

# MAX Adapter Installation

Android MAX Adapter Installation

## Adding AdView to MAX

### Step 1: Log in to your AppLovin account and select MAX to configure AdView

As explained above, you must have an AppLovin account and select the MAX option in order to integrate AdView.

### Step 2: Create a new AD unit ID

Under the MAX subtab, select Ad Units, and then click "Create Ad Unit"

Application	Ad Unit	Ad Unit ID	Ad Type	7d Earnings	Status
applovin demo app	AdView Mobile	8eba456	INTER	\$0.10	Active
applovin demo app	AdView Mobile	8c0f35f3	BANNER	\$0.15	Active
applovin demo app	AdView Mobile	71b05d	MREC	\$0.10	Active

Provide a name for your new AD unit. After selecting the device type, enter your application name, package name, bundle ID, or digital iTunes ID, and select the AD type, then go to the bottom and click "Save"

## Create New Ad Unit

Name

Enter Ad Unit Name

Platform ⓘ

☒ Android

☐ Fire OS

☐ iOS

Enter your application name, package name, bundle ID, or numerical iTunes ID

Ad Type

☒ Banner

☐ Interstitial

☐ MREC

☐ Native

☐ Rewarded

✓ Save

Back

### Native Ad Unit

Ad Type

☐ Banner

☐ Interstitial

☐ MREC

☒ Native

☐ Rewarded

Template

☐ Small ⓘ

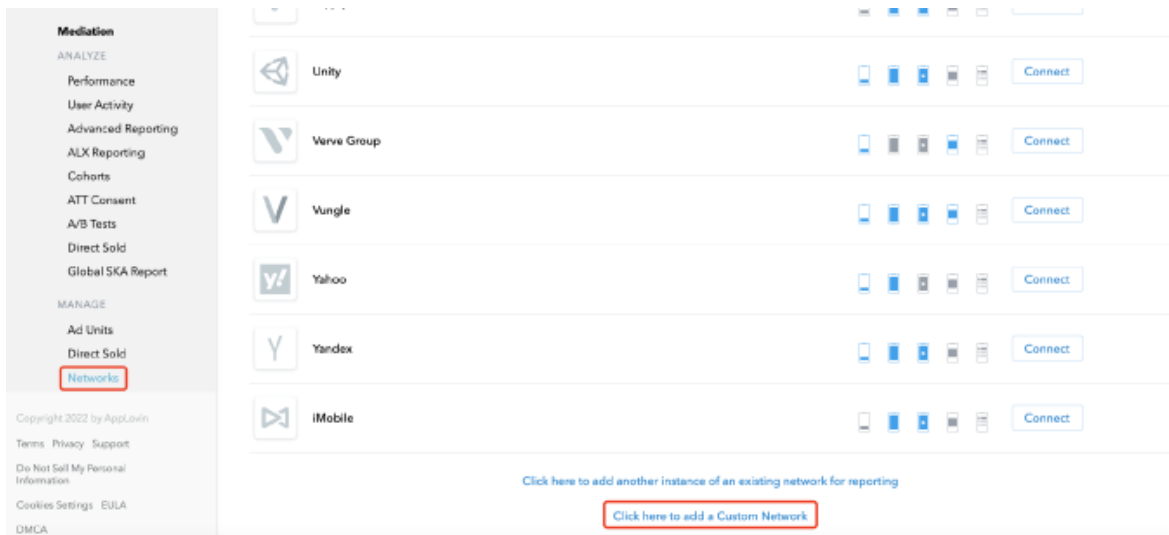
☒ Medium ⓘ

☐ Manual ⓘ

for more native detail mode, pls refer to following 5.0 .

## Step 3: Add a custom network to your AD unit

Under the MAX subtab, Click the Networks option, then go to the bottom and Click "Click here to add a Custom Network".



Provide a name for your new network, select "SDK" for network type, add the following class to the "Android / Fire OS Adapter Class Name" section for your Network, then click "Save"

## Manage Network

### Network Type

- ☐ JS Tag
- ☒ SDK

### Custom Network Name ⓘ

AdView         Banner Android

### iOS Adapter Class Name

### Android / Fire OS Adapter Class Name

com.applovin.mediation.adapters.AdVgMediationAdapter

 Save

Back

(pic 3-1)

Android / Fire OS Adapter Class Name

com.applovin.mediation.adapters.AdVgMediationAdapter

Now click Ad Units under MAX, and then click the Ad unit you just created

**MAX Ad Units**

Search

Application	Ad Unit	Ad Unit ID	Ad Type	7d Earnings	Status
applo...lepp	AdView MAX Adapter Interstitial Android	8abs45...c3e	INTER	-	●
app...pp	AdView MAX Adapter Banner Android	8c0f35f3...	BANNER	\$0.15	●
applc...pp	AdView MAX Adapter MREC Android	71b05d...e2f	MREC	\$0.10	●

Ad Units

Find "Custom Networks & Deals", select the network you just created, turn on the switch, configure the APPID and Placement ID, and set the CPM price, then click the "Save" button in the lower left corner

#### Custom Networks & Deals

Custom Network (SDK) - AdView SDK Medi... Android

Custom Network (SDK) - AdView SDK Medi... Android

Custom Network (SDK) - AdView SDK Medi... droid

Custom Network (SDK) - AdView SDK Medi... droid

Custom Network (SDK) - AdView SDK Medi... Android

☒ Status

App ID (optional) ⓘ

SDK20191630040808jt...oy

Placement ID

POSID63...

Custom Parameters

e.g. {"floor\_price": 7.0

CPM Price ⓘ

\$ 100

Country Targeting ⓘ

Include All

+ Add New Placement ID

Save

Back

Now you have completed the setup, and only need simple integration to display AdView ads.

## MAX Adapter Installation

### Android MAX Adapter Installation

#### Step 1: Get the file with the adapter

When you see this document, you should have obtained the following documents:

- com.adview-4.4.1.aar
- AdViewSDK\_MAXAdapter\_demo

Otherwise please contact your AM or [partner@adview.com](mailto:partner@adview.com).

#### Step 2: Add the AdView SDK into your Project

first, add module 'adview' to build.gradle

```
dependencies {
    implementation project(':adview')
}
```

Add the following code to the project's settings.gradle

```
include ':adview'
project(':adview').projectDir = new File(settingsDir, "../AdView")
```

Next you can have 2 options to handle adview sdk workfull.

## Option 1: Pulling the Latest SDK via mavenCentral

If you are using Gradle to build your Android applications, you can pull the latest version of the SDK from mavenCentral as described below:

1. Include mavenCentral in your top-level `build.gradle` file:

```
allprojects {
    repositories {
        mavenCentral()
    }
}
```

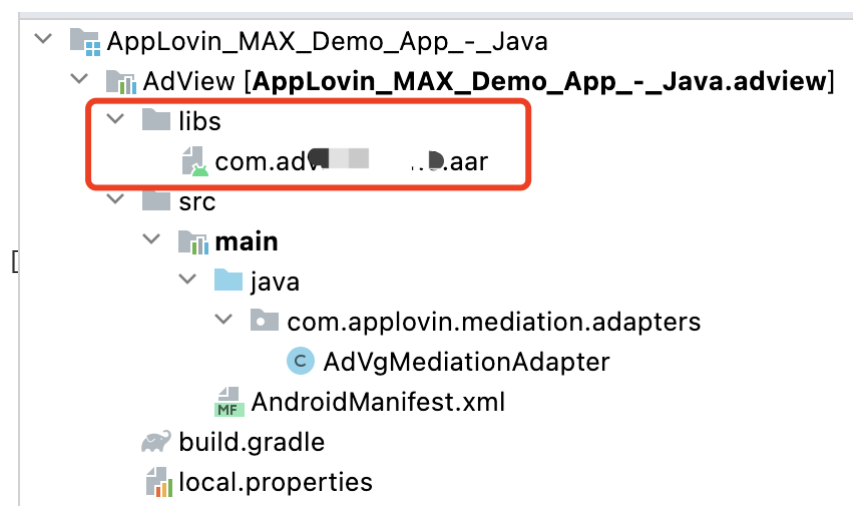
2. Add the following line to the dependencies element in '**adview**' module's `build.gradle`.

```
api 'com.adview:android-sdk:4.4.0'
```

3. Sync your Gradle project to ensure that the dependency is downloaded by the build system.

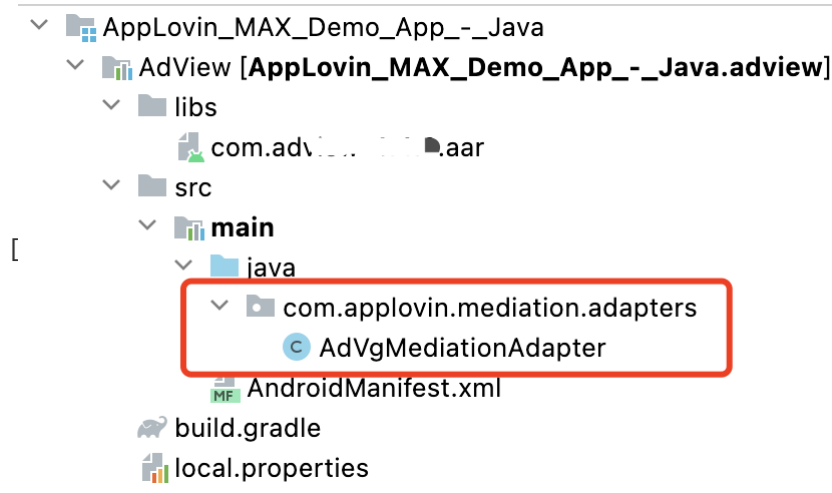
## Option 2: Adding the SDK Library to custom event Module

1. add adview's sdk lib adview module:



And then , we provide a sample reference codes for mediation adapter for adview's sdk invoking.

After that, you can find the following files in correspond directory.



note: the package name must be same as you set in AppLovin ads server setting ( see pic3-1 )

## Step 3: Use AppLovin sdk in Application framework

You only need use method same as AppLovin sdk handling the mediation adapter for AdView sdk.

here are simple example codes in app (more details pls see demo):

remember change [YOUR\_AD\_UNIT\_ID] to your app's unit id .

### 1. Banner

#### 1.1 start & load

```
adview = new MaxAdView( "YOUR_AD_UNIT_ID", this );
adview.setListener( this );
adview.setRevenueListener( this );
....
//with baner view inside
final ViewGroup rootView = (ViewGroup) findViewById( android.R.id.content );
rootView.addView( adview );
// Load the first ad.
adview.loadAd();
```

#### 1.2 banner callback

```
//region MAX Ad Listener
@Override
public void onAdLoaded(final MaxAd ad) { logCallback(); }
@Override
public void onAdLoadFailed(final String adUnitId, final MaxError maxError) {
    logCallback(); }
@Override
public void onAdHidden(final MaxAd ad) { logCallback(); }
@Override
public void onAdDisplayFailed(final MaxAd ad, final MaxError maxError) {
    logCallback(); }
@Override
public void onAdDisplayed(final MaxAd ad) { logCallback(); }
```

### 2. Mrec

## 2.1 start & load

```
adView = new MaxAdView( "YOUR_AD_UNIT_ID", MaxAdFormat.MREC, this );
adView.setListener( this );
adView.setRevenueListener( this );
//set view size
final int widthPx = AppLovinSdkUtils.dpToPx( this, 300 );
final int heightPx = AppLovinSdkUtils.dpToPx( this, 250 );
adView.setLayoutParams( new ConstraintLayout.LayoutParams( widthPx, heightPx )
);

// Set up constraints
final ConstraintLayout constraintLayout = findViewById(
R.id.main_constraint_layout );
constraintLayout.setId( ViewCompat.generateViewId() );
constraintLayout.addView( adView );
```

## 2.2 mrec callback

```
@Override
public void onAdLoaded(final MaxAd ad) { logCallback(); }
@Override
public void onAdLoadFailed(final String adUnitId, final MaxError maxError)
{ logCallback(); }
@Override
public void onAdHidden(final MaxAd ad) { logCallback(); }
@Override
public void onAdDisplayFailed(final MaxAd ad, final MaxError maxError) {
logCallback(); }
@Override
public void onAdDisplayed(final MaxAd ad) { logCallback(); }
@Override
public void onAdClicked(final MaxAd ad) { logCallback(); }
```

# 3. Interstitial

## 3.1 start & load

```
interstitialAd = new MaxInterstitialAd( "YOUR_AD_UNIT_ID", this );
interstitialAd.setListener( this );
interstitialAd.setRevenueListener( this );
// Load the first ad.
interstitialAd.loadAd();
```

## 3.2 show interstitial

after ad was loaded, show it anytime.

```
if ( interstitialAd.isReady() )
{
    interstitialAd.showAd();
}
```

## 3.3 interstitial callback

However, when ad failed occurred, you can load next ad or just let go, more logical process you can see demo app.

```
//region MAX Ad Listener
@Override
public void onAdLoaded(final MaxAd ad)
{
    logCallback();
}
@Override
public void onAdLoadFailed(final String adUnitId, final MaxError maxError)
{
    logCallback();
    ....
}
@Override
public void onAdDisplayFailed(final MaxAd ad, final MaxError maxError)
{
    logCallback();
    // Interstitial ad failed to display. We recommend loading the next ad.
    interstitialAd.loadAd();
}
@Override
public void onAdDisplayed(final MaxAd ad) { logCallback(); }
@Override
public void onAdClicked(final MaxAd ad) { logCallback(); }
@Override
public void onAdHidden(final MaxAd ad)
{
    logCallback();
    // Interstitial Ad is hidden. Pre-load the next ad
    interstitialAd.loadAd();
}
```

## 4. Reward

### 4.1 start & load

```
rewardedAd = MaxRewardedAd.getInstance( "YOUR_AD_UNIT_ID", this );
rewardedAd.setListener( this );
rewardedAd.setRevenueListener( this );
// Load the first ad.
rewardedAd.loadAd();
```

### 4.2 show reward

show ad if ready.

```
if ( rewardedAd.isReady() )
{
    rewardedAd.showAd();
}
```

### 4.3 reward callback



also, you can do your own process in callback, or refer to the demo codes or sdk suggestions.

```
//region MAX Ad Listener
@Override
public void onAdLoaded(final MaxAd ad)
{
    // Rewarded ad is ready to be shown. rewardedAd.isReady() will now
    return 'true'
    logCallback();
}
@Override
public void onAdLoadFailed(final String adUnitId, final MaxError maxError)
{
    logCallback();
    ...
}
@Override
public void onAdDisplayFailed(final MaxAd ad, final MaxError maxError)
{
    logCallback();
    // Rewarded ad failed to display. we recommend loading the next ad.
    rewardedAd.loadAd();
}
@Override
public void onAdDisplayed(final MaxAd ad) { logCallback(); }
@Override
public void onAdClicked(final MaxAd ad) { logCallback(); }
@Override
public void onAdHidden(final MaxAd ad)
{
    logCallback();
}
```

## 5. Native

### 5.1 start with different mode

you can use following 2 different modes to make native performance.

#### 5.1.1 Template mode

you just do nothing except create a NativeLoader, with this mode, MaxNativeAdView will be created by SDK, so you no need create it.

```
nativeAdLoader = new MaxNativeAdLoader( "YOUR_AD_UNIT_ID", this );
nativeAdLoader.setNativeAdListener( new MaxNativeAdListener()
{
    @Override
    public void onNativeAdLoaded(final MaxNativeAdView nativeAdView, final
MaxAd ad)
    {
        logAnonymousCallback();
        ....
        nativeAd = ad;
        // Add ad view to view.
        nativeAdLayout.removeAllViews();
    }
}
```

```

        nativeAdLayout.addView( nativeAdView );
    }
    ....
}

```

### 5.1.2 Manual mode

In this mode, you can do you self customize view with binder. with this mode, you should MaxNativeView will not be created unless you create it in app side.

```

MaxNativeAdViewBinder binder = new MaxNativeAdViewBinder.Builder(
    R.layout.native_custom_ad_view )
    .setTitleTextViewId( R.id.title_text_view )
    .setBodyTextViewId( R.id.body_text_view )
    .setAdvertiserTextViewId( R.id.advertiser_textview )
    .setIconImageViewId( R.id.icon_image_view )
    .setMediaContentViewGroupId( R.id.media_view_container )
    .setOptionsContentViewGroupId( R.id.options_view )
    .setCallToActionButtonId( R.id.cta_button )
    .build();
nativeAdView = new MaxNativeAdView( binder, this );

```

after create native view , you must add it in you layout in onNativeAdLoaded() listener.

```

nativeAdLoader = new MaxNativeAdLoader( "YOUR_AD_UNIT_ID", this ); //wilder 2022
nativeAdLoader.setNativeAdListener( new MaxNativeAdListener()
{
    @Override
    public void onNativeAdLoaded(final MaxNativeAdView nativeAdView, final MaxAd
ad)
    {
        logAnonymousCallback();
        ...
        nativeAd = ad;
        // Add ad view to view.
        nativeAdLayout.removeAllViews();
        nativeAdLayout.addView( nativeAdView );
    }
    ....
} );

```

### 5.1.3 Note

About native mode, in [Step.2] , if you choose "Template" or "Manual" mode, in app side you must choose the match mode with it. else the native ad may not be displayed.

### 5.2 load & show

when ad is loaded, it will bring back a MaxNativeAdView , you can use it to layout .

```

nativeAdLoader.loadAd();

```

after load ad when ready, then ad will be shown on .

### 5.3 native callback

native callback as following :

```

@Override
public void onNativeAdLoaded(final MaxNativeAdView nativeAdView, final MaxAd ad)
{
    logAnonymousCallback();
    // Save ad for cleanup.
    nativeAd = ad;
    // Add ad view to view.
    nativeAdLayout.removeAllViews();
    nativeAdLayout.addView( nativeAdView );
    ...
}
@Override
public void onNativeAdLoadFailed(final String adUnitId, final MaxError error)
{
    logAnonymousCallback();
}
@Override
public void onNativeAdClicked(final MaxAd ad)
{
    logAnonymousCallback();
}

```

## Step 4: AdView Mediation adapter module

And you can see the mediation adapter codes for AdView in AdVgMediationAdapter.java. this is a main entry for MAX mediation event . and you will find 4 types ads entry class :

```

public class AdVgMediationAdapter extends MediationAdapterBase implements
    MaxSignalProvider, MaxInterstitialAdapter, MaxRewardedAdapter,
    MaxAdViewAdapter
{
    ....
}

```

also the following different ad types of override functions:

banner + mrec :

```

@Override
public void loadAdViewAd(final MaxAdapterResponseParameters parameters,
                        final MaxAdFormat adFormat,
                        final Activity activity,
                        final MaxAdViewAdapterListener listener)

```

Interstitial:

```

@Override
public void loadInterstitialAd(final MaxAdapterResponseParameters parameters,
                             final Activity activity,
                             final MaxInterstitialAdapterListener listener)

```

```
@Override
public void showInterstitialAd(final MaxAdapterResponseParameters parameters,
                             final Activity activity,
                             final MaxInterstitialAdapterListener listener)
```

Reward :

```
@Override
public void loadRewardedAd(final MaxAdapterResponseParameters parameters,
                          final Activity activity,
                          final MaxRewardedAdapterListener listener)
```

```
@Override
public void showRewardedAd(final MaxAdapterResponseParameters parameters,
                          final Activity activity,
                          final MaxRewardedAdapterListener listener)
```

Native :

```
@Override
public void loadNativeAd(final MaxAdapterResponseParameters parameters,
                        final Activity activity,
                        final MaxNativeAdAdapterListener listener)
```

in this class, adview's implementation has been done already, but if you want to changed it for your customize needed. you can do your changes with AdView's sdk menu to handle it .

## Step 5: GDPR

As a publisher, you should integrate a Consent Management Platform (CMP) and request for vendor and purpose consents as outlined in IAB Europe's Mobile In-App CMP API v1.0: Transparency & Consent Framework. You can find a reference implementation of a web-based CMP and the corresponding native wrappers here in the IAB's GDPR Transparency and Consent Framework.

if you already got IABConsent\_ConsentString, you just use setGDPR2() method to provide gdpr subjects in AdVgMediationAdapter.java, and you will see the following codes in multi places:

prototype:

```
public void setGDPR2(int gdprApplies, String consentString);
```

for example:

```
adviewBIDView.setGDPR2(1,Gdpr); //Gdpr means consent string
```

## ProGuard setting

you should add the following rules in proguard-project.txt , or you may not run sdk pass through .

```
-keep public class android.webkit.JavascriptInterface { *; }  
-dontwarn com.iab.omid.library.adview.**  
-keep public class com.iab.omid.library.adview.**.* { *; }  
  
-dontwarn com.advg.**  
-keep public class com.advg.**.* { *; }
```

and also, adview's adapter codes also not be obfuscated, if your adapter package path is :

```
com.applovin.mediation.adapters.AdVgMediationAdapter
```

so you should declare the following definition in proguard-project.txt :

```
-keep public class com.applovin.mediation.adapters.AdVgMediationAdapter { *; }
```

## You're all done!

Your Max mediation SDK should start showing AdView ads immediately.

Otherwise please contact your AM or [partner@adview.com](mailto:partner@adview.com).