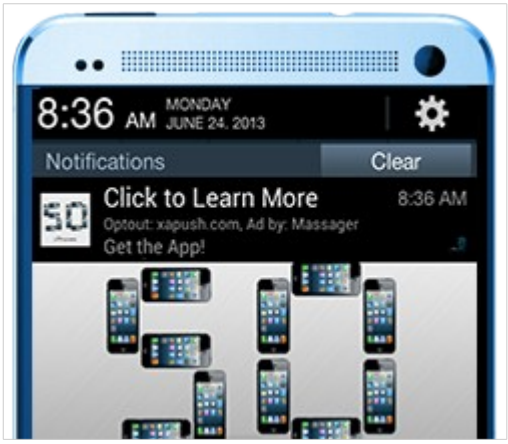
Airpush’s AirSDK 1.0 Universal is the first SDK to be designed specifically for use in marketplaces other than Google Play. It includes in-app as well as out-of-app monetization solutions such as Push Ads and Icon Ads. This SDK is perfect for driving the highest revenue streams possible from third party stores such as the following: AndAppOnline, AndroidPit App Center, Appland, Appslib, Soc.io Mall, Getjar, SnappCloud, F-Droid, Mobango, Mobile9, Moborobo, NexVa, Opera Mobile Store, AppTown, Pdassi, Apptoide, SlideME, Anzhi, Appchina, D.cn Games Center, gFan, HiAPK, N-Duo Market, PandaApp, Taobao App Market, Tencent App Gem, Yandex, and others.

Package Name

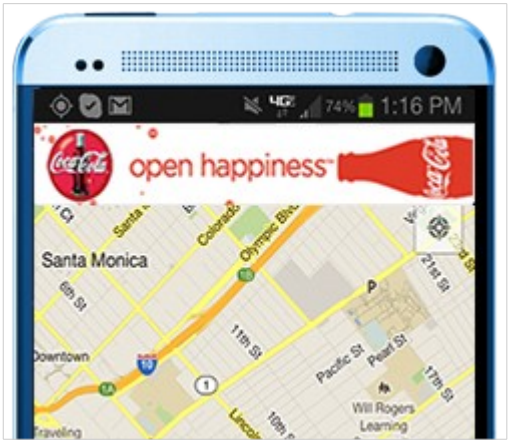
Your package Name : **com.airdsp.buzztouch**

Ad Units



Push Notification Ads

Icon Ads



In-App Banner Ads

Rich Media Banner Ads

SmartWall showing AppWall, Advanced Overlay, and Video ad formats

There are 5 types of ad units available in this SDK. To enable and optimize each of these ad units, select the corresponding check box next to each ad unit in 'Step 2' of adding your app. Free weekly payments are available for each.

Push Notification Ads

Push Ads are displayed in the notification trays of opted-in Android devices, which generates sky-high earnings and allows developers to monetize 100% of installs. With the highest overall ARPU of any mobile ad format, these ads have redefined revenue potentials for developers around the world.

Icon Ads
Icon Ads are sponsored shortcuts (“Icons”) placed on the homescreen of an Android device. Developers are paid for each Icon Ad placed on a device, and the format can be combined with other ad types to optimize revenue.

In-App Banner Ads
In-App Banner Ads are a staple of the mobile advertising world. Combined with the rest of Airpush’s industry leading ad types, In-App Banner Ads enable Android developers to monetize their users at every point in their mobile experience and maximize their revenue.

Rich Media Banner Ads
Rich Media Ads enable advertisers to deliver interactive content that drives dramatically more engagement than traditional static ads. This results in superior user experiences and most importantly, industry leading eCPMs for developers.

SmartWall
SmartWall is a revolutionary new interstitial format that dramatically outperforms all other others on the market. SmartWall’s patent-pending technology automatically mediates between Rich Media, Dialog, Video, AppWall, Advanced Overlays and more based on yield and network connection type!

Installation Instructions

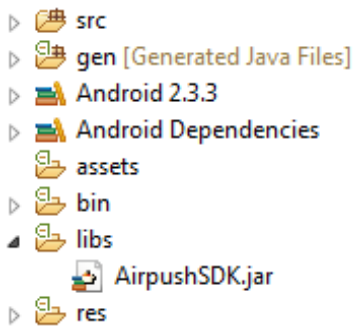
The Airpush Android SDK contains the code necessary to install Airpush ads in your application. This wiki will guide you through a simple XML implementation.

Airpush SDK Requirements:

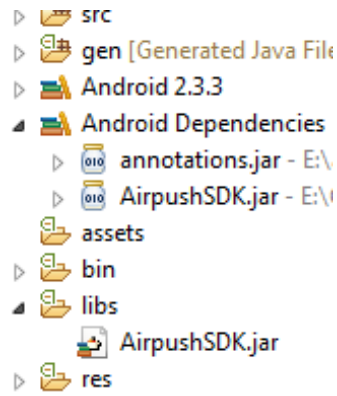
1.JDK 1.6 or later
2.Android 2.1 or later

Step 1 - Adding the JAR
For Eclipse Projects:

If you're using ADT Plugin 18 or later, copy the **com-airdsp-buzztouch.jar** to the "libs" folder of your project and move on to step 2.



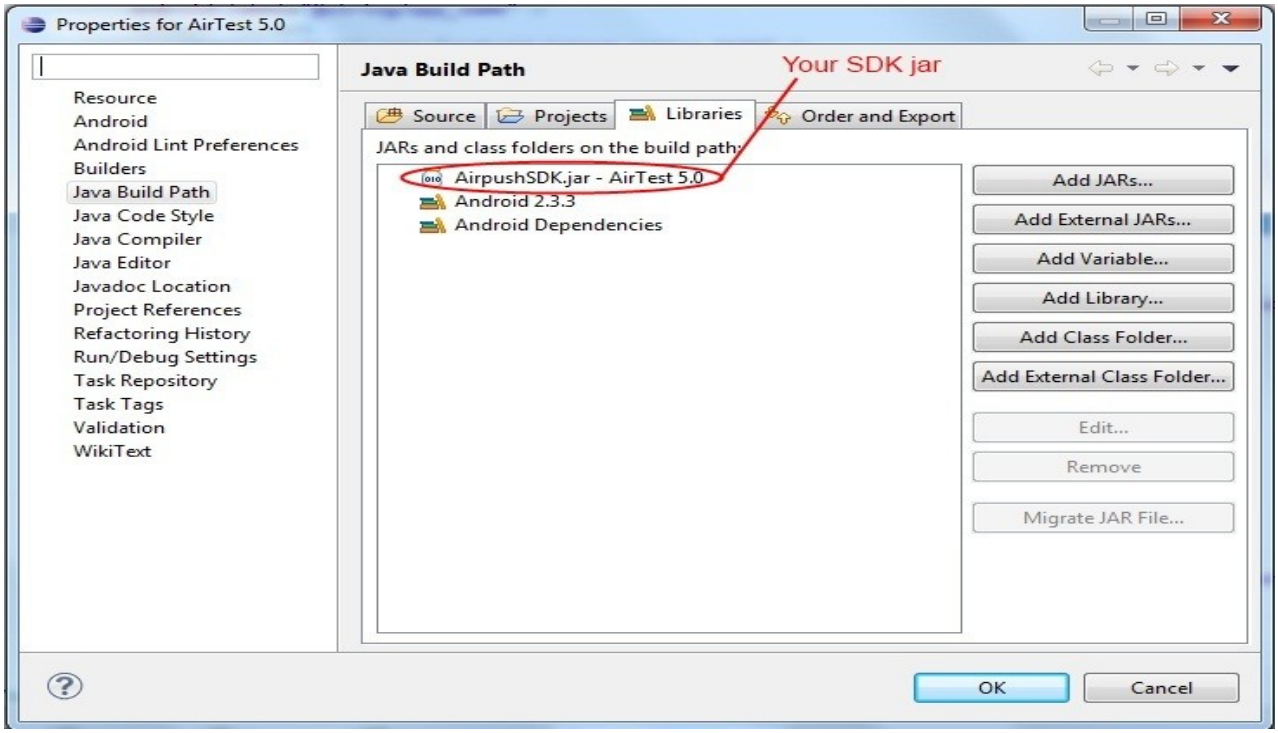
To verify whether it's added to your project, expand "Android Dependencies:"

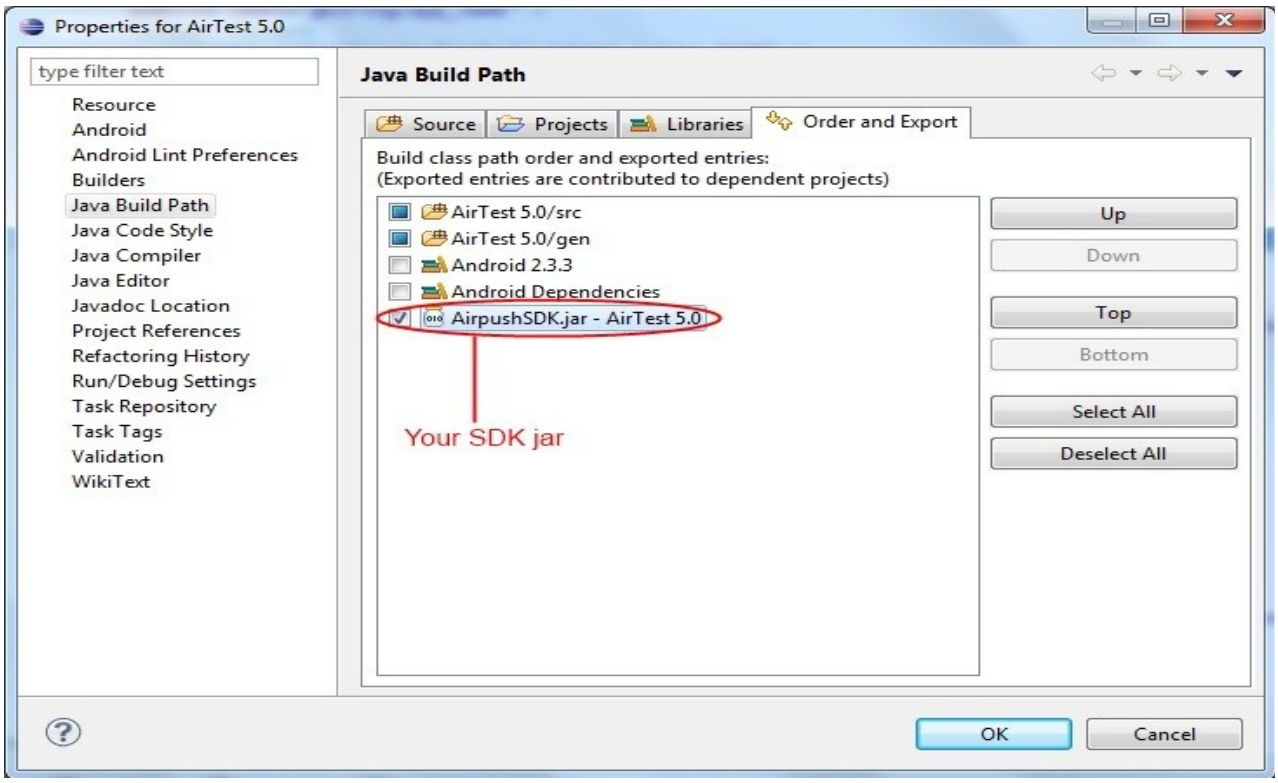


Use the option below if your ADT version is lower than 18.

Copy the **com-airdsp-buzztouch.jar** to your project's root directory.

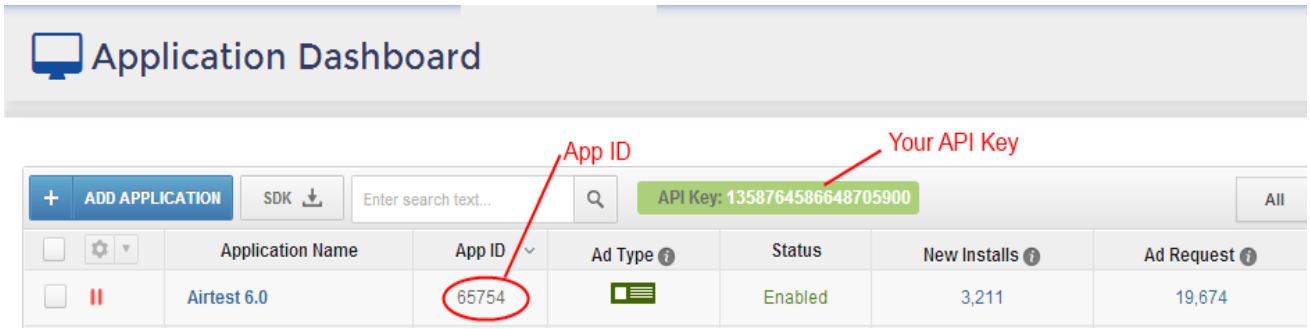
1.Right-click on your project from the Package Explorer tab and select "Properties"
2.Select "Java Build Path" from the left panel
3.Select the "Libraries" tab from the main window
4.Click on "Add JARs..."
5.Select the JAR that's been copied to the project's root directory
6.Click "OK" to add the SDK to your Android project
7.Select the "Order and Export" tab from the main window and check the SDK.





Step 2 - Editing Your Manifest File

First you'll need to note your Airpush <App Id> and <API Key>, which was given to you when registering your Android application on www.airpush.com. It's a numeric code that can be found by locating your app in the apps dashboard:



Just before the closing </application> tag of your AndroidManifest.xml file, you'll need to add the following:

1. Copy and paste the following XML

Placed just before the closing `</application>` tag:

Required declaration for all ads

```
<meta-data android:name="com.airdsp.buzztouch.APPID"
           android:value="<Your appId>" />
<meta-data android:name="com.airdsp.buzztouch.APIKEY"
           android:value="android*<Your ApiKey>"/>
<activity android:exported="false"
           android:name="com.airdsp.buzztouch.SmartWallActivity"
            android:configChanges="orientation|screenSize"
           android:theme="@android:style/Theme.Translucent" />
```

Required activity for SmartWall, rich media and in-app banner ads.

```
<activity android:name="com.airdsp.buzztouch.BrowserActivity"
          android:configChanges="orientation|screenSize" />
<activity android:name="com.airdsp.buzztouch.VideoAdActivity"
           android:configChanges="orientation|screenSize"
          android:screenOrientation="landscape"
           android:theme="@android:style/Theme.NoTitleBar.Fullscreen" >
</activity>
```

Required declarations for push notification ads.

```
<service android:name="com.airdsp.buzztouch.PushService"
         android:exported="false" />
<receiver android:name="com.airdsp.buzztouch" android:exported="false" >
  <intent-filter>
     <action android:name="android.intent.action.BOOT_COMPLETED" />
     <category android:name="android.intent.category.HOME" />
  </intent-filter>
</receiver>
```

2. Add The Following Permissions

Required permissions for all ads

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

Additional required permission for push notification.

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

Additional required permission for Icon Ad.

```
<uses-permission
android:name="com.android.launcher.permission.INSTALL_SHORTCUT" />
```

Additional required permission for Video Ad.

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

This permission is required for Video Ad but it's optional for other ad formats.

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

Optional permissions

(We strongly recommend you to add the optional permissions to enhance your Application earnings)

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission
android:name="com.android.browser.permission.READ_HISTORY_BOOKMARKS" />
```

Note: You do not need to add the above optional permissions if you are using Airpush COPPA complaint SDK. Please visit this link http://www.coppa.org for COPPA details.

Step 3 - Editing Your Main File

Inside "Activity," please add:

```
AirSDK airsdk=new AirSDK(getApplicationContext(), null, false);
```

Example:

```
import com.airdsp.buzztouch.AdCallbackListener;  //Add import statements
import com.airdsp.buzztouch.AirSDK;

public class MainActivity extends Activity{
private AirSDK airsdk; //Declare AirSDK here
@Override
protected void onCreate(Bundle savedInstanceState)
{
     super.onCreate(savedInstanceState);
     setContentView(R.layout.ad);
     if(airsdk==null)
        airsdk=new AirSDK(getApplicationContext(), null, false);
}
```

Parameters

1.Context context
2.AdCallbackListener adCallbackListener
3.**boolean enablecaching** This parameter will enable caching for SmartWall Ads. When you will use a SmartWall ad call it cached the ad locally. To show the ad you need to call`airsdk.showCachedAd(getApplicationContext(), adType);`.

Caching

This is a new feature which we have added it in this version for better user experience. You can cache an ad before displaying it to user. To enable cache pass the third parameter as`true` as mentioned above.

Example

```
if(airsdk==null)
airsdk=new AirSDK(this, adCallbackListener, true);   // ad caching is enabled

airsdk.startAppWall();   //this will start the AppWall ad but it will not show
you AppWall instantly.

@Override
public void onAdCached(AdType adType) {

//you will receive the information here if an ad is cached.

}
//now you can show the AppWall ad at any place within your app. You need to use
the following method:

airsdk.showCachedAd(this, AdType.appwall);
```

Note: You can use caching for Smartwall ad formats. Please use the above mentioned steps for caching;

1. To start push notifications, call the following method:

```
airsdk.startPushNotification(false);
```

Important: For push notifications, the developer needs to copy the `airsdk_notify.xml` file that's included along with the SDK download to the project layout folder. The developer must not make any changes to this file. We have updated this file so you must not use the

old `airsdk_notify.xml` file.

To use it in test mode please use the following code.

```
airsdk.startPushNotification(true);
```

Note: Please do not forget to set it `false` before uploading the app in Google play.

2. To start icon ads, call the following method:

```
airsdk.startIconAd();
```

3. Using SmartWall in your application:

Airpush's SmartWall is comprised of the following five sub Ad Formats:

1.Dialog Overlay Ad
2.AppWall Ad
3.Landing Page Ad
4.Rich Media Interstitial Ad
5.Video Ad

Airpush's ad server determines and displays the best sub-ad-format to maximize your revenue from Interstitial Ad Placements in your application. You just have to call the "airsdk.startSmartWallAd()" method wherever you want to show Airpush's Smartwall. For example: in the case of a gaming app, you can call "airsdk.startSmartWallAd()" between different levels and show a SmartWall app after a user clears each level. Airpush also offer flexibility for developers to choose a specific sub-ad-format from the above options to display by using the following methods within your code:

1.To start Overlay Dialog Ad: **airsdk.startOverlayAd();**
2.To start AppWall Ad: **airsdk.startAppWall();**
3.To start Landing Page Ad: **airsdk.startLandingPageAd();**
4.To start Rich Media Interstitial Ad: **airsdk.showRichMediaInterstitialAd();**
5.To start Video Ad: **airsdk.startVideoAd();**
Note: Although developers can choose to display a specific sub-ad-format, we highly recommend using the "airsdk.startSmartWallAd()" method which lets the Airpush ad server decide the best sub-format in order to maximize your earnings. We would also recommend showing Smartwall on App Launch and App Exit to maximize monetization. Here's sample code for showing Airpush's Smartwall on app exit:

If you want to show a SmartWall ad where the sub-format will be determined by SDK, please use the code below:

```
@Override
public void onBackPressed() {
    if (airsdk!=null) {
    airsdk.startSmartWallAd();
    }
super.onBackPressed();
}
```

For individual calls, use the below code:

```
@Override
public void onBackPressed() {
    if (airsdk!=null) {
        //Use only one from below.
        airsdk.startAppWall();
```

```
        airsdk.startOverlayAd();
        airsdk.startVideoAd();
        airsdk.startLandingPageAd();
        airsdk.showRichMediaInterstitialAd();
    }
 super.onBackPressed();
}
```

Note: This code can be used in the Activity file only.

If your application supports Android version 2.0 or below, please initialize the SDK as shown below:

```
if(android.os.Build.VERSION.SDK_INT >=7){
    AirSDK airsdk=new AirSDK(getApplicationContext(), null, true);
    airsdk.startPushNotification(false);     //start push notification.
    airsdk.startIconAd();                    //start icon.
    airsdk.startSmartWallAd();               //start smartwall ad.
}
```

4. AdCallbackListener:

In this SDK version we've added callback listeners for all ads, which can be called using the code below:

```
private AirSDK airsdk; //Declare Airpush here
     @Override
     protected void onCreate(Bundle savedInstanceState)
     {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.ad);
        if(airsdk==null)
        airsdk=new AirSDK(getApplicationContext(), adCallbackListener, true);

     }

     AdCallbackListener adCallbackListener=new AdCallbackListener() {

        @Override
        public void onSDKIntegrationError(String message) {
        //Here you will receive message from SDK if it detects any integration
issue.
        }

        public void onSmartWallAdShowing() {
        // This will be called by SDK when it’s showing any of the SmartWall ad.
        }

        @Override
        public void onSmartWallAdClosed() {
        // This will be called by SDK when the SmartWall ad is closed.
        }

        @Override
        public void onAdError(String message) {
        //This will get called if any error occurred during ad serving.
        }
        @Override
        public void onAdCached(AdType arg0) {
        //This will get called when an ad is cached.

        }
```

```
    };
```

Note: This listener doesn't cover Push Notification and Icon Ads, however the **onSDKIntegrationError** method will be called if any required steps are missing for any ad format.

Step 4 - MRAID 2.0 and Banners

This version of the SDK supports IAB MRAID 2.0 compliant Rich Media ads and banner ads. To show these ads in your app, you'll need to add the following entry into your **layout.xml**file (app layout file).

```
<com.airdsp.buzztouch.AdView
xmlns:ap="http://schemas.android.com/apk/res-auto"
android:id="@+id/myAdView"
android:layout_width="wrap_content"
android:layout_height="wrap_content"
ap:animation="fade"
ap:banner_type="inappad"
ap:placementType="interstitial"
ap:test_mode="false"
ap:canShowMR="false"
/>
```

banner_type: [String] There are three types of banner ads. You can use only one at a time, with the `"inappad"` banner type being the default.

1. **inappad:** Can show image/rich_media ads. If you want to show medium_rectangle banner also please use `canShowMR="true"`.
1. **canShowMR:** [Boolean] This will be used when `banner_type="inappad"`. If it's true then SDK can show medium rectangle banner.
2. **image:** Shows a banner with an image/text. Size[mobile: 320x50/468x60, tablet:728x90]
3. **medium_rectangle:** Shows an image banner of 300x250 size.
4. **rich_media:** Shows IAB MRAID 2.0 compliant Rich Media ads. To use this ad you need to provide “`placementType[String]."` There are two kinds of placement types:

1. **interstitial:** This shows an interstitial ad which is built using HTML5/JavaScript.
2. **inline:** This shows a banner with HTML5/JavaScript. Size[mobile: 468x60, tablet:728x90]

animation: [String] There are three kinds of animations which can be used to show these ads; [fade, top_down, left_to_right]. Fade is the default. This applies to banner ads only.

test_mode: [Boolean] This takes a boolean value.

You also need to place the **mraid_attrs.xml** file into your app's **res > values** folder. This file is included within the SDK download.

Use banner/Rich Media ads using Java code

```
/* AdView Class is same as View Class. You can use the Adview object as View
object. */
AdView adView=new AdView(this, AdView.BANNER_TYPE_IN_APP_AD,
AdView.PLACEMENT_TYPE_INTERSTITIAL, false, false,
     AdView.ANIMATION_TYPE_LEFT_TO_RIGHT);
     adView.setAdListener(this);
```

Please check Demo project for more details

Use MRAID Ad Callback Listener

You can use the code below in your Activity file:

```
@Override
```

```
protected void onCreate(Bundle savedInstanceState)
{
     super.onCreate(savedInstanceState);
     setContentView(R.layout.ad);
     AdView adView=(AdView)findViewById(R.id.myAdView);
     adView.setAdListener(adlistener);
}
AdCallbackListener.MraidCallbackListener adlistener = new
AdCallbackListener.MraidCallbackListener() {

     @Override
     public void onAdClickListener()
     {
     //This will get called when ad is clicked.
     }

     @Override
     public void onAdLoadedListener()
     {
     //This will get called when an ad has loaded.
     }

     @Override
     public void onAdLoadingListener()
     {
     //This will get called when a rich media ad is loading.
     }

     @Override
     public void onAdExpandedListner()
     {
     //This will get called when an ad is showing on a user's screen. This may
cover the whole UI.
     }

     @Override
     public void onCloseListener()
     {
     //This will get called when an ad is closing/resizing from an expanded
state.
     }

     @Override
     public void onErrorListener(String message)
     {
     //This will get called when any error has occurred. This will also get
called if the SDK notices any integration mistakes.
     }
     @Override
      public void noAdAvailableListener() {
        //this will get called when ad is not available

        }
};
```

Opt-in Dialog Callback Listener

We've also added a callback listener for the EULA dialog, which can be called using the code below:

```
AirSDK airsdk;
@Override
```

```
     protected void onCreate(Bundle savedInstanceState)
     {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.ad);
          //set opt-in listener
          AirSDK.setOptinListener(optinListener);
          if(airsdk==null)
          airpush=new Airpush(getApplicationContext(), adListener, true);
}

AdCallbackListener.OptinListener optinListener=new
AdCallbackListener.OptinListener() {

     @Override
     public void showingDialog() {
          //This will get called when EULA dialog is showing on screen.
     }

     @Override
     public void optinResult(boolean result) {
          //This will get called when EULA dialog is closed with a boolean
result.
     }
};
```

Using Progaurd

Keep options required for AirSDK SDK 1.0

```
# To enable ProGuard in your project, edit project.properties
# to define the proguard.config property as described in that file.
#
# Add project specific ProGuard rules here.
# By default, the flags in this file are appended to flags specified
# in ${sdk.dir}/tools/proguard/proguard-android.txt
# You can edit the include path and order by changing the ProGuard
# include property in project.properties.
#
# For more details, see
#   http://developer.android.com/guide/developing/tools/proguard.html

# Add any project specific keep options here:


-optimizationpasses 5
-dontusemixedcaseclassnames
-dontskipnonpubliclibraryclasses
-dontpreverify
-verbose
-optimizations !code/simplification/arithmetic,!field/*,!class/merging/*
-keepattributes *Annotation*

-injars      bin/classes
-injars      libs
-outjars     bin/classes-processed.jar

-keep public class * extends android.app.Activity
-keep public class * extends android.app.Application
-keep public class * extends android.app.Service
-keep public class * extends android.content.BroadcastReceiver
-keep public class * extends android.content.ContentProvider
-keep public class * extends android.preference.Preference
```

```
-keepclassmembers class **.SmartWallActivity$AppWall$JavaScriptInterface
 {
   *;
 }

 -keepclassmembers class **.MraidView$JavaScriptInterface
 {
   *;
 }
-keepclassmembers class **.OverlayAd$JavaScriptInterface
 {
   *;
 }
-keepclasseswithmembernames class *
 {
    native <methods>;
 }

 -keepclasseswithmembers class *
  {
    public <init>(android.content.Context, android.util.AttributeSet);
  }

 -keepclasseswithmembers class *
  {
    public <init>(android.content.Context, android.util.AttributeSet, int);
  }

  -keepclassmembers enum *
  {
    public static **[] values();
    public static ** valueOf(java.lang.String);
  }

  -keep class * implements android.os.Parcelable
   {
     public static final android.os.Parcelable$Creator *;
   }

   -keepclasseswithmembers class **.R$**
   {
    public static <fields>;
   }

   -keep class * extends android.view.View
    {

          public <init>(android.content.Context);
          public <init>(android.content.Context, android.util.AttributeSet);
          public <init>(android.content.Context, android.util.AttributeSet, int);
          void set*(***);
           *** get*();
    }

   -keepclassmembers class *
    {
            static final %               *;
            static final java.lang.String *;
    }
```

```
-keepattributes SetJavaScriptEnabled
-keepattributes JavascriptInterface
-keepattributes InlinedApi
```

Note: If you are using lower Android version for your app then please add the following line in above.

```
-dontwarn com.airdsp.buzztouch.**
```

Important Instructions

•For the best experience, please use the latest Android API as the build target.
•If you are upgrading/updating the SDK, please don't forget to clean & build the project after completing all steps successfully.
•Developers are no longer required to create a BootReceiver file explicitly. Only the Manifest declaration is required. The BootReceiver file is already present inside the SDK jar.
•Please enable the Smartwall ad type for your application under your account within the Airpush Portal to utilize our Interstitial Ads.

Sample Application Code and Support:

Included with this SDK is an Airtest Example Project (Airtest6.0), and in case of any issues integrating the SDK, please feel free to contact publishersupport@airpush.com.

Note: For the privacy of your users, Airpush never stores personally identifiable information. Our SDK encrypts all IMEI numbers using md5 hashing.