AdFalcon Android SDK 4.0.2 Developer's Guide

AdFalcon Mobile Ad Network

Product of

Noqoush Mobile Media Group

Table of Contents

1 Introduction

This guide contains all the information needed by Android developers to integrate with the AdFlacon Network. The guide will also show examples and code snippets on how to perform the integration.

Prerequisites

- Android API v14 or above
- Android Studio 2.0.0 or above.
- Gradle version 3.0.0 or above.
- Google play services ads 8.0.0 or above.

2 Project Configurations

This section illustrates how to integrate the AdFalcon Android SDK with your app.

Step 1: Add the SDK to your project

- **Pull the SDK via maven (Recommended)**
  Follow the steps below to pull the SDK from the "maven" repository

  a. Open your project's build.gradle, and update the "allprojects" block, as follows

```
allprojects {
   repositories {
      jcenter()
      maven {
         url "https://cdn01.static.adfalcon.com/sdk/android/maven"
      }
   }
}
```

  b. Update the "dependencies" block of your module's build.gradle file, as follows:

```
dependencies {
        // ... other dependencies
        compile 'com.noqoush.adfalcon.android.sdk:adfalcon-sdk:4.0.2'
}
```

  c. Sync the project with Gradle files.

- **Adding AdFalcon SDK library using AAR file**

    a. Download the AAR file from the link below https://cdn01.static.adfalcon.com/sdk/android/maven/com/noqoush/adfalcon/android/sdk/adfalcon-sdk/4.0.2/adfalcon-sdk-4.0.2.aar
    b. Click File > New Module.
    c. Click Import .JAR/.AAR Package then click Next.
    d. Select the location of AdFalcon SDK AAR file then click Finish.
    e. Make sure the library "adfalcon-sdk-4.0.2" is listed at the top of your "settings.gradle" file, as follows:

```
include ':app', ':adfalcon-sdk-4.0.2'
```

    f. Open the app module's "build.gradle" file and add a new line to the "dependencies" block, as follows:

```
dependencies {
        compile project( ': adfalcon-sdk-4.0.2 ')
}
```

    g. Sync the project with Gradle files.

For more information please refer to https://developer.android.com/studio/projects/android-library.html#AddDependency

Step2: Configure Manifest

The instructions in this document assume that you are using Android Studio and Gradle. All the required AndroidManifest.xml entries (permissions and activities) for the SDK will be added to your app automatically.

1. **Configure Application Permissions (Optional)**
   The Required permissions were added to the AndroidManifest.xml of the SDK, so no need to add them again.

   The below permissions are optional but recommended.

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

   Note: To enable any of the permission(s) above in Android API level 23 (Marshmallow) and above, you must ask the user to grant the permission.

   For more information about how to request the permissions, please refer to https://developer.android.com/training/permissions/requesting.html#perm-request

2. **Configure Orientation Changes Handling**
   It is highly recommended to prevent the system from restarting your activity when the screen orientation changes. Add the below attribute to your activity tag:

```
android:configChanges=
"keyboard|keyboardHidden|orientation|uiMode|screenLayout|screenSize|smallestScreenSize"
```

   For more information please refer to
   http://developer.android.com/guide/topics/resources/runtime-changes.html

3. **Hardware Acceleration**
   AdFalcon network supports HTML 5 video ads which requires enabling hardware acceleration.

   We recommended that the developer enables hardware acceleration in his application by adding the below property to application tag or to each activity that contains AdFalcon banner.

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

4. **Adding Google Play Services**
   The terms and conditions of Google Play requires using the Advertising ID to identify and track a user. The Advertising Id is included in Google Play services library, so you must add Google Play services library to your application to integrate with AdFalcon

Android SDK. Visit the below link for instruction on how to add Google play services library:

http://developer.android.com/google/play-services/setup.html

Step3: Configure Proguard

To safely use Proguard with AdFalcon SDK, Google Mobile Ads and AdFalcon Mediation Adapter, add the following to your Proguard config:

```
-keep class com.noqoush.adfalcon.android.sdk.** {*;}
-keep class com.google.ads.mediation.adfalcon.** {*;}
-keep public class com.google.android.gms.ads.** {
   public *;
}
-keep public class com.google.ads.** {
   public *;
}
```

3 Banner Ads

Adding Banner Ads

The section below illustrates the simplest way to add Banner Ads to your application

A – Without XML layout

Add the following snippet of code to your activity to add, initialize and load ad view.

```
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Vector;
import android.widget.LinearLayout;
import com.noqoush.adfalcon.android.sdk.ADFListener;
import com.noqoush.adfalcon.android.sdk.ADFTargetingParams;
import com.noqoush.adfalcon.android.sdk.ADFView;
import com.noqoush.adfalcon.android.sdk.constant.ADFAdSize;
import com.noqoush.adfalcon.android.sdk.constant.ADFErrorCode;
.
.
.

// Add private member variable of type ADFView
private ADFView adFalconView;

public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
setContentView(R.layout.main);
        try {
                // create new instance of ADFView
                adFalconView = new ADFView(this);

                //Ensure test mode is set to false before your app is released
                adFalconView.setTestMode(false);

                // initialize the view by passing a site id, ad unit size and enable auto refresh.
                // then load first ad
        adFalconView.initialize("your site ID", ADFAdSize.AD_UNIT_320x50, null, null, true);

                // Add ADFView to this activity.
                // we consider there is a layout naming linearLayout and
                // you want to add ADFView in this layout
        ((LinearLayout) findViewById(R.id.linearLayout)).addView(adFalconView);

        } catch (Exception ex) {

        }
}

@Override
protected void onDestroy() {
adFalconView.destroy();
        super.onDestroy();
}
```

Note: *Refer to the appendix ADFAdSize for the available Ad Unit sizes.*

Note: *Ensure test mode is set to true only during development. Before shipping to production set it to false.*

B – With XML layout

Follow the steps below to add AdFalcon Banner Ad view:

1. Add ADFView to your activity XML layout

```
<com.noqoush.adfalcon.android.sdk.ADFView
        android:id="@+id/adFalconView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"/>
```

2. Add the following snippet of code to your activity class to initilize, load and render the ad view

```
import java.text.SimpleDateFormat;
import java.util.Date;
import java.util.Vector;
import android.widget.LinearLayout;
import com.noqoush.adfalcon.android.sdk.ADFListener;
import com.noqoush.adfalcon.android.sdk.ADFTargetingParams;
import com.noqoush.adfalcon.android.sdk.ADFView;
import com.noqoush.adfalcon.android.sdk.constant.ADFAdSize;
import com.noqoush.adfalcon.android.sdk.constant.ADFErrorCode;
.
.
.

// Add private member variable of type ADFView
private ADFView adFalconView;

public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
setContentView(R.layout.main);
        try {

                // create new instance of ADFView
                adFalconView = ((ADFView)findViewById(R.id.adFalconView));

                //Ensure test mode is set to false before your app is released
                adFalconView.setTestMode(true);

                // initialize the view by passing a publisher id, ad unit size, params,
                // listener and enable auto refresh.
                // then load first ad
        adFalconView.initialize("your site ID", ADFAdSize.AD_UNIT_320x50, null, null, true);

        } catch (Exception ex) {

}
}

@Override
protected void onDestroy() {
        adFalconView.destroy();
        super.onDestroy();
}
```

Note: *Refer to the appendix ADFAdSize for the available Ad Unit sizes.*

Note: *Ensure test mode is set to true only during development. Before shipping to production set it to false.*

Advanced Banner Ads Options

Tracking Ad Lifecycle Events – ADFlistener

ADFListener provides ability to track Ad lifecycle events. Implement the following abstract methods.

```
public class MainActivity extends Activity implements ADFListener{

public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
setContentView(R.layout.main);
        try {

                // create new instance of ADFView
                adFalconView = ((ADFView)findViewById(R.id.adFalconView));

                //Ensure test mode is  set to false before your app is released

                adFalconView.setTestMode(true);

                // initialize the view by passing a publisher id, ad unit size, params,
                // listener and enable auto refresh.
                // then load first ad
        adFalconView.initialize("your site ID", ADFAdSize.AD_UNIT_320x50, null, this, true);

        // set the listener to ADFView
        adFalconView.setListener(this);

        } catch (Exception ex) {

}
}

public void onLoadAd(ADFAd ad) {
}

public void onError(ADFAd ad, ADFErrorCode code, String message) {
}

public void onPresentAdScreen(ADFAd ad) {
}

public void onDismissAdScreen(ADFAd ad) {
}

public void onLeaveApplication() {
}
.
.
```

ADFAdSize

ADFAdSize Enum defines the ad units supported by AdFalcon.

Enum	Name	Size	Devices
AD_UNIT_AUTO_BANNER	Auto Banner	The server will detect the best ad unit size based on the screen size and orientation.	All
AD_UNIT_320x50	Standard	320 x 50	All
AD_UNIT_300x250	Medium Rectangle	300 x 250	All
AD_UNIT_468x60	Full Banner	468 x 60	Tablet

AD_UNIT_728x90	Leaderboard	728 x 90	Tablet
AD_UNIT_120x600	Skyscraper	120 x 600	Tablet

Refresh Ad

Use the below methods to configure ad refresh parameters.

Method	Description
public void refreshAd()	Used to get a new ad programmatically.
public void setRefreshDuration(**int** duration)	Determine the refresh duration in seconds among the receivedAds. Note: The duration range starts from 30 to 120 seconds.
public void setEnableAutoRefresh(**boolean** enable)	Used to enable/disable the auto refresh for the received ads.

Disable Hardware Acceleration

If you noticed flickering in the ad view or any other user interface element in your activity, you can disable hardware acceleration by implementing the below snippet of code. However, this is not recommended because HTML 5 video ads will not play correctly.

```
ADFView.setLayerType(View.LAYER_TYPE_SOFTWARE, null);
```

4 Interstitial Ad

Interstitial is a full screen ad that appears between levels, before or after viewing content. To add interstitial ad, you have to perform the steps below:

1. Implement ADFListener

```
public class MainActivity extends Activity implements ADFListener{
.
.
        public void onLoadAd(ADFAd ad) {
        }

        public void onError(ADFAd ad, ADFErrorCode code, String message) {
        }

        public void onPresentAdScreen(ADFAd ad) {
        }

        public void onDismissAdScreen(ADFAd ad) {
        }

        public void onLeaveApplication() {
        }
}
```

2. Create instance of ADFInterstitial and load new interstitial ad.

```
protected void onCreate(Bundle savedInstanceState) {
        .
        .
        ADFInterstitial adfInterstitial = new ADFInterstitial(this, "your site ID",    this);
        adfInterstitial.loadInterstitialAd();
        .
        .
}
```

3. When the interstitial is loaded successfully and becomes ready to display the Ad, the SDK will fire onLoadAd method. Therefore, you will need to implement onLoadAd interface's method as below:

```
public void onLoadAd(ADFAd ad) {
        if(ad instanceof ADFInterstitial){
                ((ADFInterstitial)ad).showInterstitialAd();
        }
}
```

4. To know if any error has occurred during the loading including “no available ad”. You will need to implement onError interface’s method as the below

```
public void onError(ADFAd ad, ADFErrorCode code, String message) {
        if(ad instanceof ADFInterstitial){
                if(code == ADFErrorCode.NO_AD_AVAILABLE){
                        //Do anything here
                }else if(code == ADFErrorCode.COMMUNICATION_ERROR){
                        //Do anything here
                }else if(code == ADFErrorCode.MISSING_PARAM){
                        //Do anything here
                }else{
                        //Do anything here
                }
        }
}
```

To learn more about Errors codes go to: ADFErrorCode

5 Appendix

ADFAdSize enum

ADFAdSize Enum defines the ad units supported by the AdFalcon.

Enum	Name	Size	Devices
AD_UNIT_AUTO_BANNER	Auto Banner	The server will detect the best ad unit size based on the screen size and orientation.	All
AD_UNIT_320x50	Standard	320 x 50	All
AD_UNIT_300x250	Medium Rectangle	300 x 250	All
AD_UNIT_468x60	Full Banner	468 x 60	Tablet
AD_UNIT_728x90	Leaderboard	728 x 90	Tablet
AD_UNIT_120x600	Skyscraper	120 x 600	Tablet

ADFErrorCode enum

ADFErrorCode Enum defines AdFalcon SDK and Ad Server error codes.

Enum	Description
GENERIC_SDK_ERROR	Error occurred in the SDK.
INTERNAL_SERVER_ERROR	Error occurred in AdFalcon server.
COMMUNICATION_ERROR	SDK can not reach the AdFalcon server; mostly for issues related to the phone internet connection.
NO_AD_AVAILABLE	No ad is available in the AdFalcon store that meets the ad request parameters.
INVALID_PARAM	An invalid ad request parameter has been set.
MISSING_PARAM	A required ad request parameter has not been set.

ADFListener interface

ADSListener is an interface that is used to get feedback from the AdFalcon SDK.

Method	Description
public void onLoadAd(ADFAd ad)	SDK has loaded an ad.
public void onError(ADFAd ad, ADFErrorCode error, String message)	SDK has encountered an error during the loading of an ad. **Params** view: AdFalcon view error: type of error message: description for error
public void onPresentAdScreen(ADFAd ad)	Ad screen is displayed.
public void onDismissAdScreen(ADFAd ad)	Ad screen has closed.

public void onLeaveApplication()	The SDK will open the ad in another application.

ADFTargetingParams class

The ADFTargetingParams Class contains all the needed parameters about any given user and application in order to help AdFalcon network to send the most relevant and targeted ad to the user. All parameters are optional

Variable	Required	Description	Values
Language	No	A two-characters string parameter indicating Ad language	ar, en
Postal code	No	A string parameter containing the user's postal/ZIP code	11121
Area code	No	A string parameter containing the user's area code	06
Age	No	An integer parameter containing the user's age	27
Keywords	No	List of keywords in comma separated format. AdFalcon's ads selector engine will search for ads containing these keywords.	ex. sport, news, lifestyle, …etc.
Gender	No	A parameter containing the user's gender	NONE GENDER_MALE GENDER_FEMALE
Country code	No	County code of the end user in ISO 3166-1 alpha-2 format code (two-letter code)	JO, SA …etc.
Birthdate	No	Birthdate of application user in format dd.MM.yyyy	
Additional Info	No	A Map of keys and values to add additional parameters.	
Location: Latitude, Longitude	No	The geolocation information of the device. The location information is divided into two double values; latitude and longitude.	35.654, 34.6598

ADFView class

ADFView is the main SDK class and it is responsible for initializing, loading and rendering an ad.

Method	Description
public void initialize(String siteID, ADFAdSize adSize, ADFTargetingParams params, ADFListener listener, **boolean** autorefresh)	This is the main method in ADFView class and is used to initialize the SDK, load the first Ad and start the auto refresh timer. You can call this method only one time. **Parameters** ▪ site ID: You can get this ID from AdFalcon website and it is unique for each application. ▪ adSize: The size of the needed ad. ▪ params: contains all the needed params to help AdFalcon network to detect and better target an Ad. ▪ listener: SDK feedback. ▪ autorefresh: used to enable auto refresh for ads.
public void refreshAd()	Used to get a new ad.
public void setTestMode(**boolean** test)	Used to inform AdFalcon network if your application is working in test or production mode.
public void setRefreshDuration(**int** duration)	Determine the refresh duration in seconds among the receivedAds.
public void setEnableAutoRefresh(**boolean** enable)	Used to enable/disable the auto refresh for the received ads.
public boolean isEnableAutoRefresh()	Used to know if auto refresh timer is running or paused.
public void setEnableAnimation(**boolean** enable)	Used to Enable/Disable animation.
public boolean isEnableAnimation()	Returns true if the animation is enabled otherwise disabled.
public void setListener(ADFListener listener)	Set ADFListener.
public ADFListener getListener()	Get ADFListener.
public void destroy()	This method must be called either in onDestroy() method to finalize the ADFView or when you do not need to use the ADFView again. After calling this method the SDK will kill any timer, thread or running process.

ADFInterstitial class

This class is responsible for creating, loading and rendering interstitial ads.

Method	Description

public ADFInterstitial(Context context, String siteID, ADFListener listener,)	The interstitial class constructor and is responsible for initializing the interstitial ad with: • context • your site ID • listener
public ADFInterstitial(Context context, String siteID, ADFListener listener, **boolean** testMode)	Class constructor overload which initializes the interstitial ad with: • context • your site ID • listener • Test Mode Flag (Pass true during development only, and false for production release).
public ADFInterstitial(Context context, String siteID, ADFListener listener, ADFTargetingParams targetingParams)	The interstitial class constructor and is responsible for initializing the interstitial ad with: • context • your site ID • listener • targeting parameters
public ADFInterstitial(Context context, String siteID, ADFListener listener, ADFTargetingParams targetingParams, **boolean** testMode)	Class constructor overload which initializes the interstitial ad with: • context • your site ID • listener • targeting parameters • Test Mode Flag (Pass true during development only, and false for production release).
public void loadInterstitialAd()	This method is used to load new interstitial ad. You can track if the ad has been loaded successfully by implementing ADSListener.onLoadAd method.
public void showInterstitialAd()	This method is used to display the interstitial in full screen mode. But you cannot call this method if the ad is not loaded successfully. We recommended You to check isReady() method before calling it.
public boolean isReady()	This method is used to check if the interstitial ad is ready to display.
public void setTestMode(**boolean** testMode)	This method is used to set test mode flag.

6 More Information:

You can find more information in the sample project within the downloaded zip file.

For any SDK integration queries, please send us an email to support@adfalcon.com along with your login id.