Freestar Ads Mediation Native Android

Last Modified on 05/31/2022 12:18 pm EDT

Change Log

Freestar provides an effective ad mediation solution. The Freestar mediation method is universal auction, not the traditional waterfall. Universal auction is more sophisticated than waterfall and provides, by far, the best eCPM. This document describes how to integrate the Freestar SDK into your native Android app quickly and easily. This repo is a fully integrated Android sample app. Feel free to clone it, open with Android Studio and run it on a device.

> Note: You can remotely toggle on/off any of the following ad providers as you see fit using our web dashboard. *All* applicable providers are enabled by default.

Supported Ad Partners

Ad Provider	SDK Version	Ad Unit Types
Amazon TAM	9.2.1	Fullscreen Interstitial, Banner 300x250, 320x50, 728x90
AdColony	4.6.5	Fullscreen Interstitial & Rewarded, Banner 300x250, 320x50, 728x90
AppLovinMAX	11.3.1	Fullscreen Interstitial & Rewarded, Banner 300x250, 320x50, 728x90
Criteo	4.4.0	Fullscreen Interstitial & Rewarded, Banner 300x250, 320x50
Admob	20.6.0	Fullscreen Interstitial & Rewarded, Banner 300x250, 320x50, 728x90, Native
Google Ads Manager	20.6.0	Fullscreen Interstitial & Rewarded, 300x250, 320x50, 728x90, Native
Google IMA SDK	3.23.0	Preroll
Nimbus	1.10.10	Fullscreen Interstitial & Rewarded, Banner 300x250, 320x50, 728x90, Preroll
Tapjoy	12.4.2	Fullscreen Interstitial & Rewarded
Unity Ads	4.1.0	Fullscreen Interstitial & Rewarded, Banner 320x50, 728x90
Vungle	6.10.4	Fullscreen Interstitial & Rewarded, Banner 300x250, 320x50, 728x90
Pangle	3.6.0	Fullscreen Interstitial & Rewarded, Banner 320x50, 300x250, Native
HyprMX	6.0.1	Fullscreen Interstitial & Rewarded, 300x250, 320x50, 728x90
Yahoo/Verizon	1.14.0	Fullscreen Interstitial, 300x250, 320x50

Getting Started

Follow the simple steps below to start monetizing with Freestar Ads in your native Android app today!

Project Setup

Modify Gradle

Merge in the following repositories to your TOP-LEVEL build.gradle:

https://github.com/freestarcapital/SDK_documentation_Android/blob/master/build.gradle (top-level build.gradle example)

```
repositories {
        google()
        mavenCentral()
        maven { url 'https://jitpack.io' }
        maven { url 'https://freestar.jfrog.io/artifactory/freestar-mediation-android-sdk' }
        maven { url 'https://s3.amazonaws.com/moat-sdk-builds' }
        maven { url 'https://sdk.tapjoy.com/' }
        maven { url 'https://artifact.bytedance.com/repository/pangle' }
        maven { url 'https://hyprmx.jfrog.io/artifactory/hyprmx' }
        maven { url 'https://adsbynimbus-public.s3.amazonaws.com/android/sdks' }
        maven { url 'https://artifactory.verizonmedia.com/artifactory/maven/' }
}
```

Use build tools version 4.0.1 or greater:

```
buildscript {

    dependencies {
        classpath("com.android.tools.build:gradle:4.0.1")  //Make sure to use 4.0.1  or greater
        // NOTE: Do not place your application dependencies here; they belong
        // in the individual module build.gradle files
    }
}
```

Modify app-level build.gradle

Merge the following into your APP-LEVEL build.gradle:

```
android {

    compileSdkVersion 31

    defaultConfig {

        minSdkVersion 19  //Important: 19 is the minimum SDK int level supported!
        targetSdkVersion 31

        multiDexEnabled true
    }

    compileOptions {

        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8

    }

    packagingOptions {
        exclude 'META-INF/*.kotlin_module'
    }

}
```

Dependencies

Merge the following dependencies into your APP-LEVEL build.gradle:

https://github.com/freestarcapital/SDK_documentation_Android/blob/master/app/build.gradle (example app-level build.gradle)

```
dependencies {
    implementation fileTree(dir: 'libs', include: ['*.jar'])

    //begin Freestar
    implementation 'com.freestar.android.ads:freestar:4.6.4'
    implementation 'com.freestar.android.ads.admob:freestar-admob-adapter:20.6.0.0'
    implementation 'com.freestar.android.ads.applovin:freestar-applovin-adapter:11.2.2.0'
    implementation 'com.freestar.android.ads.applovinmax:freestar-applovinmax-adapter:11.2.2.0'
    implementation 'com.freestar.android.ads.tam:freestar-tam-adapter:8.4.3.3'
    implementation 'com.freestar.android.ads.adcolony:freestar-adcolony-adapter:4.6.5.0'
    implementation 'com.freestar.android.ads.google:freestar-google-adapter:20.6.0.0'
    implementation 'com.freestar.android.ads.criteo:freestar-criteo-adapter:4.2.1.2'
```

```
implementation 'com.freestar.android.ads.unityads:freestar-unity-adapter:4.1.0.0'
implementation 'com.freestar.android.ads.vungle:freestar-vungle-adapter:6.10.4.0'
implementation 'com.freestar.android.ads.tapjoy:freestar-tapjoy-adapter:12.2.1.1'
implementation 'com.freestar.android.ads.nimbus:freestar-nimbus-adapter:1.10.4.1'
implementation 'com.freestar.android.ads.pangle:freestar-pangle-adapter:3.6.0.0'
implementation 'com.freestar.android.ads.hyprmx:freestar-hyprmx-adapter:6.0.1.0'
implementation 'com.freestar.android.ads.yahoo:freestar-yahoo-adapter:1.14.0.0'

implementation 'com.applovin:applovin-sdk:11.3.1'
implementation 'com.criteo.publisher:criteo-publisher-sdk:4.4.0'
implementation 'com.google.android.gms:play-services-ads:20.6.0'
implementation 'com.google.ads.mediation:facebook:6.8.0.0' //fb fill via gam
//implementation('com.google.android.ads.consent:consent-library:1.0.8') {
//    exclude group: 'com.android.support'
//    exclude group: 'com.google.code.gson', module: 'gson'
//}
implementation('com.facebook.android:audience-network-sdk:6.8.0') {
    exclude group: 'com.google.android.exoplayer'
    exclude group: 'com.google.android.gms'
    exclude group: 'com.android.support'
}
implementation 'com.tapjoy:tapjoy-android-sdk:12.4.2@aar'
implementation 'com.vungle:publisher-sdk-android:6.10.4'
implementation 'androidx.constraintlayout:constraintlayout:2.0.4' //new dependency requirement

//Note: if you are using Preroll, un-comment the following lines:
//start preroll
/*
implementation 'com.google.ads.interactivemedia.v3:interactivemedia:3.23.0'
implementation 'com.google.android.exoplayer:exoplayer-core:2.13.3'
implementation 'com.google.android.exoplayer:exoplayer-ui:2.13.3'
implementation 'com.google.android.exoplayer:extension-ima:2.13.3'
*/
//end preroll

//nimbus
implementation "com.adsbynimbus.android:nimbus:1.10.4" // Full Nimbus SDK with NimbusAdManager
implementation "com.adsbynimbus.android:extension-okhttp:1.10.4" // Use OkHttp with NimbusAdManager
implementation "com.adsbynimbus.android:extension-facebook:1.10.4"
implementation "com.adsbynimbus.openrtb:kotlin:0.7.4"

//Amazon Transparent Ads Marketplace
implementation 'com.amazon.android:aps-sdk:9.2.1@aar'

//Pangle; also requires play-services-ads-identifier but already in Criteo
implementation 'com.pangle.global:ads-sdk:3.6.0.4'

//adcolony
implementation 'com.adcolony:sdk:4.6.5'

//hyprmx
implementation 'com.hyprmx.android:HyprMX-SDK:6.0.1'

//yahoo/verizon
implementation 'com.verizon.ads:android-vas-standard-edition:1.14.0'
implementation 'androidx.browser:browser:1.4.0'


//REACT-NATIVE NOTE: Uncomment the line below if your project is React Native
//implementation 'com.freestar.android.ads:react-native-android:1.1.1'

//App Open Ads (startup ads)
implementation "androidx.lifecycle:lifecycle-runtime:2.4.0"
implementation "androidx.lifecycle:lifecycle-common-java8:2.4.0"
implementation "androidx.lifecycle:lifecycle-process:2.4.0"

//GAM mediation adapters
implementation 'com.google.ads.mediation:applovin:11.1.2.0'
implementation 'com.google.ads.mediation:adcolony:4.6.5.0'
implementation 'com.google.ads.mediation:tapjoy:12.9.0.0'
implementation 'com.google.ads.mediation:facebook:6.8.0.0'

//end, Freestar
```

AndroidManifest.xml

> Note: The values will be different for your final production build.

Merge the following <meta-data> tags into the <application> tag of your **AndroidManifest.xml**

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.yourcompany.yourapp">


   <uses-permission android:name="com.google.android.gms.permission.AD_ID" />

   <application

       <meta-data android:name="com.google.android.gms.version" android:value="@integer/google_play_services_ve
rsion"/>

       <meta-data
           android:name="com.google.android.gms.ads.APPLICATION_ID"
           android:value="ca-app-pub-3940256099942544~3347511713"/>

       <meta-data
           android:name="applovin.sdk.key"
           android:value="hO52kFtMvEo_AoeRzED0_XXfS1B1VQp9GW50yudJO-eUUTOmRBLI3c-2GyTevLNspll_fN5PLTbA
HOakoTuHuP" />


   </application>
```

See the AndroidManifest.xml:

https://github.com/freestarcapital/SDK_documentation_Android/blob/master/app/src/main/AndroidManifest.xml

Initialize Freestar Ads

Freestar Ads must be initialized before you start requesting ads.

```
       //You may use application or activity context
       adRequest = new AdRequest(context);
       adRequest.addCustomTargeting("some target", "my value"); //optional
       FreeStarAds.init(context, "XqjhRR", adRequest);    //Use our test key until our solutions team creates your own ke
y!

       //If you do not have custom targeting parameters, then you
       //can just call FreeStarAds.init(context, "XqjhRR")
```

Custom Targeting

Freestar Ads allows for custom targeting parameters that will be sent to our Google Ads Manager adapter.

```
       adRequest = new AdRequest(context);
       adRequest.addCustomTargeting("target1", "value1"); //optional
       adRequest.addCustomTargeting("target2", "value2"); //optional
```

App Open Ad

App Open Ad is a fullscreen ad that shows when the app starts (including 'cold starts') and when the app is resumed from the background.

To utilize App Open Ads in your application, call the following method before calling FreeStarAds.init as follows:

```
FreeStarAds.requestAppOpenAds("app-open-ad-placement", true, null);   //Obtain a proper placement from our Solutio
ns Team
FreeStarAds.init(...);
```

Interstitial Ad

Note: Please do not "prefetch" the next interstial ad on app startup or after dismissals or no-fills; we do this automatically and internally for you.

```
InterstitialAd interstitialAd = new InterstitialAd(this, this); //Requires Activity context
interstitialAd.loadAd(adRequest);

//You can also load associated to a placement as follows
//interstitialAd.loadAd(adRequest, "my_placement_p1");
```

> If you plan to use more than one placement in your app, please adhere to the placement naming convention as follows:
>
> "my_placement_name_pN", where N is the number of your placement.
>
> For example, let us assume you are using 2 interstitial ad placements in your game or app. The first placement would be the default placement; simply do not specifiy a placement name by using the **loadAd()** method without the placement parameter. The second placement would be, for example, "my_search_screen_p1". The ending "p1" tells the SDK to use the second placement you created in our web dashboard for the interstitial ad unit.
>
> This process is the same for all the other ad units, such as rewarded ads and banner ads.

When the interstitial ad is ready, the **onInterstitialLoaded** callback will occur.

```
  @Override
  public void onInterstitialLoaded(String placement) {
     interstitialAd.show();  //You can display the ad now OR show it later; your choice.

     //Note: Placement will be null if not specified in the original loadAd request.

  }
```

There are other callbacks that will occur in other events, such as in the rare event where a load ad request does not result in a fill. Please see the MainActivity on this sample for those details.

> ⚠Warning: Attempting to load a new ad from the onInterstitialFailed() method is **strongly** discouraged. If you must load an ad from onInterstitialFailed(), limit ad load retries to avoid continuous failed ad requests in situations such as limited network connectivity.

Banner Ad

Freestar supports **300x250** and **320x50** banner ad formats and allows you to control the refresh intervals remotely.

```
     BannerAd bannerAd = new BannerAd(this);  //in Activity
     bannerAd.setAdSize(AdSize.BANNER_320_50);
     bannerAd.loadAd(adRequest);

     //Note: you may pass in a "placement" parameter in loadAd but it requires prior remote staff setup
```

When the banner ad is ready, the **onBannerAdLoaded** callback will occur.

```
  @Override
  public void onBannerAdLoaded(View bannerAd, String placement) {

     //Note: Placement will be null if not specified in the original loadAd request.
     //'banner_container' is your ad container

     ((ViewGroup) findViewById(R.id.banner_container)).removeAllViews();
     ((ViewGroup) findViewById(R.id.banner_container)).addView(bannerAd);   //You MUST add the banner to your con
tainer here!

  }
```

Another way to display banner ads is to put BannerAd directly into your XML layout as follows. **NOTE: If you choose this route, then you do not need any code in java or kotlin as the ad will automatically be fetched.** Also, if you need to set targeting parameters in the AdRequest, then please use the programmatic approach and do not put the banner in the layout.

See **com.freestar.android.ads.BannerAd** in the sample layout:

https://github.com/freestarcapital/SDK_documentation_Android/blob/master/app/src/main/res/layout/activity_main.xml

```
<com.freestar.android.ads.BannerAd xmlns:ads="http://schemas.android.com/apk/res-auto"
android:id="@+id/freestarBannerAd_2"
android:layout_width="320dp"
android:layout_height="50dp"
ads:FreeStarAdSize="BANNER"/>
```

Adaptive Banner Ad (new)

Freestar now supports Adaptive Banner Ads, which are 320x50 banner ads that will automatically fill the full width of the device. AdMob and Google Ads Manager are two supporting demand source partners at this time.
To start displaying Adaptive Banner Ads today, follow the same instructions per our standard banner ads (above). We recommend setting your ad container dimensions to WRAP_CONTENT x WRAP_CONTENT. Then, call the following method before or after our init method:

```
FreeStarAds.init(....)
FreeStarAds.showAdaptiveBannerAdsWhenAvailable( true );  //default is false
```

So, using Adaptive Banner Ads is not much different from our standard 320x50 banner ads.

NOTE: If there is no-fill for Adaptive Banner (from AdMob or Google Ads Manager), then Freestar will fallback to the standard sized banner ad (320x50) provided by our other demand source partners. You may receive a banner ad that fills the full width of the screen OR you may get a banner ad that fills 320x50.

Rewarded Ad

Note: Please do not "prefetch" the next rewarded ad on app startup or after dismissals or no-fills; we do this automatically and internally for you.

> A common myth regarding Rewarded Ads is publishers are required to *give something* to the user. But, that's not true. You can simply tell the user they must watch the ad in order to be able to proceed to the next level or proceed to content.

```
RewardedAd rewardedAd = new RewardedAd(this, this); //Must use Activity context
rewardedAd.loadAd(adRequest);

//You can also load an ad tied to an Interstitial 'placement' as follows
//rewardedAd.loadAd(adRequest, "my_placement_p1"); //'placement' is OPTIONAL and only if
                                                   //you plan to have more than one Rewarded
                                                   //placement
```

When the rewarded ad is ready, the **onRewardedVideoLoaded** callback will occur.

```
@Override
public void onRewardedVideoLoaded(String placement) {
    rewardedAd.showRewardAd("my_optional_secret", "my_optional_userid", "Gold Coins", "100");  //Display Ad

    //Note: Placement will be null if not specified in the original loadAd request.

    //Note: If you want to use server-to-server rewarded callbacks, the Freestar team will provide
    //      you with a secret and only that will work.
    //      However, if you want to use client-side only rewarded callbacks, then you can supply
    //      your own "secret" or null as the first parameter.
}
```

When the user has fully watched the rewarded ad, the following callback will occur:

```
@Override
public void onRewardedVideoCompleted(String placement) {
  //allow user to proceed to app content or next level in app/game
}
```

When the user has closed the rewarded ad, the following callback will occur:

```
@Override
public void onRewardedVideoDismissed(String placement) {

}
```

> If the user does not watch the rewarded ad thru to completion, **onRewardedVideoCompleted** will not occur. However, the **onRewardedVideoDismissed** will always occur when the rewarded ad is dismissed regardless if the user watched the entire rewarded ad or not.

> ⚠ Please assume that ads will expire in about 1 hour after the loaded callback. Meaning, you may *cache* an ad in your app or game, but must be displayed within the alloted hour. Also note that some SDK partners, at their own discretion, may expire their ads earlier than one hour.

Preroll Ad

Freestar supports preroll video ads.

```
    PrerollAd prerollAd = new PrerollAd(this);  //Activity context
    prerollAd.loadAd(adRequest, AdSize.PREROLL_320_480, "optional-placement-string", listener);
```

When the preroll ad is ready, the **onPrerollAdLoaded** callback will occur:

```
@Override
public void onPrerollAdLoaded(View prerollView, String placement) {

    ((ViewGroup) findViewById(R.id.ad_container)).removeAllViews();
    ((ViewGroup) findViewById(R.id.ad_container)).addView(prerollView); //add view to layout
    prerollAd.showAd();  //play the video ad

}
```

Additional Preroll Dependencies

As mentioned in the dependencies section above, please take of note of the preroll-specific dependencies.

Please see the `MainActivity.java` of our sample app at the bottom of this page to see how preroll ads work.

Native Ad Unit

Freestar supports Native Ad format where you may customize the look and feel of the ad itself.

Sample Project

All of this and more, such as *Preroll Ads* can be seen in the sample MainActivity:

https://github.com/freestarcapital/SDK_documentation_Android/blob/master/app/src/main/java/com/freestar/android/sam

Privacy - Google Play Families Policy Compliance

If your game or app is officially under the Google Play Families program, Freestar provides such support:

```
FreeStarAds.setGoogleFamilyPolicyMode( GoogleFamilyPolicyMode.app, true); //If your app is designed only for childre
n
//FreeStarAds.setGoogleFamilyPolicyMode( GoogleFamilyPolicyMode.mixed, false);  //If your app is designed for famili
es with children
FreeStarAds.init(...);
```

If your app is not officially under the Google Play Families program, then you do not need to set the Google Family Policy mode.

```
/**
 * Only set Google Families Policy Mode if your app is required to comply with Google Play's
 * Families Policy program.
 *
 * @param googleFamilyPolicyMode app: the app is child-directed and will not receive
 *                               personalize or contextual ads.
 *                               mixed: the app is directed at mixed audiences.
 *                               none: (default) the app is not required to comply with Google
 *                                     Play's Family Policy
 *
 * @param onlyNonPersonalizedAds true: if 'mixed' mode, then only personalized or contextual ads
 *                                     may be served.
 *                               false: if 'mixed' mode, then personalized or contextual ads
 *                                     may be served.
 *                               note:  if 'app' mode, then personalized or contextual ads
 *                                     may not be served regardless of this parameter.
 */
public static void setGoogleFamilyPolicyMode(GoogleFamilyPolicyMode googleFamilyPolicyMode,
                                             boolean onlyNonPersonalizedAds)
```

GDPR

Freestar SDK is GDPR compliant. In order for your users to be able to receive any ad fills in GDPR-affected countries, you, as a publisher, will need to implement a 3rd party Consent Management Platform (CMP). Freestar SDK will automatically detect user interactions with the CMP and act accordingly. Please see our Freestar GDPR Frequently Asked Questions for complete details and our recommended list of CMP service providers.

Testing

For Android, please use our test key **XqjhRR** for all your testing runs and enable test mode true. You will usually get 100% fill on all ad units. It is not recommended to use your production key for testing runs as that is strictly prohibited by our partners and bad things may happen to us on the business side of things.
Do not forget to uninstall and re-install your app when changing keys on your device.

When you are satisfied with your testing, please make a release build with your production key, and turn test mode off. Publish to store.

Automated Testing - Bypassing Ads

Are your automated tests failing after integrating Freestar Mediation Ads into your mobile application or game? Are you not sure it could be due to Freestar or something else? We have a feature called **Automated Test Mode** where you can run your automated tests to bypass Freestar or run Freestar in 'Limited Mediation' mode without making drastic changes to your code:

In your automated test suite code, **before** FreeStarAds.init is called:

```
FreeStarAds.setAutomatedTestMode( FreeStarAds.AutomatedTestMode.BYPASS_ALL_ADS )
  //OR
FreeStarAds.setAutomatedTestMode( FreeStarAds.AutomatedTestMode.LIMITED_MEDIATION )    //only runs AdMob &
GAM
```

Again, this is only for your automated tests and not for production use.

Release Build

Final note when creating your release build: if you use **proguard or minify**, please add our proguard

rules to your app proguard file.