Mobclix SDK Integration Guide

Version 4.0.2

Rev. 6/6/12

Contents

1. Setting up the Mobclix SDK 1

1.1 What's Inside 1

1.2 Adding the Mobclix SDK file to your Project 1

1.3 Referencing the Mobclix SDK 1

2. Integrating with Your Code 2

2.1 Adding Mobclix Meta-data and Permissions to the Application 2

2.2 Integrating with Interface Builder 4

2.3 Integrating without Interface Builder 5

2.4 Integrating a Mobclix FullScreen AdView 6

3. Open Allocation 7

1. Introduction to Open Allocation 7

1.1 What is Open Allocation? 7

1.2 How does it work? 7

1.3 Ad Network Adapters 7

1.4 How do I set it up? 7

2. Dashboard Setup 8

2.1 Enabling Open Allocation 8

2.2 Configuring Allocation 10

3. Code Integration 11

3.1 Adding additional SDK's to your Application 11

3.2 Implementing MobclixAdView Event Methods 13

4. ProGuard 15

4.1 Configuring the Android Project 15

4.2 Configuring proguard.cfg 15

4.3 Configuring proguard.cfg with Open Allocation Google AdMob 17

4.4 Configuring proguard.cfg with Open Allocation Millennial Media 17

5. Troubleshooting 18

5.1 Common Problems 18

5.1.1 The message “An error has occurred” appears in the AdView. 18

5.1.2 Nothing is shown in the AdView. 18

6. Additional Support 19

1. Setting up the Mobclix SDK

1.1 What's Inside

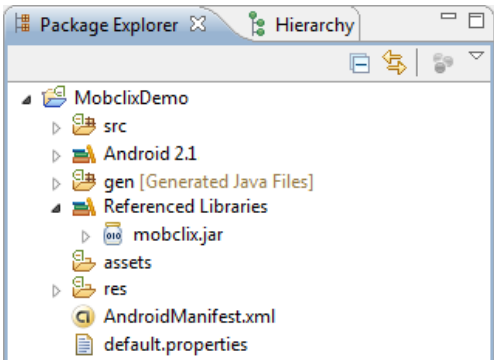
mobclix.jar	Contains the Mobclix for Android SDK
Demo Application	Fully functional Android application that integrates all features of the Mobclix SDK.
Integration Guide	This document that's designed to help new developers integrate the Mobclix for Android SDK into their applications

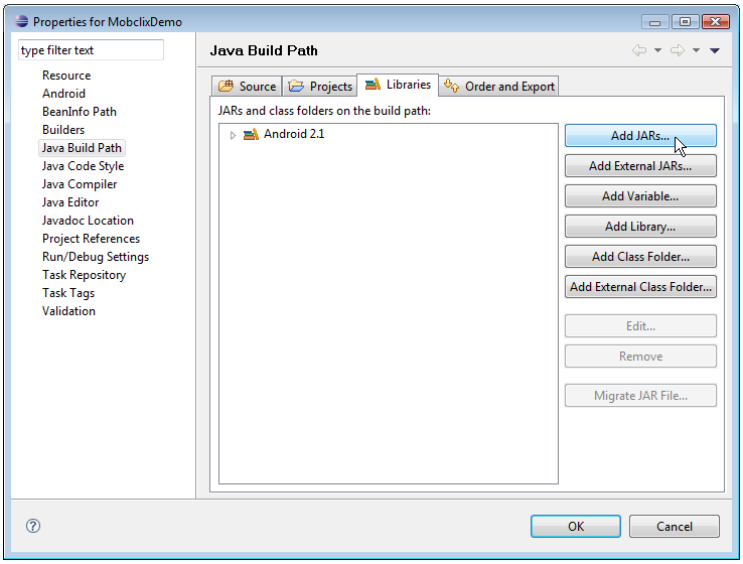
1.2 Adding the Mobclix SDK file to your Project

Drag the "**mobclix.jar**" file into your project's root directory.

1.3 Referencing the Mobclix SDK

Open up your Android project in Eclipse. Click on "**Properties**" in the "**Project**" menu bar. Select "**Java Build Path**" on the left and then select the "**Libraries**" tab. Click "**Add JARs...**" and select the "**mobclix.jar**". The "**mobclix.jar**" should now appear under your project's Referenced Libraries.





2. Integrating with Your Code

2.1 Adding Mobclix Meta-data and Permissions to the Application

Open your application's "**AndroidManifest.xml**". Inside the **<application>** flags, add the following:

1. **APPLICATION_ID**
   Mobclix Application ID, found in the Mobclix Dashboard by clicking the "**Account**" tab and then clicking into the "**Applications**" section. If you haven't added your application to Mobclix, now would be a great time to do so. You can follow the steps found here, and you'll receive your Application ID when you're finished adding it.

2. **MobclixBrowserActivity**
   An activity used by the Mobclix SDK to display certain rich media ads and fullscreen advertisements.

Outside of the **<application>** flags, add or make sure your application has the following permissions:

1. **INTERNET**
   Used by the Mobclix Android SDK to fetch advertisements and relay analytics information.

2. **READ_PHONE_STATE**
   Used to access a unique identifier that some ad networks require to track impressions (See:http://developer.android.com/reference/android/telephony/TelephonyManager.html#getDeviceId()). The reason some of them choose not to use ANDROID_ID for unique identification is this bug: http://code.google.com/p/android/issues/detail?id=10603

3. **ACCESS_NETWORK_STATE** (Optional)
   Used to determine the type of network state for analytics and optimized ad serving.

For applications that support Android tablets (Android SDK Version 3.0+), also add **android:hardwareAccelerated="true"** to the application flag. If this causes issues with your application, at least add **android:hardwareAccelerated="true"** to the MobclixBrowserActivity. This flag is used to ensure that HTML5 Video ads play on Android tablet devices. This flag also requires that you set the build target to Android SDK Version 3.0+. However, you can still set **minsdkversion** to the oldest version that you want to support.

See the following code as an example. The Mobclix related code is highlighted in **green**.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.mobclix.demo" android:versionName="1.0.0"
    android:versionCode="1">
  <application android:icon="@drawable/icon"
      android:label="@string/app_name"
      android:debuggable="true"
      android:hardwareAccelerated="true" >
    <activity android:name=".MobclixDemo"
        android:label="@string/app_name">
      ...
    </activity>
    ...

    <meta-data android:name="com.mobclix.APPLICATION_ID"
        android:value="insert-your-application-key"/>
    <activity
        android:name="com.mobclix.android.sdk.MobclixBrowserActivity"
        android:theme="@android:style/Theme.Translucent.NoTitleBar"
        android:hardwareAccelerated="true" />

  </application>
  ...

  <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" />
</manifest>
```

2.2 Integrating with Interface Builder

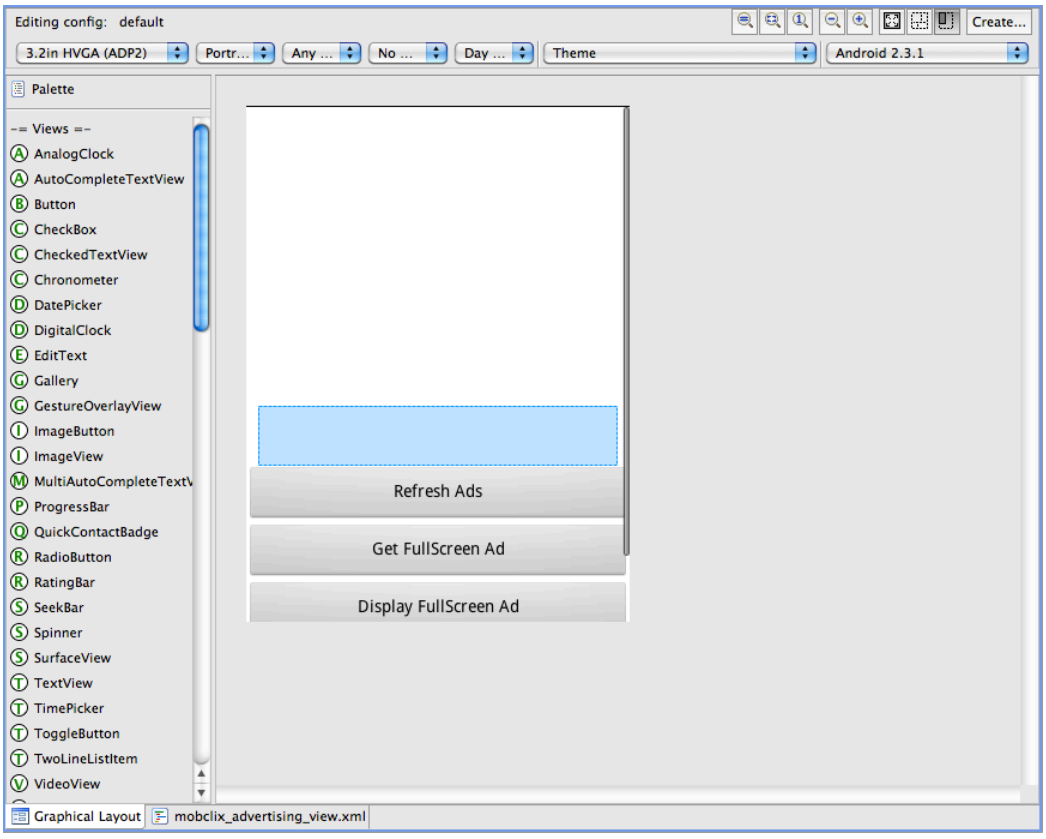
1. Instantiate the MobclixMMABannerXLAdView into your page's layout xml:

```
<com.mobclix.android.sdk.MobclixMMABannerXLAdView
    android:id="@+id/banner_adview"
    android:layout_width="320dip"
    android:layout_height="50dip"
    android:layout_gravity="center" />
```

2. Adding this code to the xml will create a rectangle in your layout editor representing the MobclixAdView for you to position.

   2.1. Finish laying out the MobclixAdView in your page's layout.

   2.2. You can now save and close your page's xml file. Your application is now all set to receive ads. The MobclixAdView you've added will automatically refresh based on the refresh rate that you can set here on the Mobclix Developer Dashboard.



2.3 Integrating without Interface Builder

For more advanced implementations of ads, you'll want to create and place the views in your code. We're going to show you a basic implementation in your Activity.

Import the MobclixAdView classes to your Activity , as shown below.

```
import com.mobclix.android.sdk.MobclixAdView;
import com.mobclix.android.sdk.MobclixMMABannerXLAdView;
```

Then, all you need to do is create the view, and add it to your view hierarchy, as shown below.

```
@Override
public void onCreate(Bundle savedInstanceState) {
  super.onCreate(savedInstanceState);
  setContentView(R.layout.activity_layout);
  ...

  MobclixAdView adview = new MobclixMMABannerXLAdView(this);
  parentView.addView(adview);

}
```

You must either add the MobclixAdView to the layout or make sure you have a reference to the MobclixAdView, otherwise it can be garbage collected before it can complete fetching an ad.

If you at any point remove your MobclixAdView from its parent, you should pause the MobclixAdView before doing so by calling pause().

```
adview.pause();
parentView.removeView(adview);
```

2.4 Integrating a Mobclix FullScreen AdView

The MobclixFullScreenAdView is only available for integration through code. A basic example is provided but it is recommended that you look at the full SDK documentation for full details.

Import the MobclixAdView classes to your Activity , as shown below.

```
import com.mobclix.android.sdk.MobclixFullScreenAdView;
```

Then, all you need to do is create the view, and add it to your view hierarchy, as shown below.

```
@Override
public void onCreate(Bundle savedInstanceState) {
  ...

  MobclixFullScreenAdView adview = new MobclixFullScreenAdView(this);
  adview.requestAndDisplayAd();

}
```

3. Open Allocation

1. Introduction to Open Allocation

1.1 What is Open Allocation?

Open Allocation is a feature provided to Mobclix developers that allows them to "allocate" a certain portion of their advertising impressions to networks that don't participate in the Mobclix exchange (i.e. these impressions are "free" or "open", allowing them to be "allocated" anywhere). The most common use of this feature is to advertise with Mobclix and Google AdMob together.

1.2 How does it work?

At a high level Open Allocation works as follows: The developer sets up Open Allocation by specifying a percentage of the total number of impressions they'd like to be used by open allocation as well as integrating the ad networks that they'd like to use into their application. Based on the specified percentages of Open Allocation, the Mobclix SDK will decide to either request an ad from the Mobclix servers or selects one of the Open Allocations. If the Mobclix SDK selects an Open Allocation network, an event method is called allowing the application to then invoke whichever advertising network SDK it wants.

1.3 Ad Network Adapters

A large majority of the developers who utilize Open Allocation do so in order to integrate Google AdMob or Millennial Media. We at Mobclix recognized this and wanted to make things even easier for our developers so we built in extra support for those networks. We call this feature "Ad Network Adapters". Look for AdMob specific steps within this guide if you plan on utilizing these networks.

1.4 How do I set it up?

This guide outlines all of the steps required to successfully utilize Open Allocation in an application.

There are 3 parts to setting up Open Allocation:

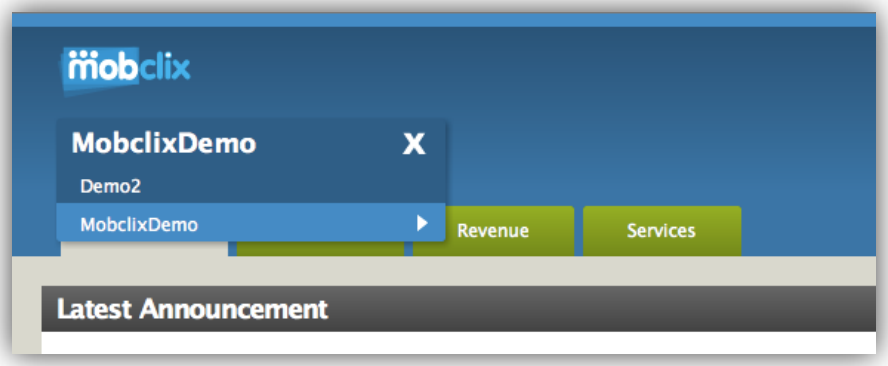
1. Enabling Open Allocation on the Mobclix Developer Dashboard
2. Integrating Ad Network SDKs into your application
3. Implementing the MobclixAdView event methods to handle Open Allocation requests

2. Dashboard Setup

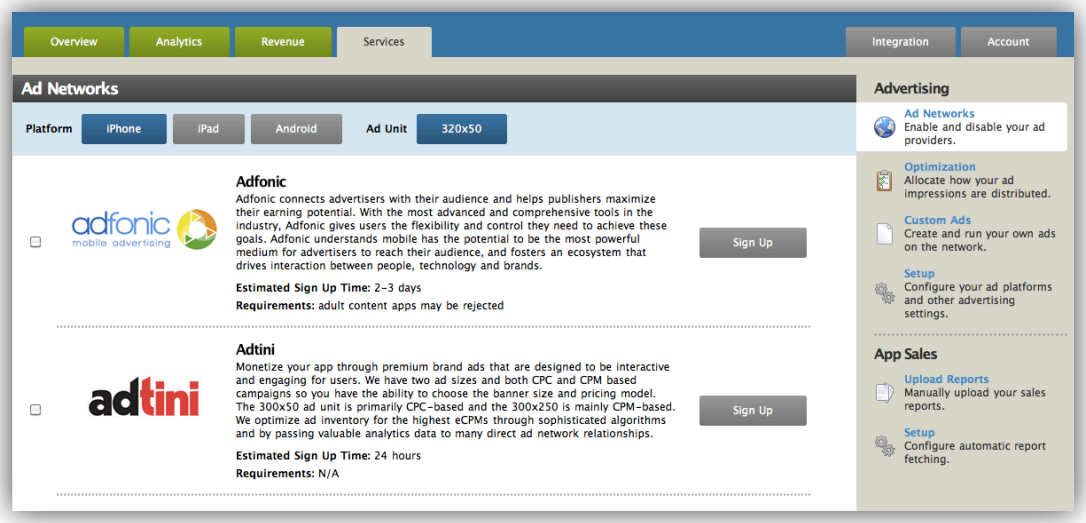
The first step in setting up Open Allocation is letting the Mobclix Ad servers know that your application will be utilizing this feature. This is done through the Mobclix Developer Dashboard.

2.1 Enabling Open Allocation

1. Once you've logged into the Mobclix developer dashboard first make sure that the application that you're enabling Open Allocation for is selected in the application drop down list located in the top left.

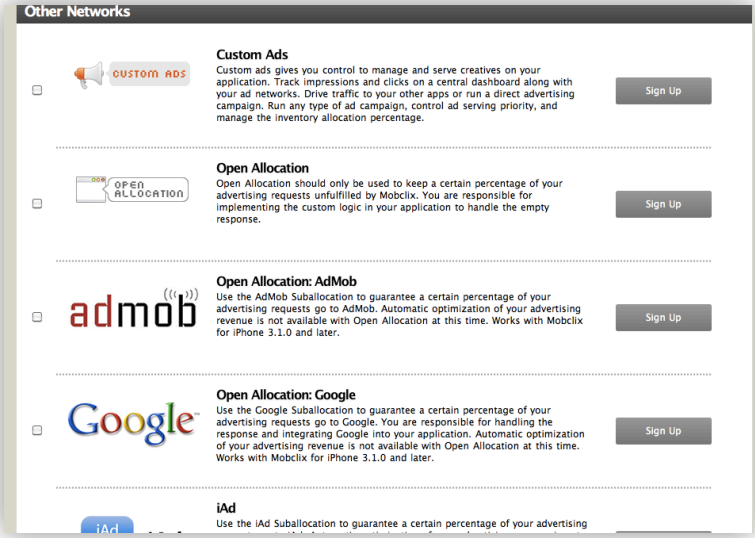


2. Then click the "Services" tab along the top and afterwards the "Ad Networks" button under "Advertising" on the right.



3. Select the correct platform and Ad Unit size along the top for which you'd like to enable Open Allocation.

4. Scroll Down to the bottom of the page where you'll see the heading "Other Networks"



5. The options that you see will depend on the platform and ad size that you're enabling. The different options are explained below:

   a. **Open Allocation: AdMob** - Click "Sign Up" if you plan on utilizing AdMob advertisements in your application

   b. **Open Allocation: Google** - Click "Sign Up" if you plan on utilizing Google AdSense advertisements in your application. Note that Google has merged their AdMob and Google AdSense for Mobile Apps into a single library. Please use Open Allocation: AdMob if you're using the updated Google AdMob SDK.

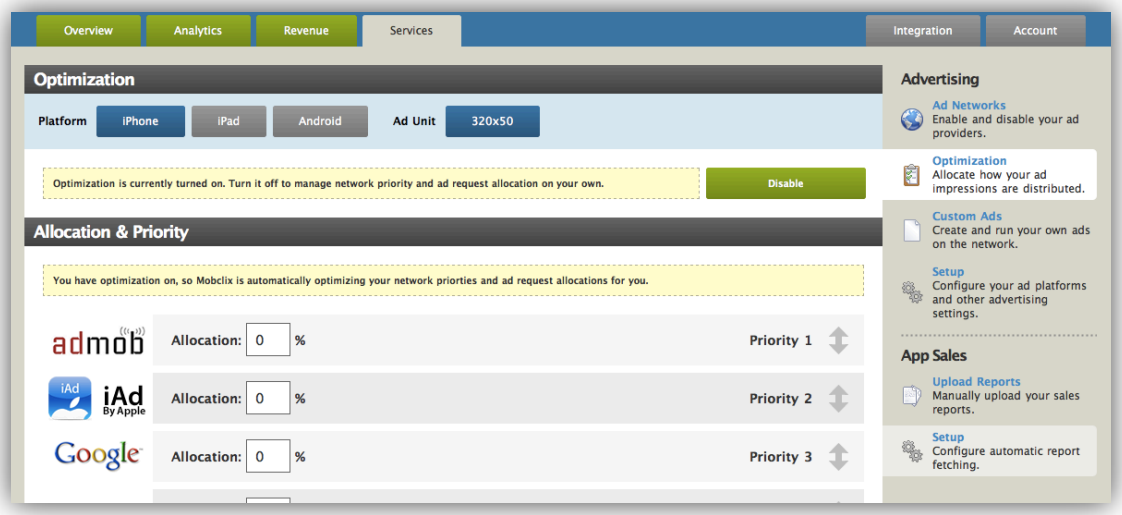
   c. **Open Allocation: Millennial Media** - Click "Sign Up" if you plan on utilizing Millennial Media advertisements in your application.

   d. **Open Allocation** - Click "Sign Up" if you plan on utilizing any other advertising network

2.2 Configuring Allocation

1. Once you've enabled the open allocation network(s) you're planning on using you need to specify what percentage of your impressions should be sent to the network(s).

2. On the Mobclix Developer Dashboard click the "Services" tab along the top then "Optimization" under "Advertising" on the right side.



3. At the top select the Platform and Ad Unit size for which you're configuring Open Allocation

4. Under the "Allocation & Priority" heading you'll see each of the advertising networks (including the recently added open allocation networks) that are enabled for your application's Platform and Ad Unit size.

5. Allocation Percentages

   a. The allocation percentage next to an advertising network (or open allocation network) specifies the percentage of impressions that will be directed to that network. For example if

the allocation for the network adtini is 10% and your application serves a total of 10k impressions per day then 1k of them will be adtini ads.

6. Setting Allocation Percentages

   a. By default Mobclix works with networks integrated on its exchange to optimize your allocations and maximize your advertising revenue. This is made possible since the directly integrated networks share revenue data with Mobclix. However since Mobclix is not able to obtain revenue data from networks integrated via Open Allocation **you must manually set the allocation percentages for these networks to numbers of your choosing**. To do this simply edit the number next to "Allocation" for each of the Open Allocation networks that you enabled previously.

> Note: We recommend keeping optimization on as this allows Mobclix to optimize all of your non-open allocation inventory, making you the most money. However if you decide to turn Optimization off you must manually specify the allocation percentage for EVERY network, not just the open allocation ones. Otherwise if optimization is on then there is no need to specify every networks allocation, the Mobclix networks will be automatically optimized. Optimization can be turned on and off on the same page where you set the Allocation percentages.

3. Code Integration

Now that we've set up the dashboard it's time to integrate open allocation into the application.

This guide assumes that you've already integrated the Mobclix SDK into your application and setup advertising as explained earlier in this document. If this is not the case you should pause here and follow the steps found here, then return to this section after.

3.1 Adding additional SDK's to your Application

Open Allocation involves essentially integrating each of the ad networks into your application. The first step therefore is adding their SDK into your application. Since the integration of each ad network SDK is different and constantly changing you should refer to their specific integration guide for how to properly add their SDK to your project. However in general the way to achieve this is:

Open up your Android project in Eclipse. Click on "**Properties**" in the "**Project**" menu bar. Select "**Java Build Path**" on the left and then select the "**Libraries**" tab. Click "**Add JARs...**" and select the AdMob, Google, or Millennial Media SDK.

Ad Network Adapter - AdMob/Google

AdMob requires you to reference the "**GoogleAdMobAdsSDK.jar**" file that can be downloaded from the AdMob website. Once referenced, you will need to add the "**com.google.ads.AdActivity**" and the "**ACCESS_NETWORK_STATE**" permission to the AndroidManifest.xml. Details can be found in the AdMob/Google documentation found here. In addition, please add the meta-data line to your AndroidManifest.xml:

```
<meta-data android:name="ADMOB_PUBLISHER_ID"
     android:value="xxxxxxxxxxxxxx"/>
```

No additional code should be necessary to have AdMob ads load. However, please read the following section to make sure any **MobclixAdViewListener**s you have enabled are set up properly. In addition, while testing on an Android Emulator, only test AdMob ads will be loaded.

Ad Network Adapter - Millennial Media

Millennial Media requires you to reference the "**MMAdView.jar**" file that can be downloaded from the Millennial Media website. Once referenced, you will need to add the "**com.millennialmedia.android.MMAdViewOverlayActivity**" and "**com.millennialmedia.android.VideoPlayer**" activities to your AndroidManifest.xml. In addition, you must add the "**ACCESS_NETWORK_STATE**" and "**WRITE_EXTERNAL_STORAGE**" permissions to your AndroidManifest.xml. Details can be found in the Millennial Media Android SDK documentation that comes with their SDK.

In addition, please visit the Mobclix Developer Dashboard, where you will be able to set your Millennial Media Application Id, which will be passed to the Mobclix AdViews when an Open Allocation Millennial Media is returned.

No additional code should be necessary to have Millennial Media ads load. However, please read the following section to make sure any **MobclixAdViewListener**s you have enabled are set up properly.

3.2 Implementing MobclixAdView Event Methods

When the Mobclix SDK receives a message from the Mobclix servers that an open allocation network should be used to show an advertisement it looks for specific listeners for instructions on how to proceed. These methods should all be defined within the class **MobclixAdViewListener**, which must then be registered to the **MobclixAdView** by the method **addMobclixAdViewListener**.

3.2.2 onOpenAllocationLoad

This method will be called when the Mobclix SDK receives a message to show an advertisement from an open allocation network. The method indicates to the Mobclix SDK whether it should try to automatically call the appropriate Network Adapter or if the application will manually request the ad directly. If the method returns **true**, the Mobclix SDK will assume that the application has handled the Open Allocation. If **false**, the Mobclix SDK will attempt to use the corresponding Network Adapter to load the Open Allocation ad. An example of a fully implemented **MobclixAdViewListener** would look like this:

```
public void onSuccessfulLoad(MobclixAdView view) {
  view.setVisibility(View.VISIBLE);
  mAdsense.setVisibility(View.GONE); // Hide AdSense ads
}

public void onFailedLoad(MobclixAdView view, int errorCode) { }

public boolean onOpenAllocationLoad(MobclixAdView adView,
                                    int openAllocationCode) {
  if (openAllocationCode == MobclixAdViewListener.SUBALLOCATION_ADMOB ||
      openAllocationCode == MobclixAdViewListener.SUBALLOCATION_GOOGLE) {
    // If the Google Admob jar file is added to the Java Build Path
    // the Mobclix Android Library will handle the ads automatically.
    return false;
  }

  if (openAllocationCode == MobclixAdViewListener.SUBALLOCATION_OTHER) {
    view.setVisibility(View.GONE);
    // Handle other open allocation case here
    return true; // The open allocation has already been handled.
  }
  return false;
}

public void onAdClick(MobclixAdView adView) { return; }
public void onCustomAdTouchThrough(MobclixAdView adView, String string) {
  return;
}
public String keywords() { return null; }
public String query() { return null; }
```

3.2.3 continueRequest

In the case that the onOpenAllocationLoad method returns **true**, the Mobclix SDK will assume that the application has handled the Open Allocation. However, in the case that the application fails to successfully handle the Open Allocation (such as if a 3rd party SDK fails to return an ad), the application may wish to attempt the next Open Allocation ad network that was specified on the Mobclix Developer Dashboard.

For example, given the following priority order:

```
1. Open Allocation: AdMob
2. Open Allocation: Other
3. Mobclix
```

If Open Allocation: AdMob fails to return an ad, the next network will be tried, in this case Open Allocation: Other. Let's say the application responds to the Open Allocation: Other callback by requesting an ad from a third-party network. However, no ad is returned by this third-party network. The application would wish to continue the next request in this list, in this case Mobclix.

Therefore, we have added a call so that the developer can do so: **MobclixAdView.continueRequest()**.

A psuedo-code example of using this method is as follows:

```
public boolean onOpenAllocationLoad(MobclixAdView mobclixAdView,
                                    int openAllocationCode) {
  if (openAllocationCode == MobclixAdViewListener.SUBALLOCATION_OTHER) {
    thirdPartySdk.setListener(new ThirdPartySdkListener() {
      public void onSuccess() {
        thirdPartySdk.setVisibility(View.VISIBLE);
      }
      public void onFail() {
        thirdPartySdk.setVisibility(View.GONE);
        mobclixAdView.continueRequest();
      }
    });
    thirdPartySdk.requestAd();
    return true; // The open allocation has already been handled.
  }
  return false;
}
```

4. ProGuard

The ProGuard tool shrinks, optimizes, and obfuscates your code by removing unused code and renaming classes, fields, and methods with semantically obscure names. ProGuard is integrated into the Android build system, so you do not have to invoke it manually. ProGuard runs only when you build your application in release mode, so you do not have to deal with obfuscated code when you build your application in debug mode. Having ProGuard run is completely optional, but highly recommended.

This section describes how to enable and configure ProGuard while using the Mobclix Android Library.

4.1 Configuring the Android Project

In order to use ProGuard tool with the Mobclix Android Library, you must reference a version of the Android SDK greater than or equal to **API level 8**. This does not mean that your application requires this API level and can still target lower API levels. The minimum requirement for the Mobclix Android Library is still API level 3.

Through reflection, the Mobclix Android Library attempts to use higher levels of Android SDK APIs. If you don't reference a recent version of the Android SDK, this reflection will fail and ProGuard will throw errors.

4.2 Configuring proguard.cfg

Please add the following lines to your proguard.cfg file found in the root of your Android project to ensure that the Mobclix Android Library will continue to work as expected.

```
-keep public class com.mobclix.android.sdk.*

-keep class com.mobclix.android.sdk.MobclixContactsSdk3_4
-keep class com.mobclix.android.sdk.MobclixContactsSdk5
-keep class com.mobclix.android.sdk.MobclixWebViewClientSdk11
-keepclassmembers class com.mobclix.android.sdk.MobclixWebViewClientSdk11
{
  <init>(...);
  public void *(...);
}
-keep class com.mobclix.android.sdk.MobclixWebChromeClientSdk5
-keepclassmembers class com.mobclix.android.sdk.MobclixWebChromeClientSdk5
{
  <init>(...);
  public void *(...);
}
-keep class com.mobclix.android.sdk.MobclixWebChromeClientSdk7
-keepclassmembers class com.mobclix.android.sdk.MobclixWebChromeClientSdk7
{
  <init>(...);
  public void *(...);
}

-keep class com.mobclix.android.sdk.MobclixJavascriptInterface

-keepclassmembers class com.mobclix.android.sdk.MobclixJavascriptInterface
{
  public void *(...);
  <methods>;
}

-keepclassmembernames class
com.mobclix.android.sdk.MobclixJavascriptInterface {
  public void *(...);
  <methods>;
}
```

4.3 Configuring proguard.cfg with Open Allocation Google AdMob

In addition to the above lines, please add the following lines to your proguard.cfg file to ensure that the Mobclix Android Library works with the Google AdMob library. Please note that this is only tested against versions Google AdMob 4.0.2 and above.

```
-keep public class com.google.ads.*

-keepclassmembers class com.google.ads.AdView {
  <init>(...);
  public void *(...);
}

-keepclassmembers class com.google.ads.AdSize {
  public static <fields>;
}

-keepclassmembers class com.google.ads.AdRequest {
  <init>(...);
  public void *(...);
}

-keepclassmembers class com.google.ads.AdListener {
  <init>(...);
  public void *(...);
}
```

4.4 Configuring proguard.cfg with Open Allocation Millennial Media

In addition to the above lines, please add the following lines to your proguard.cfg file to ensure that the Mobclix Android Library works with the Millennial Meida library.

```
-keep public class com.millennialmedia.android.*

-keepclassmembers class com.millennialmedia.android.MMAdView {
  <init>(...);
  public void *(...);
}
-keepclassmembers class com.millennialmedia.android.MMAdView$MMAdListener
{
  <init>(...);
  public void *(...);
}
```

5. Troubleshooting

5.1 Common Problems

5.1.1 The message “An error has occurred” appears in the AdView.

This error occurs when your ad feed hasn’t been set up yet. Please visit the Mobclix Developer Dashboard here to set up your ad feed.

5.1.2 Nothing is shown in the AdView.

There are a few reasons why nothing may be showing up in the AdView. First, check the following:

1. The “**APPLICATION_ID”** has been added to the AndroidManifest.xml and is inside the **<application>** flags.
2. The “**INTERNET”** and “**READ_PHONE_STATE”** permissions have been added to the AndroidManifest.xml and is outside the **<application>** flags.

If these are properly set, it is likely that all of the Ad Networks aren’t active for the application yet. In the **AndroidManifest.xml**, try replacing the **APPLICATION_ID** with “**insert-your-application-key**”. This is a test id that should allow Mobclix banners appear in the AdView. If the banners appear, the AdViews have been implemented correctly and real ads will start appearing once the Ad Networks have processed the application.

6. Additional Support

If you have any additional questions, there are a few more resources available to you:

- The Mobclix Example Application, found here, shows the Mobclix SDK fully integrated and running.
- A complete overview of the SDK and all classes and methods available can be found in the Documentation folder that came in the Mobclix for Android SDK zip file.
- The header files provided with the Mobclix SDK contain documentation above each method.
- If you have any other questions, send us an email at support@mobclix.com and we'll be happy to help!