VELTI
MOBCLIX EXCHANGE

SDK Integration Guide for Android 4.1.0

DN: 05-02-0403-1

mobclix™ | a velti company

velti®

Table of Contents

Audience

This guide is intended for 3rd-party developers who:

- Create Android applications and want their applications to receive ads served by the Mobclix Exchange.
- Integrate their applications with the Mobclix Exchange SDK for Android.

Introduction To Mobclix Exchange SDK For Android

The Mobclix Exchange SDK for Android allows your application to receive ads from the Mobclix Ad Exchange. The Mobclix Exchange SDK helps you monetize your application in the following ways:

- Provides access to multiple ad networks for better performance and fill rate.
- Allows you to insert different ad units from Mobclix Exchange into your application (such as banner ads, full-screen interstitial ads, and rich media ads).
- Lets you receive ads from non-Mobclix Exchange SDKs through Open Allocation.
- Integrates with ProGuard, a program that helps obfuscate and shrink your application code. For details, see "Configuring ProGuard for the Mobclix Exchange SDK" on page 32.
- Collects demographic data from a user's device that can be used to target an audience. For details, see Passing Demographics Data with an Mobclix Exchange Ad Request.

When you downloaded the Mobclix Exchange SDK for Android from the Mobclix Developer Dashboard, the SDK contained these items:

- `mobclix.jar` - An archive file that contains the Mobclix Exchange SDK classes for the Android platform.
- Demo Application - A fully-functional Android application that integrates all features of the Mobclix Exchange SDK.
- Integration Guide - A document has the explanations that help new developers integrate the Mobclix Exchange SDK for Android into their applications.

Using the Mobclix Exchange SDK for Android Guide

This guide is organized into required and optional information for integrating the Mobclix Exchange SDK for Android into your application. The required information explain the minimum integration steps you must perform for your app to start receiving ads served by Mobclix Ad Exchange. The optional information explain the extra features that the Mobclix Exchange SDK for Android provides to help you maximize your revenue earnings from publishing mobile ads.

Required:

You must complete these steps before you can start earning revenue from ads served by the Mobclix Exchange:

- "Registering Your Application with Mobclix Exchange and Obtaining Your Application ID" on page 8
- "Configuring Your Application on the Mobclix Developer Dashboard" on page 10
- "Adding the Mobclix Exchange SDK file to Your Android Project " on page 12
- "Referencing the Mobclix Exchange SDK" on page 12
- "Adding Mobclix Exchange Meta-Data and Permissions to Your Application" on page 14
- "Integrating the Mobclix Exchange Ad into Your Application Using Interface Builder or Java Code" on page 16
- "Using Code to Integrate a Mobclix Full Screen Ad View into Your Application" on page 19
- "Configuring ProGuard for the Mobclix Exchange SDK" on page 32 (*Required only if you are using ProGuard)

Optional:

If you want your application to receive ads from other SDKs, you will need to perform the procedures in following sections:

- "Configuring Mobclix Open Allocation and Optimization for Your Application" on page 21
- "Setting Up Open Allocation Using the Mobclix Developer Dashboard" on page 22
- "Configuring the Open Allocation Networks" on page 23
- "Adding Additional SDK's to your Application" on page 25
- "Implementing MobclixAdView Event Methods" on page 28

If you want your application to pass demographic metadata for the user of a device to the Mobclix Exchange SDK, you will need to follow the instructions in this section:

- "Passing Demographics Data with a Mobclix Exchange Ad Request" on page 31

What's New In The Mobclix Exchange SDK For Android

The Mobclix Exchange SDK for Android 4.0.3 has added support for the most recent Google AdMob and Millennial Media SDKs, which are Google AdMob 6.2.x and Millennial Media 4.6.x. For details about how to integrate these third-party SDKs with the Mobclix Exchange SDK for Android, see "Adding Additional SDK's to your Application" on page 25

Documentation Update

This guide now includes information about the `MobclixDemographics`.class. This class is part of the Mobclix Exchange Feedback API, which passes a user's metadata in a Map object to the Mobclix Exchange SDK for Android.

The calls in this `MobclixDemographics`.class enable your application to submit an ad request to Mobclix Exchange that includes demographic metadata for a user (such as Area Code, City, Education, and Postal Code). Mobclix can report on your application's metadata, which can increase the attractiveness of your application for advertisers who are targeting a demographic segment for their ad campaigns. For details about how to include demographic data in an ad request to the Mobclix Ad Exchange, see "Passing Demographics Data with a Mobclix Exchange Ad Request" on page 31.

Upgrading The Mobclix Exchange SDK for Android

The information is intended for users who have deployed previous versions of the Mobclix Exchange SDK for Android and who want to upgrade to the current version of the SDK.

Upgrading From Mobclix Exchange SDK for Android 4.0.0

Recommended Changes

If you are using "Open Allocation: Other" as one of the settings for Open Allocation, please review the section on the event method named `continueRequest()`. This method lets your application fetch the next available Open Allocation ad network if the previous Open Allocation ad network failed to deliver an ad. For details, see "continueRequest()" on page 29.

Upgrading From Mobclix Exchange SDK for Android 3.0.0

Recommended Changes

Add this line to your `AndroidManifest.xml` (as shown in "Adding Mobclix Exchange Meta-Data and Permissions to Your Application" on page 14):

```
android:hardwareAccelerated=”true”
```

This line ensures that HTML 5 video ads plan on Android tablet devices.

Upgrading From an Older Version of Mobclix Exchange SDK for Android (earlier than 3.0.0)

Recommended Changes

Please remove your older Mobclix code and follow the instructions for upgrading to the Mobclix Exchange SDK for Android 4.0.3. Significant aspects of the SDK have changed since the early versions of Android. If you re-implement from scratch, you will be less likely to run into errors.

Setting Up The Mobclix Exchange SDK For Android

The setup for the integration of the Mobclix Exchange SDK for Android into your application requires

- "Registering Your Application with Mobclix Exchange and Obtaining Your Application ID" on page 8.
- "Configuring Your Application on the Mobclix Developer Dashboard" on page 10
- "Adding the Mobclix Exchange SDK file to Your Android Project " on page 12.
- "Referencing the Mobclix Exchange SDK" on page 12 (so that your Android application recognizes the SDK).
- "Adding Mobclix Exchange Meta-Data and Permissions to Your Application" on page 14 (so your Android application recognizes the different components that need to be integrated from the Mobclix Exchange SDK for Android 4.0.3.)
- "Using Java Code to Integrate the Mobclix Exchange Ad" on page 18.

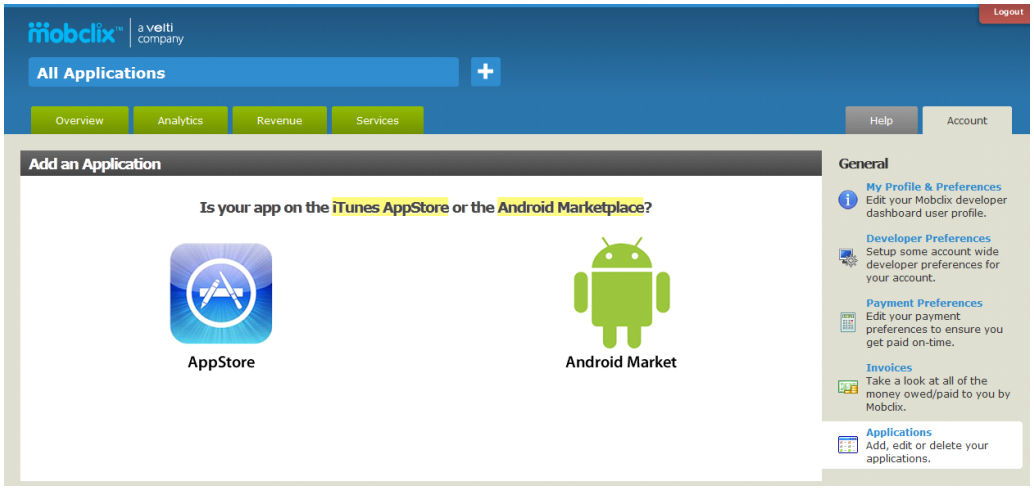
Each of the above sections describe how to perform these setup procedures. When you have completed all of these procedures, your application will be ready to receive ads from the Mobclix Exchange SDK for Android.

Registering Your Application with Mobclix Exchange and Obtaining Your Application ID

Before you can start integrating Mobclix Exchange SDK for Android for your application, you should register your application with the Mobclix Exchange. After you have registered your application, the Mobclix Exchange generates an Application ID. This ID allows your application to be recognized by the Mobclix platform, and consequently allow your application to receive ads from the Mobclix Exchange. The steps are:

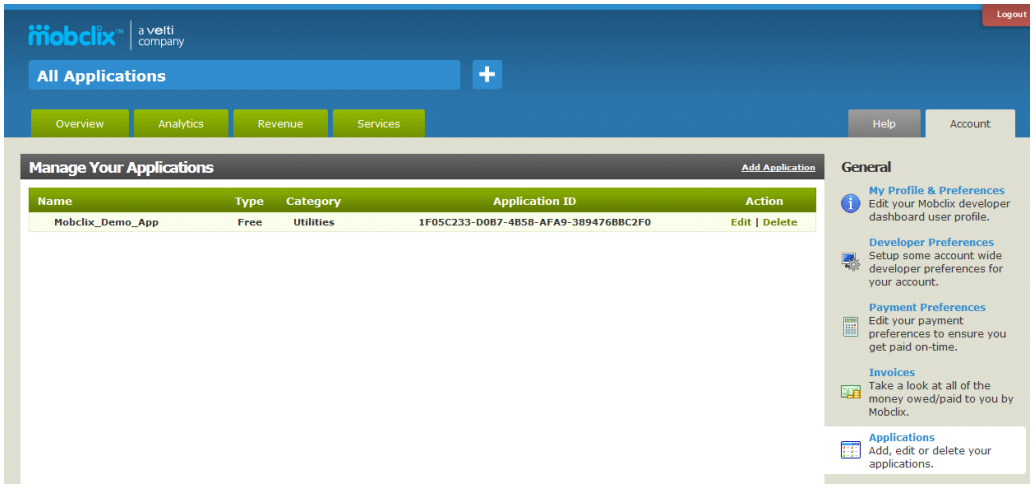
1. Log into the Mobclix Developer Dashboard at `https://developer.mobclix.com`.
2. At the top of the Dashboard, click the plus sign (**+**) next to All Applications (The Add an Application page opens and displays the **Account** tab with **Applications** selected).

Figure 1-1: Adding an Application in the Mobclix Developer Dashboard



3. Click **Android Market** icon. (The Basic Information and Detailed Information sections display.)
4. Fill out the fields for registering your Application. (Required fields are marked with a star)
    - **Application Name*** (Information about your application to help identify it throughout the Mobclix platform.)
    - **Website** (The URL for your App)
    - **Type** (Is it a Free or Paid application?)
    - **Start Date** (When did this application first go live to the public?)
    - **Description*** (This information is critical to helping target the best-paying ads for your application. Your app will not be served ads by most networks without it. For more information about what goes into a description to increase your revenue, see http://support.mobclix.com/entries/20257596-enter-your-app-info-for-better-ads.
    - **Category*** (Use the category that your app exists in the AppStore)
    - **Location Based?** (Whether your App uses location-based services)
    - **Level of "Adult Ads" Allowed** (This information is more detailed and will allow us to serve your app better ads that are more targeted for your audience. Select the best options that describe your app. This will directly affect the ads that your app receives.)
    - **Content Rating** (
    - **Supported Languages** (Select which languages your App supports.)
    - **Apple ID** (Information you provided to Apple in iTunesConnect for AppStore. If available for iOS, your Apple ID is required by some networks to serve ads.
5. Click **Save Settings**. (Page refreshes and displays the title "Manage Your Applications".)

Figure 1-2: Manage Your Applications Screen showing the Application Id



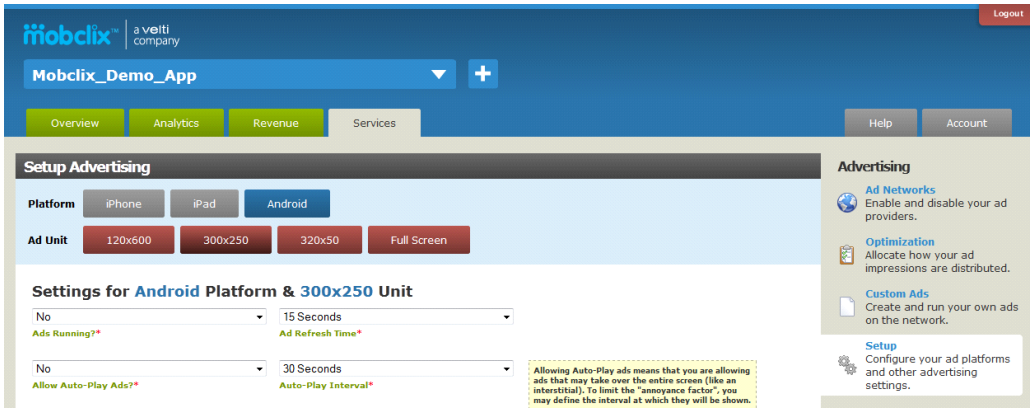
The Application ID for your App appears.

Configuring Your Application on the Mobclix Developer Dashboard

To enable the Mobclix Exchange platform to deliver ads within your application requirements, you must configure your application settings on the Mobclix Developer Dashboard. In the following procedure, you will perform three tasks: 1) specifying how you want the ads to appear in your application (such as the size of your ad, the time interval for refreshing the ad, whether to display rich media ads), 2) whether you are showing ads in test mode or production mode, and 3) signing up for the Ad Networks on the Mobclix Ad Exchange that you want to receive ads from.

1. Log into `https://developer.mobclix.com`.
2. Click **Services** tab.
3. In right column, click **Setup**. (Setup Advertising appears.)

   Figure 1-3: Setup Advertising screen

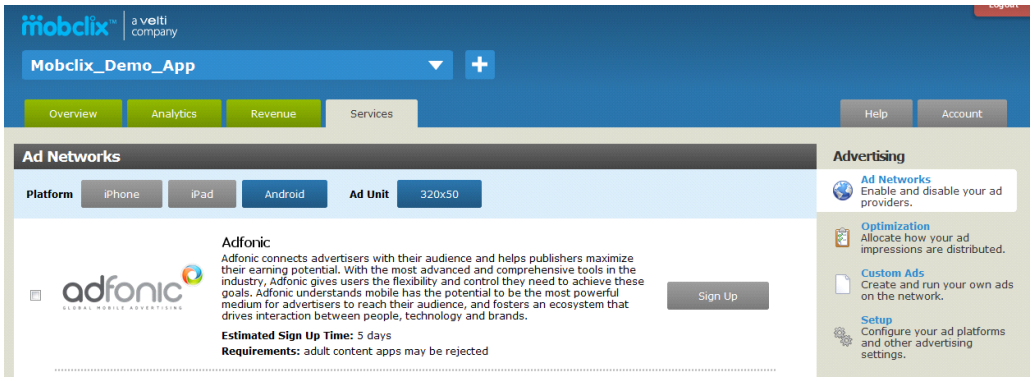


4. For **Platform**, click Android. For **Ad Unit**, click the size of your ad.

5. Fill out the following fields:
   - **Ads Running?*** (Select Yes or No to run ads in your application)
   - **Ad Refresh Time*** (Select the refresh rate that you want for this ad unit.)
   - **Allow Auto-Play Ads?*** (Set Yes to allow ads that may take over the entire screen without user interaction. Set No if you do not want ads to take over the entire screen without user interaction.)
   - **Auto-Play Interval*** (Set the time in seconds or minutes interval between displaying ads that may take over the entire screen)
   - **Ask Permission for Rich Media?*** (Set Yes to ask the user before displaying a rich media ad; No to not ask user before displaying a rich media ad)
   - **Test Mode** (Select "On" if you are testing your application. Select "Off" if you are ready to put your application into production.)

     Note: If you are using Open Allocation with your application, you must set the value of Test Mode to "Off". Open Allocation will not work if the value is set to "On". For more details about Open Allocation, see "Configuring Mobclix Open Allocation and Optimization for Your Application" on page 21.
6. Click **Save**.
7. In right column, click **Ad Networks**. (Ad Networks appear with the selected ad units you chose in the Setup tab.)

   Figure 1-4: Available Ad Networks Page



8. Click **Sign Up** for each ad network you want to receive ads from the Mobclix Ad Exchange.

   The ad networks you see on the screen depends on the specific platform and ad unit size you choose. Not every ad unit is available for every ad network. In other words, you will only see the list of ad networks that support the platform and ad unit that you selected.

   At the bottom of the list, you will see a section named "Other Networks". The "Custom Ads" network is for inserting your own ads (or "house ads"). The rest of the networks are for Open Allocation. For details about Open Allocation, see "Configuring Mobclix Open Allocation and Optimization for Your Application" on page 21.

Note: When you sign up for an ad network, the button will change to "Disable". This indicates that you have chosen the ad network to serve ads to your App. Later, if you do not want to receive ads from that ad network, just click **Disable** to deactivate it.

You have now completed configuring your application on the Mobclix Developer Dashboard. Your application will be "onboarded" with the selected ad networks, which can take a few hours or a few days depending upon the ad network.

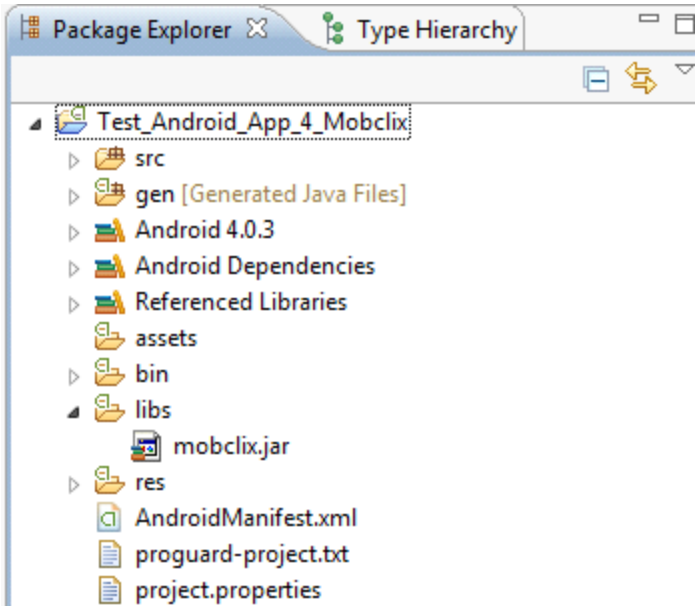
The next setup task involves "Adding the Mobclix Exchange SDK file to Your Android Project " on page 12

Adding the Mobclix Exchange SDK file to Your Android Project

This step copies the Mobclix classes to the project for your application. To perform this step, drag the `mobclix.jar` file from your install directory into your project's **libs** directory.

Note: If this directory does not exist, create it in your project's root directory.

Figure 1-5: Libs directory in Android project that contains the mobclix.jar file

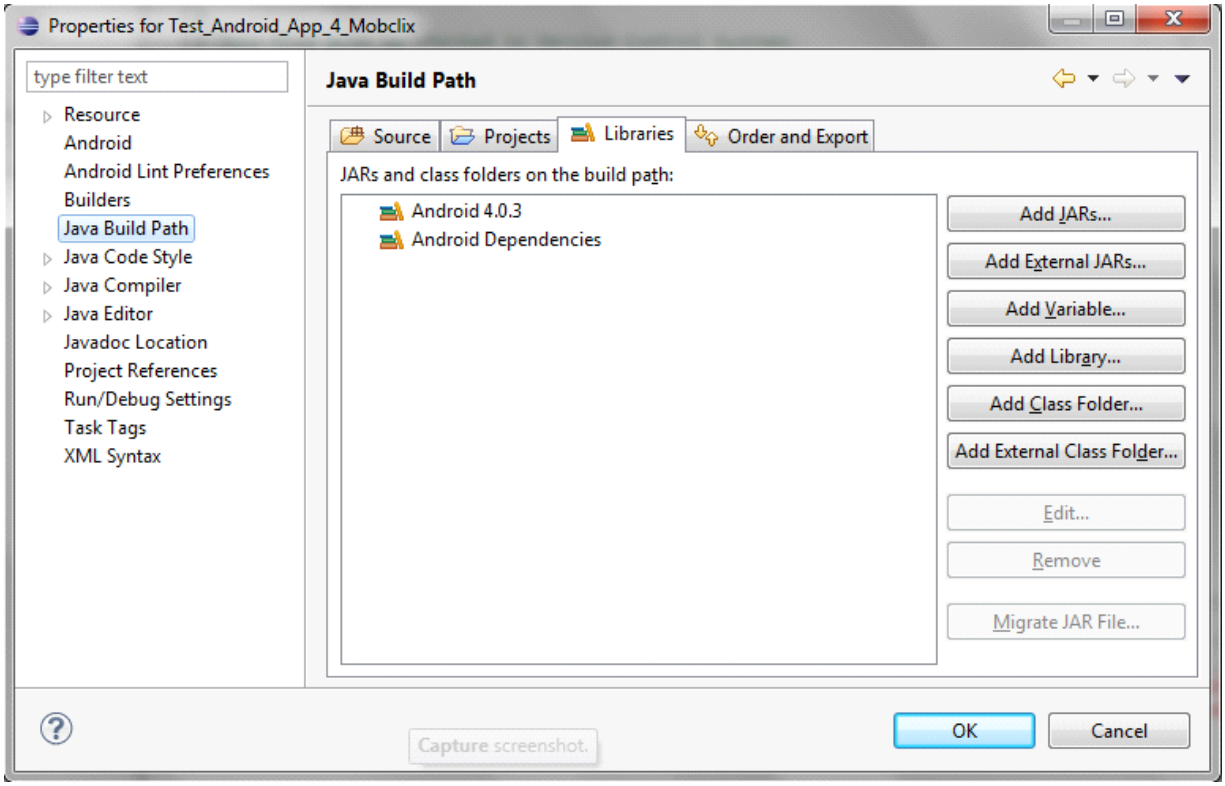


Referencing the Mobclix Exchange SDK

This step associates the Mobclix Exchange SDK with your Android application. This association enables your Android application to recognize the SDK so that the two entities can communicate with each other.

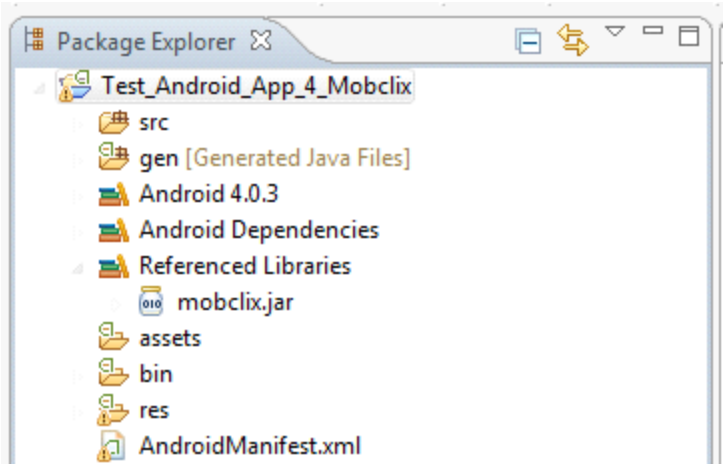
1. In Eclipse, open your Android project.
2. In the menu bar, click **Project** > **Properties**.

Figure 1-6: Properties Dialog



3. Select **Java Build Path** on the left and then select the **Libraries** tab.
4. Click **Add JARs...** and select the **mobclix.jar**.
5. Click **OK**.

Figure 1-7: mobclix.jar file located in the project's Referenced Libraries directory.



Adding Mobclix Exchange Meta-Data and Permissions to Your Application

In this step, you are declaring the components and permissions that your Android app uses.

1. In your application, open up the `AndroidManifest.xml` file.
2. Inside the `<application>` tags, add the following:

   `APPLICATION_ID`

   This is the Mobclix Application ID, found in the Mobclix Dashboard. If you have not yet registered your application with the Mobclix Exchange, you should perform the instructions in "Registering Your Application with Mobclix Exchange and Obtaining Your Application ID" on page 8. You will need the Application ID to finish integrating your application with Mobclix Exchange SDK.

   `MobclixBrowserActivity`

   An activity used by the Mobclix Exchange SDK to display rich media ads and full-screen ads.

3. Outside of the `<application>` tags, add or make sure your application has the following permissions:

   `INTERNET`

   Used by the Mobclix Exchange SDK to fetch advertisements and relay analytics information.

   `READ_PHONE_STATE`

   Used to access a unique identifier that some ad networks require to track impressions. For details, see http://developer.android.com/reference/android/telephony/TelephonyManager.html#getDeviceId().

   Some ad networks choose not to use the `ANDROID_ID` for unique identification because of a bug with DROID2 devices, which is documented at this link: http://code.google.com/p/android/issues/detail?id=10603

   `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+), Mobclix also recommends adding this line of code to the `<application>` tag:

   ```
   android:hardwareAccelerated="true"
   ```

   If this line of code causes issues with your application, then 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.

   Note: 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>
```

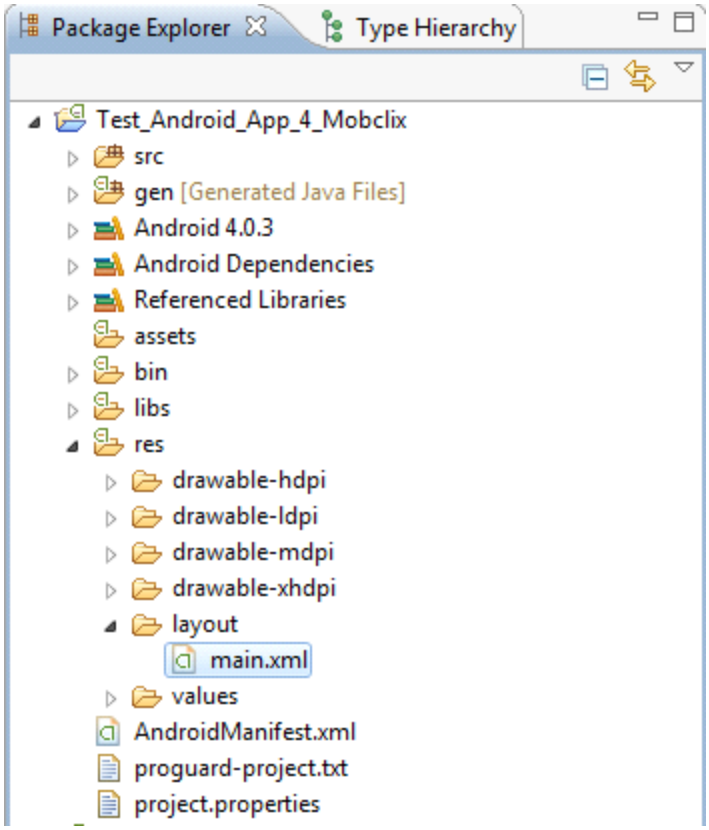
Integrating the Mobclix Exchange Ad into Your Application Using Interface Builder or Java Code

To place a Mobclix Exchange ad into your Android application, you can either use the Interface Builder plugin from Eclipse for the Android Development Tools (ADT) or use Java code. The following procedures describe how to use Interface Builder or Java code to integrate the Mobclix Exchange ads into your application.

Using Interface Builder Plugin from Eclipse for ADT to place the Mobclix Exchange Ad

1. Open up Eclipse and navigate to directory where the XML file for your page layout exists.

   Figure 1-8: Directory to XML file for Page Layout

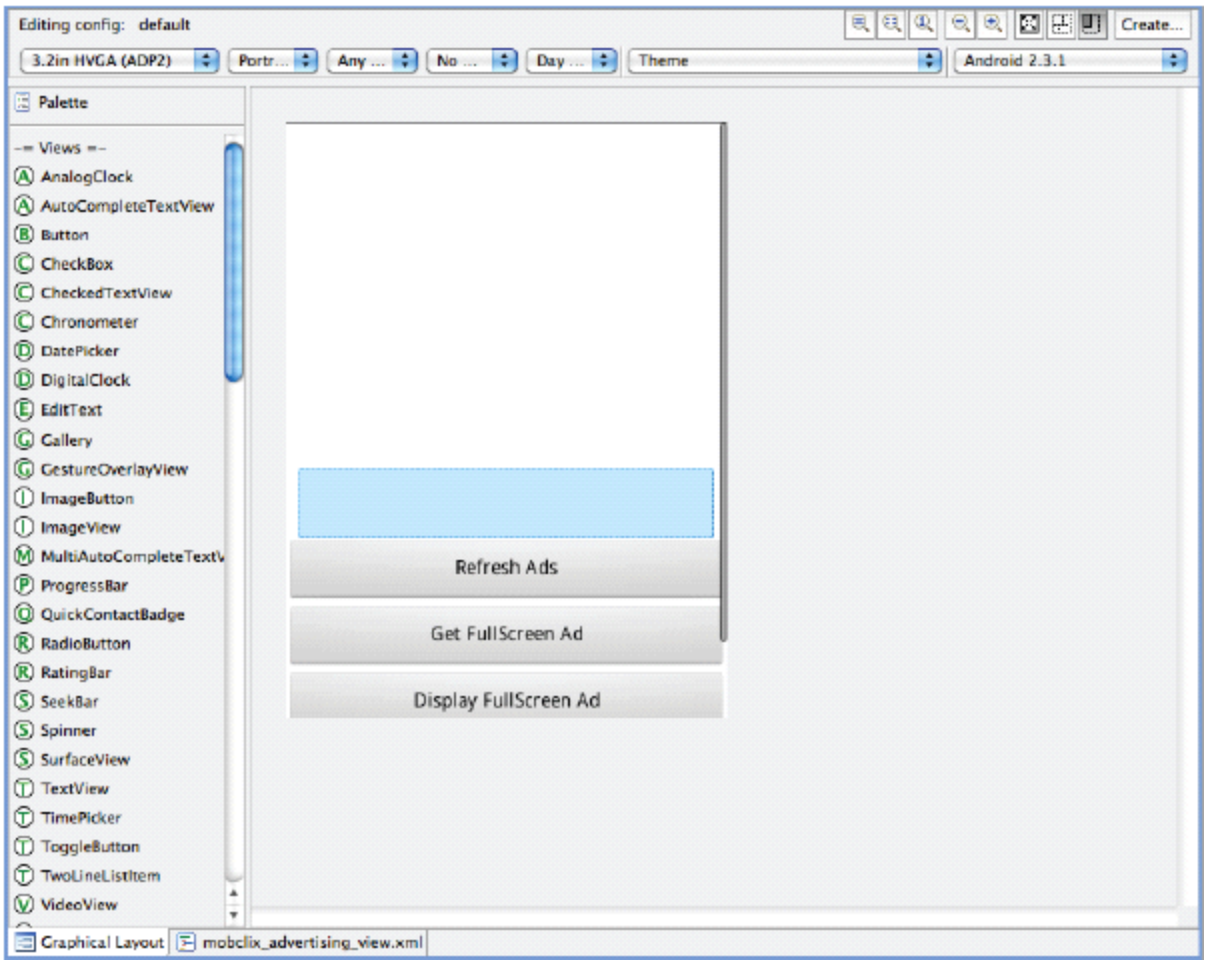
   

2. In the page layout file, add the following code to instantiate the `MobclixMMABannerXLAdView` object for displaying Mobclix ads.

```
<com.mobclix.android.sdk.MobclixMMABannerXLAdView

   android:id="@+id/banner_adview"

   android:layout_width="320dip"

   android:layout_height="50dip"

   android:layout_gravity="center" />
```

Adding this code to the xml will create a rectangle in your layout editor representing the `MobclixAdView` object for you to position, which is shown in the following screenshot.

Figure 1-9: Eclipse Page Layout Editor



3. Finish laying out the `MobclixAdView` in your page's layout.
4. Save and close your page layout's XML file. Your application is now all set to receive ads.

Note: The `MobclixAdView` that you've added will automatically refresh based on the rate that you set on the Mobclix Developer Dashboard when you registered your application. If you need to change or confirm the refresh rate, see "Configuring Your Application on the Mobclix Developer Dashboard" on page 10 for the procedure to set up or change the refresh rate. If you have not set up the refresh rate, you **must** do it now or the `MobclixAdView` will not behave properly.

Using Java Code to Integrate the Mobclix Exchange Ad

If you need to provide a more advanced implementation of an ad, you will need to create views and place them in your code. Some examples of an advanced implementation would be displaying an ad in a vertical scrolling list (using the Android `ListView` class) or creating an ad but not adding the view for it until a specific event occurs in your application.

In this guide, you will see a basic implementation of an ad from Mobclix Exchange in your Activity. These are the steps:

1. Import the `MobclixAdView` classes for each Activity in your application. The following Java code shows the import statements to add to your activity:

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

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

Note: For banner ads, Mobclix provides two classes: `MobclixMMABannerXLAdView`, which is a 320 x 50 banner ad, and `MobclixIABRectangleMAdView`, which is a 300 x 250 banner ad.

2. Then, all you need to do is create the view for the Mobclix ad, and add it to your hierarchy of views. The code that is specific to Mobclix is shown in green.

```
@Override

public void onCreate(Bundle savedInstanceState) {

   super.onCreate(savedInstanceState);

   setContentView(R.layout.activity_layout);

   ...

   MobclixAdView adview = new MobclixMMABannerXLAdView(this);

   parentView.addView(adview);

}
```

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

3. If you at any point remove your `MobclixAdView` from its parent, you should pause the `MobclixAdView` before doing so by calling `pause()`.The following example shows the code you must implement:

```
adview.pause();

parentView.removeView(adview);
```

If you do not pause the ad, your application could respond in an unintended manner. To avoid this behavior, Mobclix recommends that you pause the ad before your remove it.

Using Code to Integrate a Mobclix Full Screen Ad View into Your Application

To integrate your app so it can receive full-screen ads from the Mobclix Exchange, you must modify your code. The `MobclixFullScreenAdView` object is only available through code. Some of the available methods for controlling the preloading and timing of the display of ads include:

- `adview.requestAd();`

  Requests an advertisement which will download and render the ad but not display it. Any additional calls will cancel any pending requests to the Mobclix Exchange ad server.

- `adview.displayRequestedAd();`

  Displays an advertisement loaded by the `MobclixFullScreenAdView` object if the ad exists. Mobclix recommends you use the `hasAd` method to verify that an ad has loaded prior to calling this method.

- `adview.hasAd();`

  Returns `true` if the `MobclixFullScreenAdView` has an ad loaded via the `requestAd` method.

For full documentation about these methods, see the SDK Documentation that was included in your SDK download. To access this documentation, navigate to the **Documentation** folder (which is in the directory that you downloaded the Mobclix Exchange SDK for Android) and then click **index.html** file. Inside this file, click the link **Full SDK API Documentation**.

This is a basic code example for integrating a full screen ad from the Mobclix Exchange in your application:

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

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

2. Create the view, and add it to your view hierarchy, as shown by the code in green below.

```
@Override

public void onCreate(Bundle savedInstanceState) {

...

   MobclixFullScreenAdView adview = new MobclixFullScreenAdView(this);

   adview.requestAndDisplayAd();

}
```

Configuring Mobclix Open Allocation and Optimization for Your Application

Open Allocation is an optional feature that Mobclix provides to allow a Publisher to "allocate" or run a certain portion of their advertising impressions in their app to ad networks that do not participate in the Mobclix Ad Exchange. In other words, these impressions are "free" or "open", which allows them to be "allocated" to other ad networks besides Mobclix.

The Mobclix Exchange SDK for Android enables you to utilize other Ad network SDKs in your application using a single management framework. Through the Mobclix Developer Dashboard, you can easily allocate a percentage of the ad requests from the Mobclix Exchange SDK to Mobclix, iAd , Admob, Millennial or even your own code. The Dashboard also enables you to remotely change the Open Allocation behavior of the Mobclix Exchange SDK after your application goes live.

The most common use of Open Allocation is to receive ads from the Mobclix Exchange and Google AdMob together.

About Open Allocation

At a high level, Open Allocation works as follows:

- Using the Mobclix Developer Dashboard, you specify a percentage of the total number of impressions that you would like to use for each ad network through Open Allocation.
- Based on the allocation percentages you specified for Open Allocation, the Mobclix Exchange SDK for Android will decide to either request an ad from the Mobclix Exchange servers or select one of the networks you specified for Open Allocation. During the percentage of time that you are running Open Allocation, the requests for ad served by the Mobclix Exchange will trigger an event method (`onOpenAllocationLoad`) in your application that you can customize as to what the application should do when the event method is called. You can choose to show no ads or show ads from another ad network. [In either case, your application **must** invoke the non-Mobclix Exchange SDK along with the Mobclix Exchange SDK for Open Allocation to work with the specific ad network.]
- The Mobclix Exchange SDK for Android also includes code called Ad Network Adapters. The Adapters make it easier to integrate common third-party SDKs with Open Allocation. After you configure these Ad Network Adapters, you will not need to add any additional code to invoke the third-party SDK. For details about configuring the Ad Network adapters, see "Adding Additional SDK's to your Application" on page 25.

About Yield Optimization and Open Allocation

The Mobclix Exchange Yield Optimizer automatically selects the highest-paying ad network for each specific ad call. The Optimizer makes its selection based on payment rate information that ad network provides. The Optimizer also considers targeting parameters (such as geo-location) to avoid sending impressions to networks that do not support specific locations.

You can choose to enable the Optimizer or disable it. If you enable the Optimizer, the Mobclix Exchange automatically chooses the highest-paying ad network for traffic allocation and priority. If you disable the Optimizer, you manually choose the priority and the percentage allocation for each Open Allocation network. In general, Mobclix recommends

against disabling the Optimizer because you can miss out on revenue opportunities as a result of not manually adjusting the priority for an ad network that starts or stops paying high rates.

Note: The Yield Optimizer never changes the percentages for Open Allocation that you set. However, the Yield Optimizer does adjust the priority based on the rate that each ad network pays. Because the Mobclix Exchange does not track revenue statistics, the priority of the Open Allocation networks that do not report revenue falls to the bottom of the stack.

Setting Up Open Allocation

There are 3 parts to setting up Open Allocation for your app:

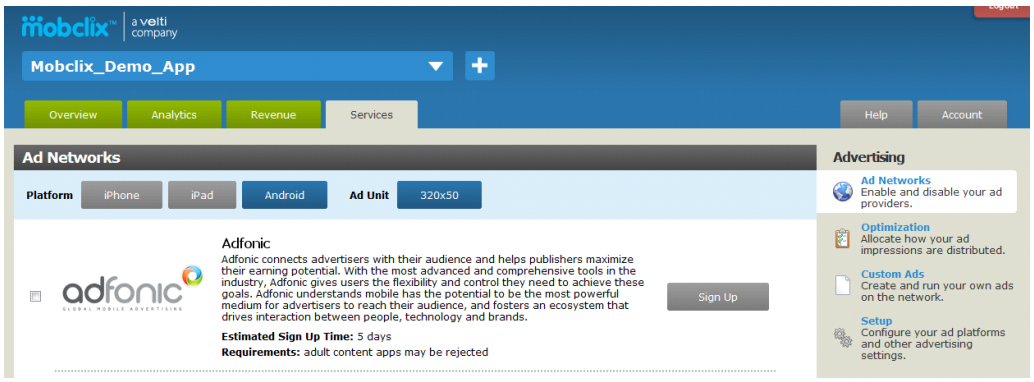
1. Enabling and configuring Open Allocation for your application using the Mobclix Developer Dashboard. See "Setting Up Open Allocation Using the Mobclix Developer Dashboard" on page 22
2. Integrating SDKs from other ad networks into your application. See "Adding Additional SDK's to your Application" on page 1.
3. Implementing the MobclixAdView event methods to handle Open Allocation requests (such as if an Open Allocation request fails to serve an ad). See "Implementing MobclixAdView Event Methods" on page 28.

Setting Up Open Allocation Using the Mobclix Developer Dashboard

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

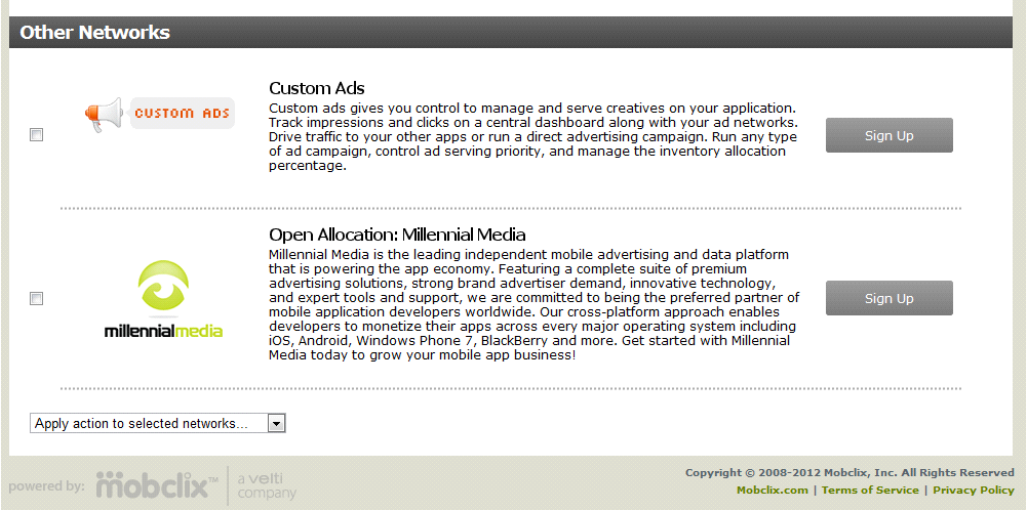
1. Log into the Mobclix Developer Dashboard at `https://developer.mobclix.com`.
2. At the top of the screen, select the application you want to enable for Open Allocation from the application drop-down list.
3. Click the **Services** tab along the top.
4. In the right column under Advertising, click **Ad Networks**.

   Figure 1-10: Ad Networks selection screen



5. Select the correct **Platform** and **Ad Unit** size along the top for which you would like to enable Open Allocation.
6. Scroll down to the bottom of the page to locate the heading **Other Networks**.

Figure 1-11: Open Allocation networks selection page



7. Depending on the platform and ad size that you're enabling, the Dashboard will display the available Open Allocation options. If you choose to use an available Open Allocation network, click **Signup** for each network you choose. Listed below are a few options for Open Allocation networks:

   - **Open Allocation: AdMob** - for running advertisements from AdMob in your application.
   - **Open Allocation: Google** - for running advertisements from Google AdSense in your application.

     Note: Google has merged their AdMob and Google AdSense for Mobile Apps into a single library. If you're using the updated Google AdMob SDK, please select **Open Allocation: AdMob** .
   - **Open Allocation: Millennial Media** - for running advertisements from Millennial Media in your application.
   - **Open Allocation** - for running a certain percentage of your ads to be fulfilled by ad networks not integrated with Mobclix Ad Exchange.

Configuring the Open Allocation Networks

Once you have enabled the Open Allocation network(s) you are planning to use, you need to specify what percentage of your impressions should be sent to the network(s). The total percentage of impressions must equal 100 percent. If the total percentage exceeds 100 percent, you will not be allowed to save your Open Allocation preferences.

To illustrate how Open Allocation works, suppose you selected InMobi as one of your ad networks. If you set the allocation for InMobi at 10 percent and your application serves a total of 10,000 impressions per day, then 1,000 of those impressions served will be ads from InMobi.

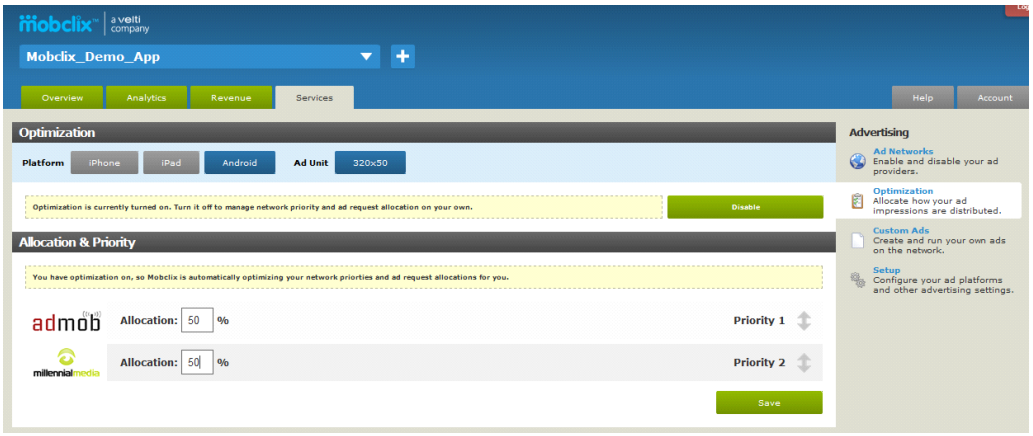
To illustrate how Open Allocation works, suppose you selected the InMobi and adMob ad networks. Assume that your application serves a total of 10,000 impressions per day. If you set the allocation for InMobi at 10 percent and adMob at 20 percent, then 1,000 of those impressions served will be ads from InMobi and 2,000 of those impressions served will be ads from adMob.

Note: By default, Mobclix automatically optimizes the ad networks on the Mobclix Ad Exchange because these networks share revenue data with Mobclix. However, for ad networks that are integrated via Open Allocation, Mobclix does not have access to the revenue data to optimize the allocations. Thus, you must manually set (or optimize) the allocation percentages for these Open Allocation networks.

The following procedure describes how to manually set the percentages for Open Allocation:

1. On the Mobclix Developer Dashboard, click the **Services** tab.
2. Under "Advertising" in the right column, click **Optimization**.

   Figure 1-12: Allocation of Impressions and Priority of Ad Networks



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

   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.

4. To do this simply edit the number next to **Allocation:** for each of the networks that you enabled previously for Open Allocation.

5. Set the Open Allocation priority for each network by clicking **Priority** and dragging it up or down to the order you want that network to be called first when Open Allocation is active.

   Note: Mobclix recommends keeping optimization "Enabled". This setting allows the Mobclix Exchange to optimize all of your non-open allocation inventory, which maximizes the revenue you can earn. However, if you disable Optimization, you **must** manually specify the allocation percentage for **every** ad network, not just the ad networks that use Open Allocation. Otherwise, if you enable optimization, then there is no need to specify each network's allocation, and the ad networks on the Mobclix Ad Exchange will be automatically optimized. Optimization can be enabled and disabled on the same page where you set the Allocation percentages.

Now that you have configured Open Allocation for your app (through the Mobclix Developer Dashboard), the next step is to integrate the 3rd-party SDKs into your application. For details, see "Adding Additional SDK's to your Application" on page 25.

Note: If you have not already integrated the Mobclix Exchange SDK for Android, then you should follow the steps in "Setting Up The Mobclix Exchange SDK For Android 4.0.3" on page 8, before adding additional SDKs for your application.

Adding Additional SDK's to your Application

Open Allocation involves essentially integrating each ad network into your application, which means adding the non-Mobclix Exchange SDKs 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 into your project. The following procedure describes the typical way to add a 3rd-party SDK to your application from the Eclipse IDE:

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

For details of setting up these adapters, see See "Configuring Mobclix Open Allocation and Optimization for Your Application" on page 21 or See "Configuring the Ad Network Adapter for Millennial Media" on page 25.

Configuring the Ad Network Adapter for AdMob/Google

To make it easier for you to integrate with Google or AdMob, Mobclix provides a feature called "Ad Network Adapters". The adapter allows you to integrate ads served from Google AdMob without modifying the Java code in your app. To configure the adapter for Google AdMob and integrate it with Open Allocation, you must perform this procedure:

1. Reference the `GoogleAdMobAdsSDK.jar` file. AdMob requires this step.You can download the `GoogleAdMobAdsSDK.jar` file from https://developers.google.com/mobile-ads-sdk/download.
2. 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 at https://developers.google.com/mobile-ads-sdk/docs/android/fundamentals.
3. Add this meta-data line to your `AndroidManifest.xml`:

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

After completing these three steps, the AdMob ads should load.

No additional code should be necessary to have Google AdMob ads load. However, see "Implementing MobclixAdView Event Methods" on page 28 to make sure any MobclixAdViewListeners are set up properly. Otherwise, your application may not respond properly to any Open Allocation requests. In addition, while testing on an Android Emulator, AdMob will only display "test" ads in the emulator.

Configuring the Ad Network Adapter for Millennial Media

To make it easier for you to integrate with Millennial Media, Mobclix provides a feature called "Ad Network Adapter". The adapter allows you to integrate ads served from Millennial Media without modifying the Java code in your app. To configure the adapter for Millennial Media and integrate it with Open Allocation, you must perform this procedure:

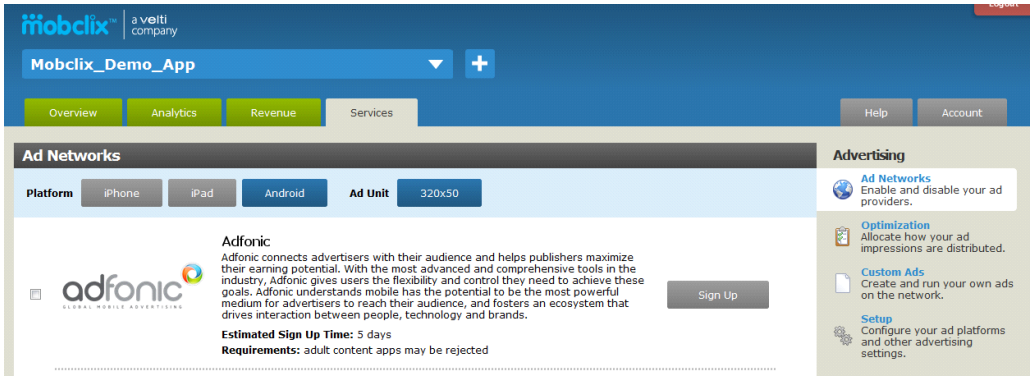
1. Reference the MMAdView.jar file. Millennial Media requires you to reference this .jar file.You can download the MMAdView.jar file from http://mmedia.com/platform-sdk/android.
2. Once referenced, you will need to add the "com.millennialmedia.android.MMAdViewOverlayActivity", com.millennialmedia.android.VideoPlayer" activities, and the "ACCESS_NETWORK_STATE" and `WRITE_EXTERNAL_STORAGE`permissions to the `AndroidManifest.xml`. Details can be found in the Millennial Media documentation at http://mmedia.com/platform-sdk/android.
3. Obtain the unique identifier from Millenial Media. This identifier will be passed to the Mobclix AdViews when an ad from Millennial Media (using Open Allocation) is returned. This is the procedure for obtaining the identifier from Millenial Media:

   a. Log into `https://developer.mobclix.com`.

   b. Click the **Services** tab along the top.

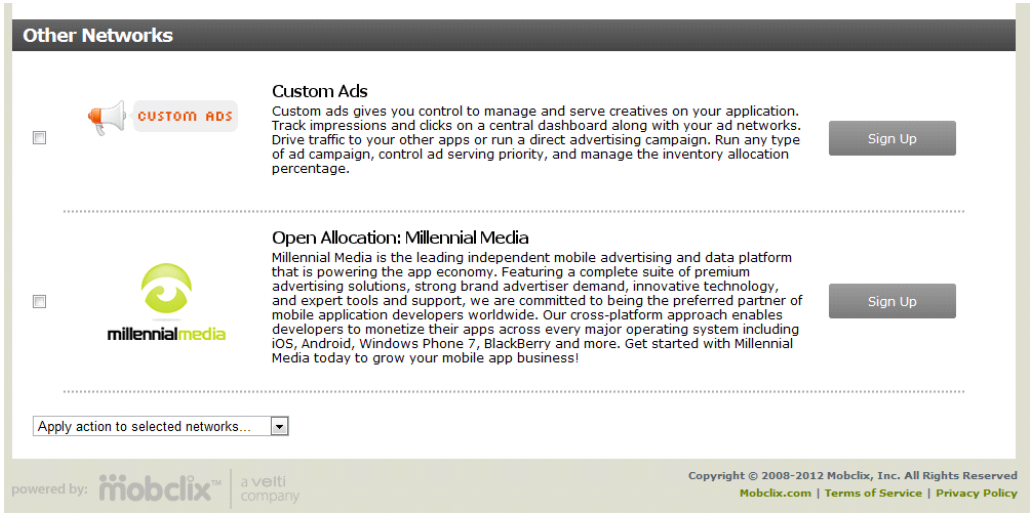
   c. In the right column under Advertising, click **Ad Networks**.

   Figure 1-13: Available Ad Networks page



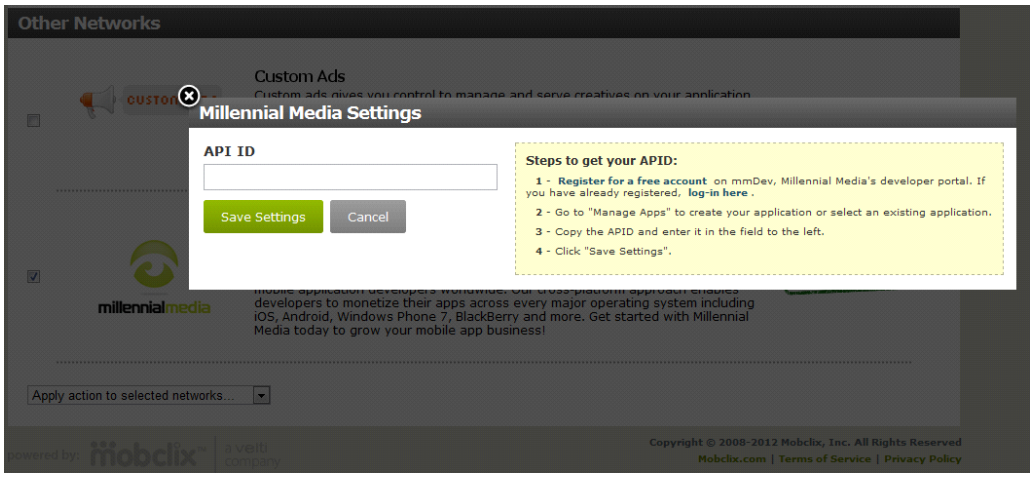
   d. Scroll down to the bottom of the page to locate the heading **Other Networks**.

   Figure 1-14: Other Networks and Open Allocation networks



e. Next to Open Allocation:Millenial Media, click **Signup.** (A dialog pops up).

Figure 1-15: Dialog box for obtaining APID for Millenial Media



f. Follow the instructions to get the unique identifier (APID) for Millennial Media.

g. Click **Save Settings**.

4. Paste the APID for Millennial Media into the `com.mobclix.APPLICATION_ID` key in your application's `AndroidManifest.xml` file.

Your application is now ready to receive ads from Millennial Media using Open Allocation. The APID will be passed to the Mobclix AdViews when an ad from Millennial Media (using Open Allocation) is returned.

No additional code should be necessary to have Millennial Media ads load. However, see "Implementing MobclixAdView Event Methods" on page 28 to make sure any MobclixAdViewListeners are set up properly. In addition, while testing on an Android Emulator, only test AdMob ads will be loaded.

Implementing MobclixAdView Event Methods

When the Mobclix Exchange SDK for Android receives a message from the Mobclix servers that an ad network from your Open Allocation selections should serve an advertisement, the SDK looks for specific event listeners that give 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`.

onOpenAllocationLoad()

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

Note: The code in green is specific for an Open Allocation request.

```
public void onSuccessfulLoad(MobclixAdView view) {

   view.setVisibility(View.VISIBLE);

}

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;

}
```

continueRequest()

If the `onOpenAllocationLoad` method returns `true`, the Mobclix Exchange SDK for Android will assume that your application has handled the request for an ad from Open Allocation. However, if the application fails to successfully handle the request for an ad through Open Allocation (such as a 3rd-party SDK failing to return an ad), you can ask your application to try the next ad network that was specified during the setup of Open Allocation on the Mobclix Developer Dashboard.

For example, suppose you had the following priority order for the ad networks you set for Open Allocation:

- Open Allocation: AdMob (Priority 1)
- Open Allocation: Other (Priority 2)
- Mobclix (Priority 3)

If **Open Allocation: AdMob** fails to return an ad, then the Mobclix Exchange SDK for Android would try the next network **Open Allocation: Other**. Your application responds to the **Open Allocation: Other** callback by requesting

an ad from a third-party ad network. However, the third-party network does not return an ad. In this case, you want your application to continue the next request in this list, which is Mobclix.

To enable this behavior, the Mobclix Exchange SDK for Android provides a method called `MobclixAdView.continueRequest()`.

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

Note: The code in green is specific for an Open Allocation request.

```
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();

           // The open allocation has already been handled.

           return true;

   }

   return false;

}
```

Passing Demographics Data with a Mobclix Exchange Ad Request

To help you achieve higher eCPM rates for your application, the Mobclix Exchange SDK for Android allows your application to submit an ad request to Mobclix that includes demographic metadata for a user. Advertisers can use this metadata to better target an application for their ad campaigns.

To pass the demographic data from the user's device to an ad request from Mobclix Exchange, you use the `MobclixDemographics` class, which is part of the Mobclix Exchange Feedback API. You will need to perform three things to implement this feature:

1. Use the following call from `MobclixDemographics` class to pass user metadata to the Mobclix Exchange SDK for Android:

   ```
   public void MobclixDemographics.sendDemographics(Activity activity, Map<String, Object>
   demographics)
   ```

   where `activity` is the name of the Activity for saving the demographics data from the user device and `Map` is an object where the saved demographic data is stored.

2. Generate a Map object to store the user metadata (similar to the following line of sample code):

   ```
   Map<String, Object> demographics = new HashMap<String, Object>();
   ```

3. Load the Map for each demographic enumeration (similar to the following line of sample code):

   ```
   demographics.put (MobclixDemographics.AreaCode, 650);
   ```

   Note: You can find the complete listing of keys and values for all the enumerations of the `MobclixDemographics` class by navigating to the directory where you unzipped the SDK and looking for this path:

   \Documentation\html\sdk.html\Feedback Classes\MobclixDemographics

To see a complete implementation of the `MobclixDemographics` class, navigate to directory where you unzipped the SDK and look for this path:

\MobclixDemo\src\com\mobclix\android\demo\MobclixDemographicsActivity.java

Configuring ProGuard for the Mobclix Exchange SDK

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. Running ProGuard is completely optional. Mobclix strongly recommends that you run it for better performance and security for your application.

If you choose to use ProGuard with your application, then you must configure it for the Mobclix Exchange SDK for Android. This section describes how to configure ProGuard for the Mobclix Exchange SDK for Android.

Configuring the Android Project

In order to use ProGuard tool with the Mobclix Exchange SDK for Android, you must reference a version of the Android SDK equal to or greater than API level 11. However, you are not required to target your application at this same level; you can still target API levels at less than 11. The minimum requirement for the Mobclix Exchange SDK for Android is target API level 3. But if you build against a version of the Android SDK with a target API level less than 11, your code will not build and you will run into compilation errors.

Through Java reflection, the Mobclix Exchange SDK for Android attempts to use higher levels of Android SDK APIs. If you do not reference a recent version of the Android SDK (3.0 and above), the Java reflection will fail and ProGuard will throw errors.

Configuring the proguard.cfg File

Please add the following lines to your `proguard.cfg` file found in the root of your Android project to ensure that the Mobclix Exchange SDK for Android 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>;

}
```

Configuring proguard.cfg with Open Allocation Google AdMob

In addition to the lines contained in the code example for "Configuring the proguard.cfg File" on page 32, please add the following lines to your proguard.cfg file to ensure that the Mobclix Exchange SDK for Android 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 *(...);
}
```

Configuring proguard.cfg with Millennial Media as an Open Allocation Ad Network

In addition to the lines contained in "Configuring the proguard.cfg File" on page 32, you must also add the following lines to your `proguard.cfg` file to ensure that the Mobclix Exchange SDK for Android works with the Millennial Media 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 *(...);

}
```

Troubleshooting

Q. The message "An error has occurred" appears in the AdView.

A. 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.

Q. Nothing is shown in the adView.

A. First, check that the following items have been added correctly to your `AndroidManifest.xml` file:

1. The meta-data for the `com.mobclix.APPLICATION_ID` parameter is **inside** the `<application>` tags.
2. The permissions `android.permission.INTERNET` and `android.permission.READ_PHONE_STATE` have been added to the AndroidManifest.xml file and are **outside** the `<application>` tags.

   If the meta-data and permissions are correctly set, then it is most likely that the ad networks you selected are not yet active for your application. However, there is another way to test your application for receiving ads from the Mobclix Exchange SDK for Android. In the `AndroidManifest.xml`, try replacing the `APPLICATION_ID` parameter with this string: "`insert-your-application-key`". The string is a test id that should allow Mobclix banners to appear in the AdView. If a banner ad appears, the AdView has been implemented correctly and "real" ads will start appearing after the Ad Networks you selected in the Mobclix Developer Dashboard have approved your application.

Support

If you have any additional questions, Mobclix offers these available resources to you:

- The Mobclix Example Application, which is included in the download for the Mobclix Exchange SDK for Android shows the SDK fully integrated and running. The filepath is Mobclix Android SDK > MobclixDemo.
- 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 filepath is: Documentation > index.html > Full SDK Documentation. Alternatively, you can also view the SDK API documentation online.
- If you have any other questions, send email to **support@mobclix.com** and we'll be happy to help!

Glossary

interstitial

A full screen ad.

Open Allocation

An optional feature that allows you to include the Mobclix Exchange SDK for Android and other 3rd-party SDKs in your application. When you implement Open Allocation, it allows your application to receive ads from both Mobclix and non-Mobclix ad networks.

ad unit

The combination of the content for the ad and the size of the ad.

optimization

The automated mediation service that Mobclix Ad Exchange provides to find the highest-paying ad for a Publisher's application.

adView

The object in your application that represents the space where the ad from Mobclix Exchange SDK for Android will be placed.