Ripple Mobile SDK

Smart Media for Smarter Screens

Your How-to Guide

Using Ripple SDK Ads within Android App

Version: ripple1.0.0

Table of Contents

1. Introduction

1.1. Disclaimer

This document is provided "as-is." Information and views expressed in this document, including URL and other Internet Web site references, may change without notice. You bear the risk of using it.

This document does not provide you with any legal rights to any intellectual property in any Ripple product. You may copy and use this document for your internal, reference purposes.

1.2. App Market places

Various app marketplaces apply conditions on app developers that they must comply with, when participating or uploading applications to these marketplaces. Please note, the inclusion of this SDK does not transfer that responsibility and it remains the responsibility of the App developer to achieve compliance with any app marketplace conditions.

Where possible, our SDK include features and functions to support an App developers efforts with marketplace compliance and we strongly recommend including or utilizing those features where possible.

1.3. Before You Start

Starting as Ripple App publisher

Please send all details below to developer@ripple.ad to sign up to Ripple Ad Network:

- First Name
- Last Name
- Company Name
- Address
- Phone Number
- Your Email
- Country
- Your Domain Name
- Where will you be running our ads? (Enter mobile site or app address)

Your account will be processed within 1 working day so please bear with us. If approved, we will send your publisher log in details within 1 working day.

Each Ad has a unique alias, network Id, subNetwork Id that you should use for each ad request as mentioned in the documentation below.

Get the Ripple Ads SDK

The Mobile SDK provides you with all of the necessary tools for integrating in-app advertising for Android. You can get the Ripple SDK from the Ripple Publisher portal for App publisher including this guide and a sample app which can be used for showcase purposes and for testing along with the program library. You will need to add these files into your App project directory as indicated in the next section

Please note, minimum Android OS Platform version required is 2.3 to use this SDK. Please compile with API level 9 or greater.

Incorporating the SDK

It is recommended that you use the Eclipse IDE with the ADT Plugin to build your Android Apps:

- Once a new project created, make sure there a “libs” folder exists in your Project.
- If not, simply right-click on the Project and click New -> Folder and enter “libs” as the Folder name

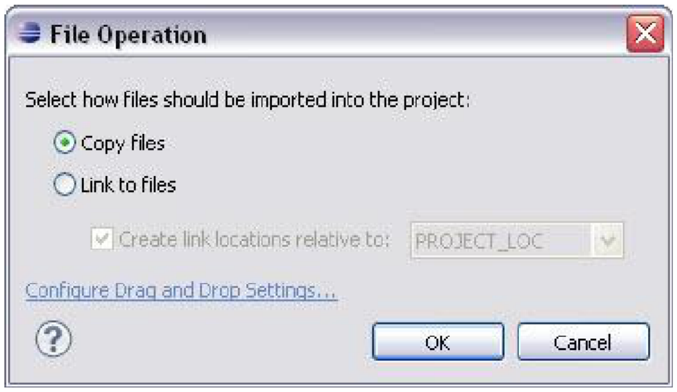
2. Quick Start: Ad Integration Guide

To add banner ad in your application, please follow steps below:

Step 1: Add the Ripple static library

To add the Ripple Ad SDK static library, follow these steps:

1. In the Eclipse IDE, simply drag the ripplead.jar file into your Project's "libs" folder.
2. When prompted "Select how files should be imported into the project:" and tick "Copy Files" option and click "OK"



File Operation

Select how files should be imported into the project:

[✓ Copy files] Link to files

Create link locations relative to: PROJECT_LOC

3. Then, right-click on your Project and click "Build Path" -> "Configure Build Path".
4. Under the window displayed, select "Libraries" tab. Click "Add JARs", then select the ripplead.jar file from the libs folder of your Project.

Step 2: Modifying AndroidManifest

The following permission is required for the Ads to be served:

- INTERNET

Make sure that these are defined in your Project's AndroidManifest.xml file

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

Optional permissions that may improve ad serving results / revenues:

- SEND_SMS
- CALL_PHONE
- WRITE_CALENDAR
- READ_CALENDAR
- ACCESS_FINE_LOCATION
- ACCESS_COARSE_LOCATION
- ACCESS_NETWORK_STATE
- WRITE_EXTERNAL_STORAGE

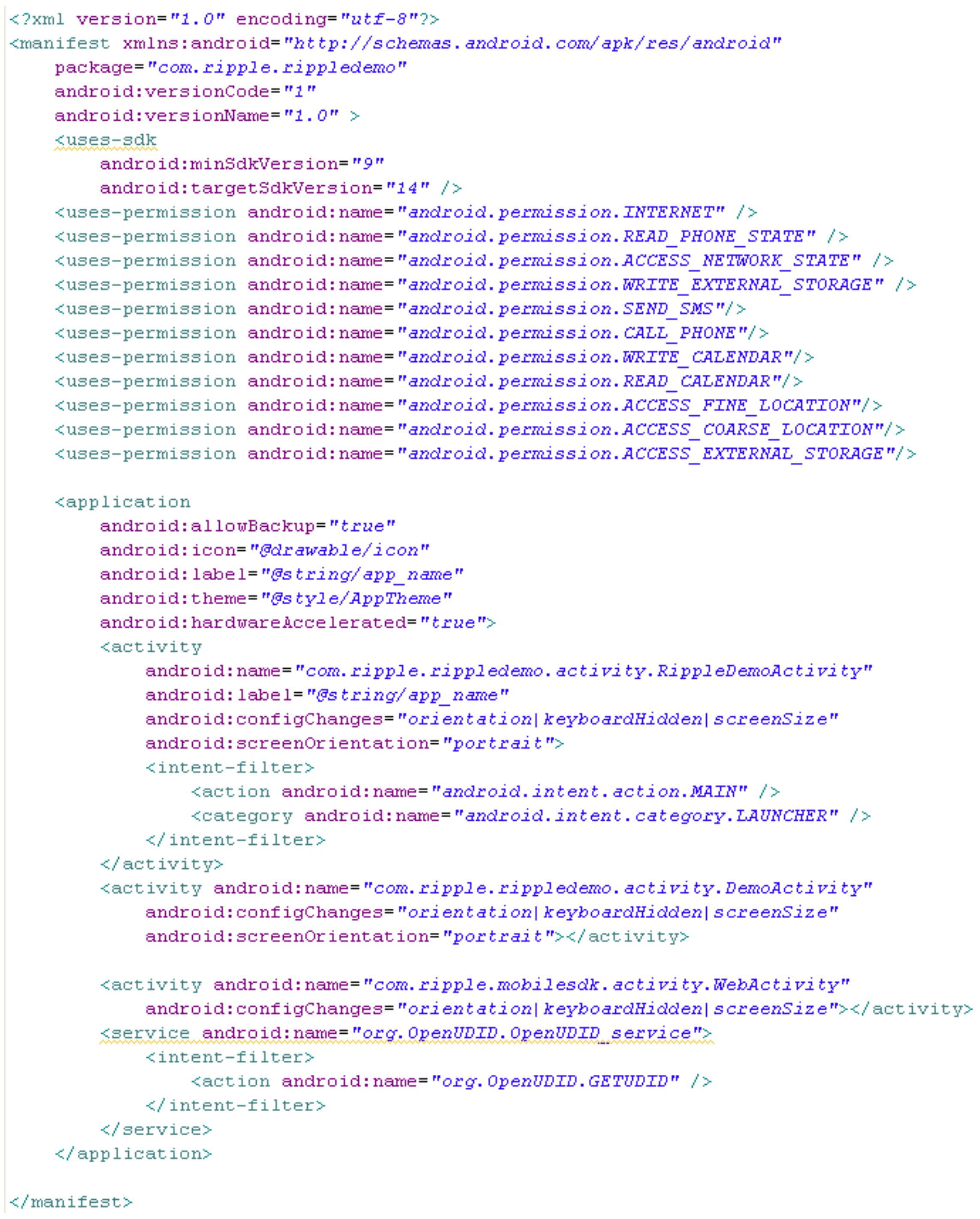
Please ensure these are setup in your Project's AndroidManifest.xml file, see the example shown below:

```
<uses-permission android:name="android.permission.SEND_SMS"/>
<uses-permission android:name="android.permission.CALL_PHONE"/>
<uses-permission android:name="android.permission.WRITE_CALENDAR"/>
<uses-permission android:name="android.permission.READ_CALENDAR"/>
<uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION"/>
<uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>
<uses-permission android:name="android.permission.
ACCESS_NETWORK_STATE"/>
<uses-permission android:name="android.permission.
ACCESS_EXTERNAL_STORAGE"/>
```

Add the following code inside the application:

```
<activity android:name="com.ripple.mobilesdk.activity.WebActivity"
android:configChanges="orientation|keyboardHidden|screenSize"></activity>
<service android:name="org.OpenUDID.OpenUDID_service">
    <intent-filter>
        <action android:name="org.OpenUDID.GETUDID" />
    </intent-filter>
</service>
```

Sample Manifest File:



```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.ripple.rippledemo"
    android:versionCode="1"
    android:versionName="1.0" >
    <uses-sdk
        android:minSdkVersion="9"
        android:targetSdkVersion="14" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.SEND_SMS"/>
    <uses-permission android:name="android.permission.CALL_PHONE"/>
    <uses-permission android:name="android.permission.WRITE_CALENDAR"/>
    <uses-permission android:name="android.permission.READ_CALENDAR"/>
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
    <uses-permission android:name="android.permission.ACCESS_EXTERNAL_STORAGE"/>

    <application
        android:allowBackup="true"
        android:icon="@drawable/icon"
        android:label="@string/app_name"
        android:theme="@style/AppTheme"
        android:hardwareAccelerated="true">
        <activity
            android:name="com.ripple.rippledemo.activity.RippleDemoActivity"
            android:label="@string/app_name"
            android:configChanges="orientation|keyboardHidden|screenSize"
            android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="com.ripple.rippledemo.activity.DemoActivity"
            android:configChanges="orientation|keyboardHidden|screenSize"
            android:screenOrientation="portrait"></activity>

        <activity android:name="com.ripple.mobilesdk.activity.WebActivity"
            android:configChanges="orientation|keyboardHidden|screenSize"></activity>
        <service android:name="org.OpenUDID.OpenUDID_service">
            <intent-filter>
                <action android:name="org.OpenUDID.GETUDID" />
            </intent-filter>
        </service>
    </application>

</manifest>
```

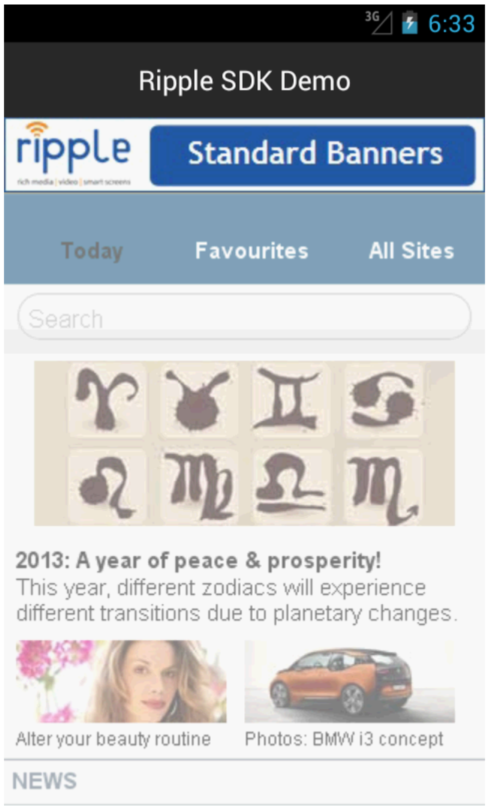
Step 3: Test your ads using the Test configuration

The simplest way to run a Ripple ad unit in your App is to add the code in your App's Activity XML as seen by the example code below:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:ads="http://schemas.android.com/apk/lib/com.ripple.ads"
   android:layout_width="fill_parent"
   android:layout_height="fill_parent">
  <com.ripple.mobilesdk.view.BannerView
           android:id="@+id/banner"
           android:layout_width="fill_parent"
           android:layout_height="wrap_content"
           ads:networkId="1333"
           ads:subnetworkId="1"
           ads:alias="RippleSDK_SB-bottom-5"
           ads:loadAdOnCreate="true"
           ads:refreshInterval="60"
        />
</RelativeLayout>
```

Step 4: Testing Banner Ad

Run your application and you should see something like the diagram below:



6:33
Ripple SDK Demo
ripple
Standard Banners
Today
Favourites
All Sites
Search
2013: A year of peace & prosperity!
This year, different zodiacs will experience different transitions due to planetary changes.
Alter your beauty routine
Photos: BMW i3 concept
NEWS

After you have tested your application please change this test configuration to your own publisher configuration as highlighted in "Step 3: Test your ads using the Test configuration". Then please re-run your application, to make sure that you can pull out live ads. Below is the parameters that must be changed:

Standard Banner Test Ad

Network Id: 1333

Sub Network Id: 1

Alias: RippleSDK_SB-bottom-5

3. Advance Banner Ad Integration Guide

3.1. Pause and Resuming Ads

Pause and Resume is required for hidden Ads, for example if an overlay is shown on-top of the ad format, we should pause the ad. Should the ad be visible again then the Application developer should call the resumeAd() function.

To pause the ad section simply call pauseAd() function on the BannerView instance in your App. Pausing an Ad means no ad request will be sent to the Ad servers even though the loadAd() has been called. To resume the ad section call resumeAd() function on the BannerView instance.

3.2. Refreshing Ads

It is possible to refresh ads by calling loadAd() function.

3.3. Implementing Event Listeners

For Ads, it is possible to listen for specific events with the Ad listener interface and write app specific code at these events. Event listeners exist for:

- Ad Load – When the ads are loaded on for the specific placement.
- NoAds – No Ads were returned from the server.

- Ad Switch – When the page is shown for the number of seconds configured in the ads:refreshInterval for the BannerView instance, a new ad will be requested. When this happens this trigger will be called.
- Ad Pause – This will be called after you have called the pauseAd() function mentioned above.
- Ad Resume – This will be called after you have called the resumeAd() function mentioned above.
- Ad Fail – This will be called when there is an error with the system, which may include loss of internet connection, and so forth.

Firstly, your main activity class needs to implement the AdListener class, as shown in the example below:

```
public class MainActivity extends Activity implements AdListener {
}
```

Once your class implements AdListener, All the abstract methods below will need to be implemented in your class, as shown below. To utilize these methods just add the app specific code you want to implement.

```
@Override
public void onAdsLoad() {
        System.out.println("On ads load");
}
public void onNoAds() {
        System.out.println("No ads available.");
}
public void onAdsSwitch() {
        System.out.println("On ads switch");
}
public void onAdsPause() {
        System.out.println("On ads pause");
}
public void onAdsResume() {
        System.out.println("On ads resume");
}
public void onAdsFail() {
        System.out.println("On ads fail");
}
```

Now, finally when initializing the BannerView class, you will need to pass the AdListener to it, as shown in the example below:

```
((BannerView) findViewById(R.id.banner)).setAdListener(this);
```

3.4. Using ProGuard

If you are planning to use ProGuard when exporting your app, you will need to add the following lines of code in your App's ProGuard config file – typically located in the root directory on your App and is named proguard.cfg or proguard-project.txt.

```
-keep class com.ripple.** {
        *;
}
```

4. Interstitial Ad Integration Guide

To add interstitial ad in your application, firstly you need to follow the steps 1 & 2 on section 4. Once steps 1 & 2 on section 4 is implemented, please follow the steps below:

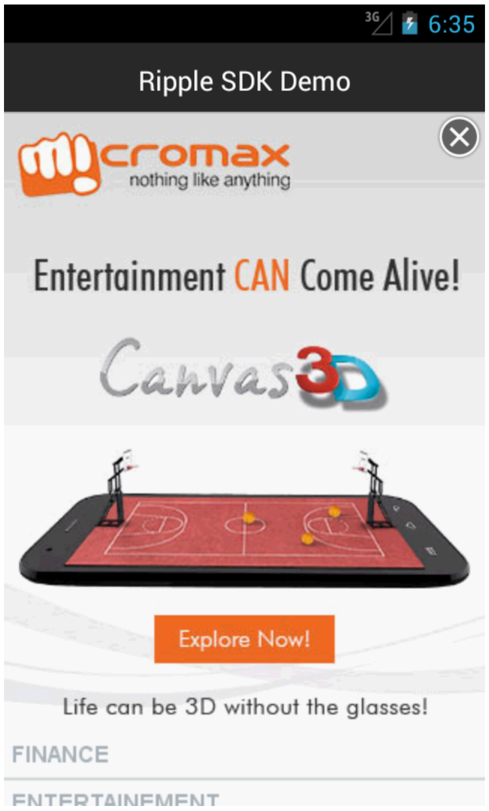
Step 1: Interstitial Ad Integration

The simplest way to run an interstitial ad unit in your App is to add the code in your App’s Activity XML as seen by the example code below:

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:ads="http://schemas.android.com/apk/lib/com.ripple.ads"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
   <com.ripple.mobilesdk.view.InterstitialView
            android:id="@+id/interstitial"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            ads:networkId="1333"
            ads:subnetworkId="1"
            ads:alias="RSDK_Inter-bottom-5"
            ads:loadAdOnCreate="true"
        />
</RelativeLayout>
```

Step 2: Testing Interstitial Ad

Run your application and you should see something like the diagram below:



After you have tested your application please change this test configuration to your own publisher configuration as highlighted in "Step 1: Interstitial Ad Integration". Then please re-run your application, to make sure that you can pull out live ads. Below is the parameters that must be changed:

Interstitial Test Ad

Network Id: 1333

Sub Network Id: 1

Alias: RSDK_Inter-bottom-5

5. Wall Ad Integration Guide

To add wall ad in your application, firstly you need to follow the steps 1 & 2 on section 4. Once steps 1 & 2 on section 4 is implemented, please follow the steps below:

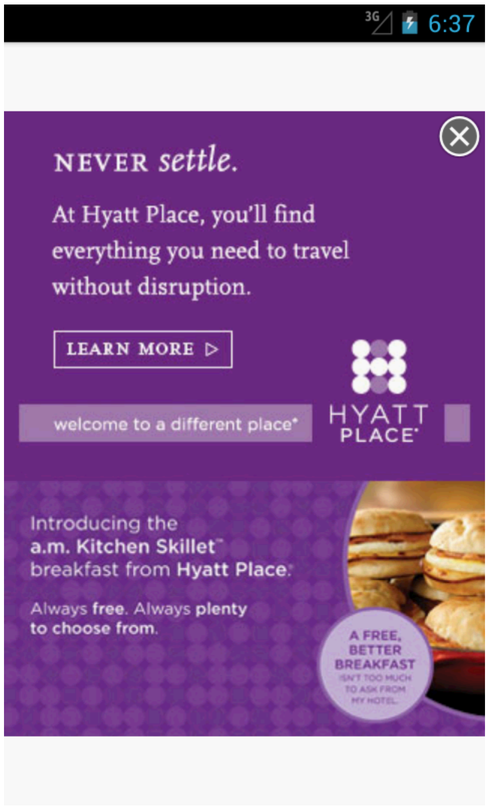
Step 1: Wall Ad Integration

The simplest way to run a wall ad unit in your App is to add the code in your App's Activity by the example code below:

```
public class RippeDemoActivity extends Activity {
    private AdWall mAdWall;
    protected void onCreate(Bundle savedInstanceState){
        super.onCreate(savedInstanceState);
        // Your initialization code here
        mAdWall=new AdWall(this);
        // Test configuration please change below with your own configuration
        // after testing.
        mAdWall.setAlias("RSDKWall-bottom-5");
        mAdWall.setNetworkID(1333);
        mAdWall.setSubnetworkID(1);
        // End test configuration
        mAdWall.loadAd();
    }
    protected void onDestroy(){
        mAdWall.loadAd();
        super.onDestroy();
    }
}
```

Step 2: Testing Wall Ad

Run your application and you should see something like the diagram below:


3G 6:37
NEVER settle.
At Hyatt Place, you'll find everything you need to travel without disruption.
LEARN MORE ▷
welcome to a different place®
HYATT PLACE®
Introducing the a.m. Kitchen Skillet™ breakfast from Hyatt Place.
Always free. Always plenty to choose from.
A FREE, BETTER BREAKFAST ISN'T TOO MUCH TO ASK FROM MY HOTEL.

After you have tested your application please change this test configuration to your own publisher configuration as stated in "Step 1: Wall Ad Integration". Then please re-run your application, to make sure that you can pull out live ads. Below is the parameters that must be changed:

Wall Test Ad

Network Id: 1333

Sub Network Id: 1

Alias: RSDKWall-bottom-5

6. Ripple Ads for Mobile FAQ

General Information

1. Q: What ads formats does the SDK support?

   A: This SDK compliant with IAB's Rich Media Standard MRAID 1.0 and MRAID 2.0.

Development

1. Q: Where can I find more help content specific to the Ripple network?

   A: You can contact our support team at support@ripple.ad to get more help

2. Q: I think I've implemented everything correctly, so why am I not seeing ads?

   A: While maintaining the highest possible fill rate is one of our top priorities, we may not always have an ad available for every ad request. This can be especially common during development, when ad requests are typically made infrequently from a small number of users and devices. When apps are newly registered on Ripple it may also take some time and several requests before impressions are consistently delivered. Developers generally see more consistent results once they have released their app and ad requests arrive more frequently from a more diverse user base.