACCESS_COARSE_LOCATION` or `ACCESS_FINE_LOCATION` permission, optionally instruct the AdMob SDK whether to use location for targeting ads. The default is not to use location and to log a warning asking you to explicitly opt in or opt out. Either call `AdManager.setAllowUseOfLocation` before making ad requests or insert the following line in AndroidManifest.xml just before the closing `</application>` tag with either `"true"` or `"false"`:

```
<meta-data android:value="true" android:name="ADMOB_ALLOW_LOCATION_FOR_ADS"
/>

</application>
```

Your final `AndroidManifest.xml` may look something like this:



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.example.SampleApp"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".SampleApp"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    <!-- The application's publisher ID assigned by AdMob -->
    <meta-data android:value="a14b82e4cbbf898" android:name="ADMOB_PUBLISHER_ID" />

    <!-- AdMobActivity definition -->
    <activity android:name="com.admob.android.ads.AdMobActivity"
      android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
      android:configChanges="orientation|keyboard|keyboardHidden" />

    <!-- Track Market installs -->
    <receiver android:name="com.admob.android.ads.analytics.InstallReceiver" android:exported="true">
     <intent-filter>
       <action android:name="com.android.vending.INSTALL_REFERRER" />
    </intent-filter>
    </receiver>

    <meta-data android:value="true" android:name="ADMOB_ALLOW_LOCATION_FOR_ADS" />

    </application>

  <!-- AdMob SDK requires Internet permission -->
  <uses-permission android:name="android.permission.INTERNET" />

</manifest>
```

Step 3 – Adding AdMob AdView Attributes to Your attrs.xml File

The attrs.xml file specifies custom AdView attributes in XML layout files. Paste the following into your attrs.xml file:

```
<declare-styleable name="com.admob.android.ads.AdView">
      <attr name="backgroundColor" format="color" />
      <attr name="primaryTextColor" format="color" />
      <attr name="secondaryTextColor" format="color" />
      <attr name="keywords" format="string" />
      <attr name="refreshInterval" format="integer" />
</declare-styleable>
```

If your project does not already have an `attrs.xml` file, then create one in the `/res/values/` directory of your project, and paste the following:

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
      <declare-styleable name="com.admob.android.ads.AdView">
            <attr name="backgroundColor" format="color" />
            <attr name="primaryTextColor" format="color" />
            <attr name="secondaryTextColor" format="color" />
            <attr name="keywords" format="string" />
            <attr name="refreshInterval" format="integer" />
      </declare-styleable>
</resources>
```

Step 4 – Editing Your Layout

Create a reference to the `attrs.xml` file in your layout element by adding an `xmlns` line that includes your package name (your package name is specified in `AndroidManifest.xml`). For example, if your package name were `com.example.SampleApp`, you would include this line:

```
xmlns:myapp="http://schemas.android.com/apk/res/com.example.SampleApp"
```

For a simple screen with only an ad, your layout element would look like this:

```
<?xml version="1.0" encoding="utf-8"?>
  <LinearLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
      xmlns:myapp="http://schemas.android.com/apk/res/com.example.SampleApp"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent">

      <com.admob.android.ads.AdView
          android:id="@+id/ad"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          myapp:backgroundColor="#000000"
          myapp:primaryTextColor="#FFFFFF"
          myapp:secondaryTextColor="#CCCCCC"
      />
</LinearLayout>
```

AdMob AdViews can be included in any type of XML layout. If you'd prefer to create your AdView programmatically, please see the Javadoc for a description of relevant constructors.

Step 5 – Setting Test Mode

When integrating AdMob ads into your application it is recommended that you use test mode. In test mode, ads are always returned.

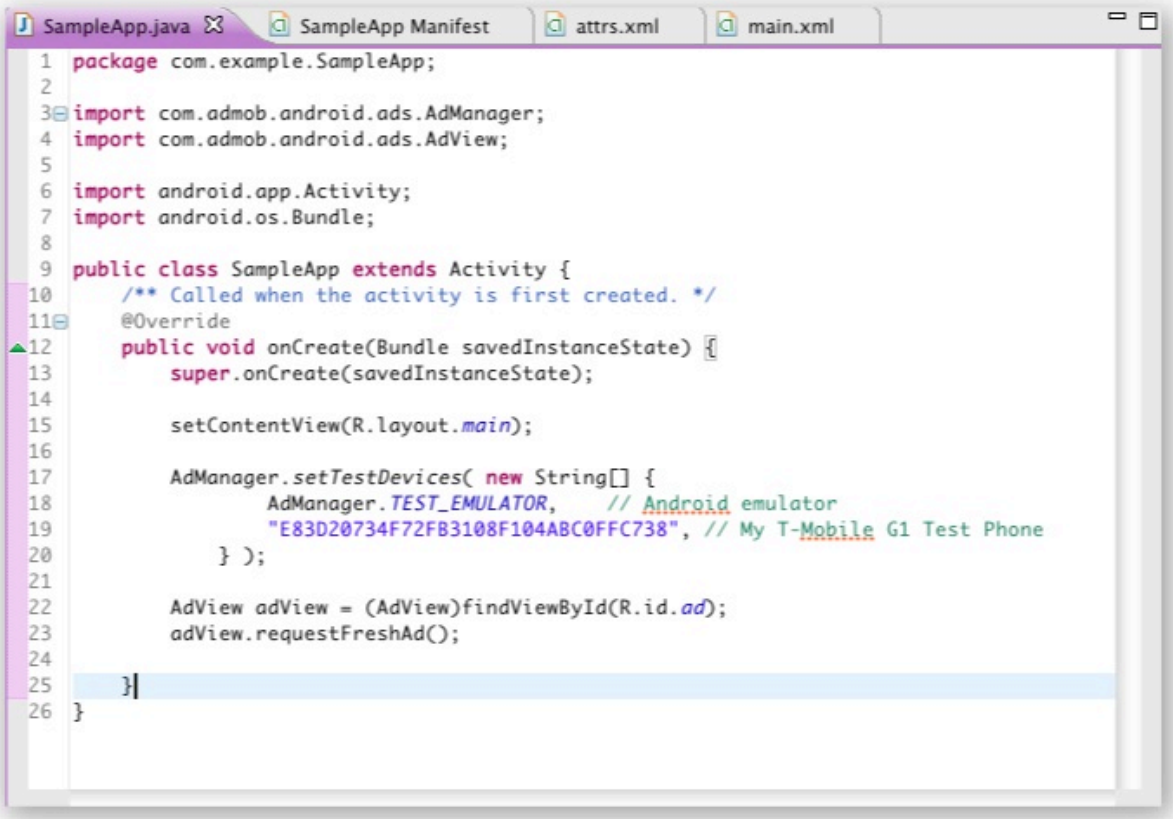
Test mode is enabled on a per-device basis. To enable test mode for a device, first request an ad, then look in LogCat for a line like the following:

```
To get test ads on the emulator use AdManager.setTestDevices...
```

Once you have copied the device ID from this LogCat line, you can enable test mode by calling `AdManager.setTestDevices` as follows:

```
AdManager.setTestDevices( new String[] {
      AdManager.TEST_EMULATOR,      // Android emulator
      "E83D20734F72FB3108F104ABC0FFC738", // My T-Mobile G1 Test Phone
         } );
```

A very simple class file might look like this:



```
package com.example.SampleApp;

import com.admob.android.ads.AdManager;
import com.admob.android.ads.AdView;

import android.app.Activity;
import android.os.Bundle;

public class SampleApp extends Activity {
    /** Called when the activity is first created. */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.main);

        AdManager.setTestDevices( new String[] {
                AdManager.TEST_EMULATOR,     // Android emulator
                "E83D20734F72FB3108F104ABC0FFC738", // My T-Mobile G1 Test Phone
            } );

        AdView adView = (AdView)findViewById(R.id.ad);
        adView.requestFreshAd();

    }
}
```

When you disable test mode, live ads may not be returned immediately for requests from a new publisher ID or one that has not been used in the past 24 hours. You should begin seeing more live ads returned about 15 minutes after your initial live ad requests.

Performance generally becomes more consistent once your app is in wider distribution and ad requests are frequently being made from a variety of devices.

Tips for Ad Placement

Ads are best placed at natural break points in your application. Correctly answering "when" and "where" to show ads can significantly increase revenue. Users are most likely to click on ads at the end of some activity. For example, ads in games perform best on game-over screens and worst when shown during game play.

Example Project – Lunar Lander

Included with this SDK is a Lunar Lander example project. This is the same example found in the Android SDK except that it shows an AdMob ad when the game is paused or finished.

Debugging

If something is going wrong, the first step is to look in Android's LogCat window. Make sure the Android Eclipse plug-in is installed. Then open it from the menu `Window -> Show View -> LogCat`. All AdMob logging is done using the tag "`AdMobSDK`".

Additional information can be found in the Javadoc and sample project contained within the AdMob SDK.

You can connect with other developers using the AdMob Android SDK here:
http://groups.google.com/group/admob-publisher-discuss

If you have any other questions, please feel free to contact support@admob.com.

Upgrade Instructions

If you're upgrading from a previous version of the AdMob Android SDK, you'll need to swap in the new JAR file and also follow a few additional steps.

Step 1 – Swapping in the New JAR File

In Eclipse, simply delete the existing admob-sdk-android.jar file from your libs directory. Then drag and drop the newer JAR file into the libs directory. When prompted, select "Copy" rather than "Link to" this new JAR file.

Step 2 – Editing Your Manifest File

Just before the closing `</application>` tag of your AndroidManifest.xml file, add these lines of code:

```
<!-- AdMobActivity definition -->
<activity android:name="com.admob.android.ads.AdMobActivity"
android:theme="@android:style/Theme.NoTitleBar.Fullscreen"
android:configChanges="orientation|keyboard|keyboardHidden" />

<!-- Track Market installs -->
<receiver android:name="com.admob.android.ads.analytics.InstallReceiver"
android:exported="true">
 <intent-filter>
   <action android:name="com.android.vending.INSTALL_REFERRER" />
 </intent-filter>
</receiver>
```

Note: If you are using Google Analytics, use the code above, and do not separately list the Google Analytics receiver in your Manifest (package name: com.google.android.apps.analytics.AnalyticsReceiver). AnalyticsReceiver will be called by InstallReceiver in the code you've just added here.