MAX Adapter Installation

Android MAX Adapter Installation

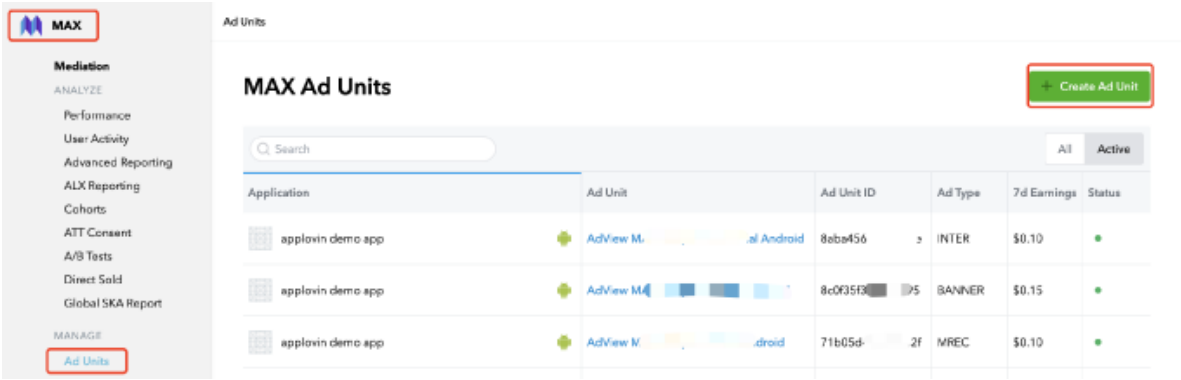
Adding AdView to MAX

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

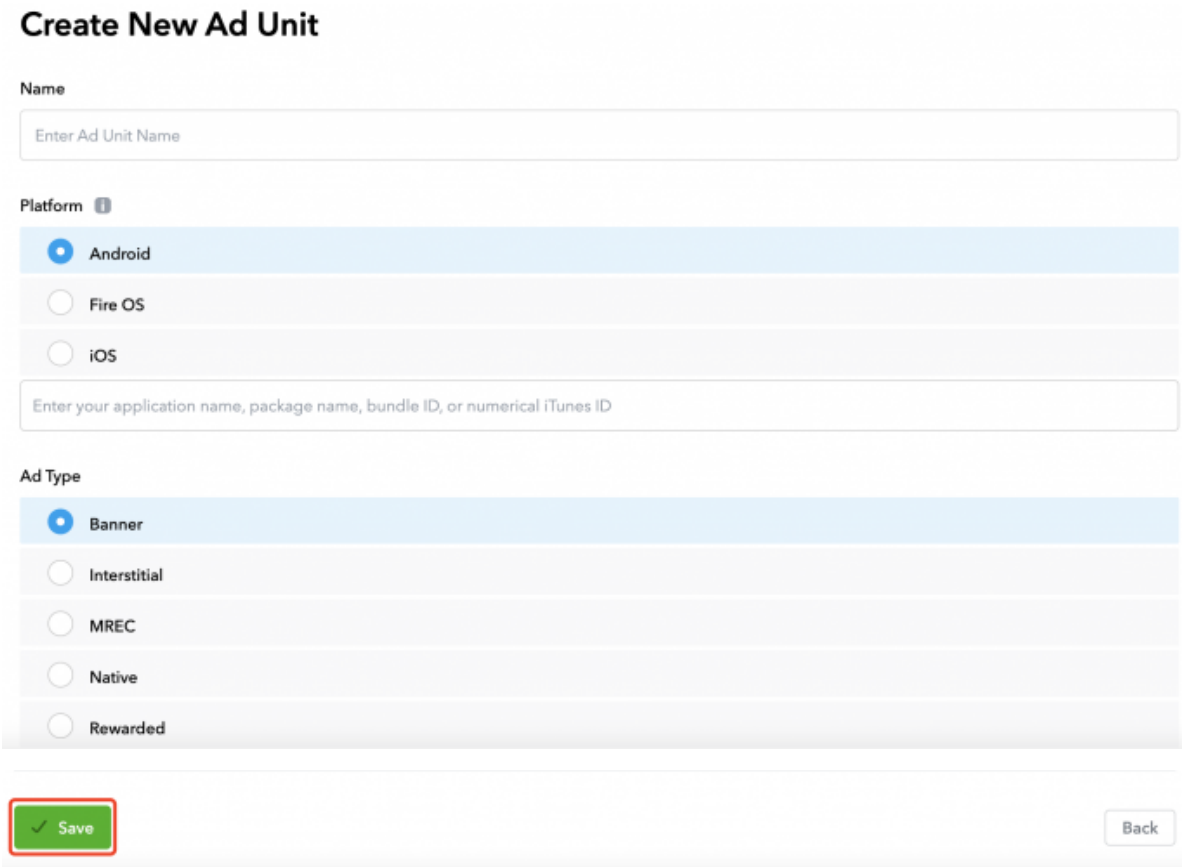
As explained above, you must have an AppLovin account and select the MAX option in order to integrate AdView.

Step 2: Create a new AD unit ID

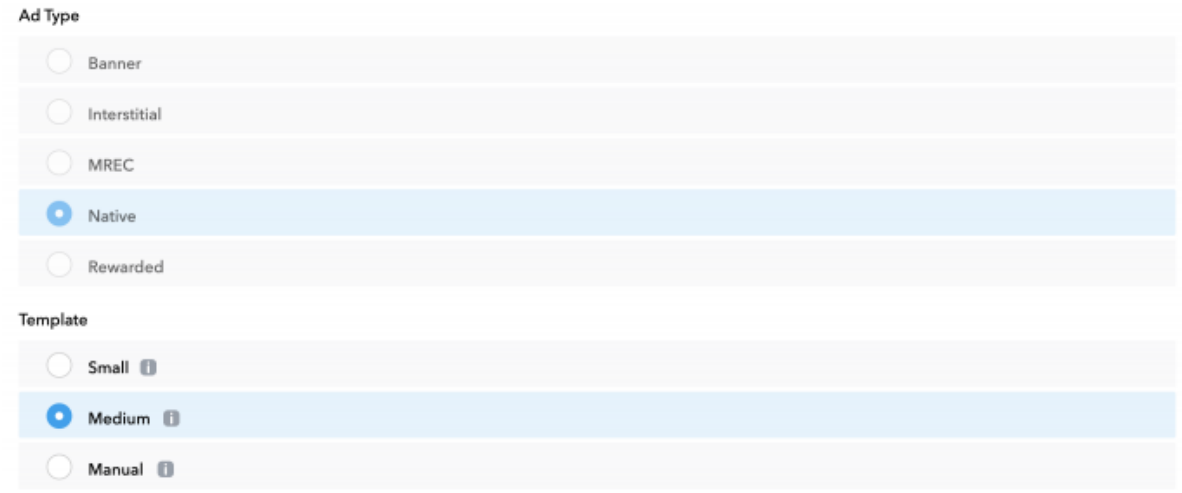
Under the MAX subtab, select Ad Units, and then click "Create Ad Unit"



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"



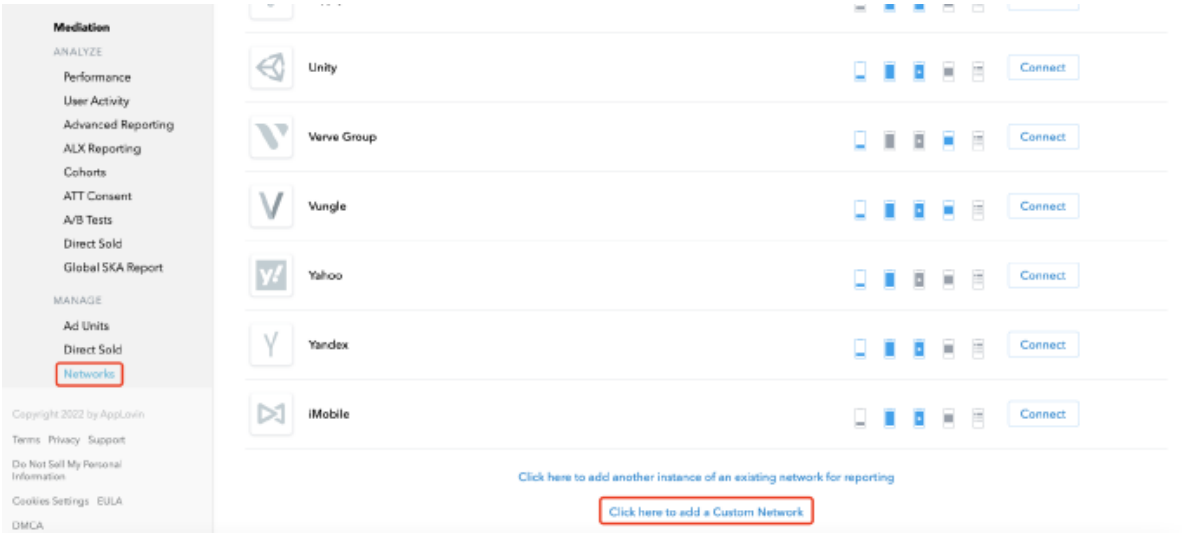
Native Ad Unit



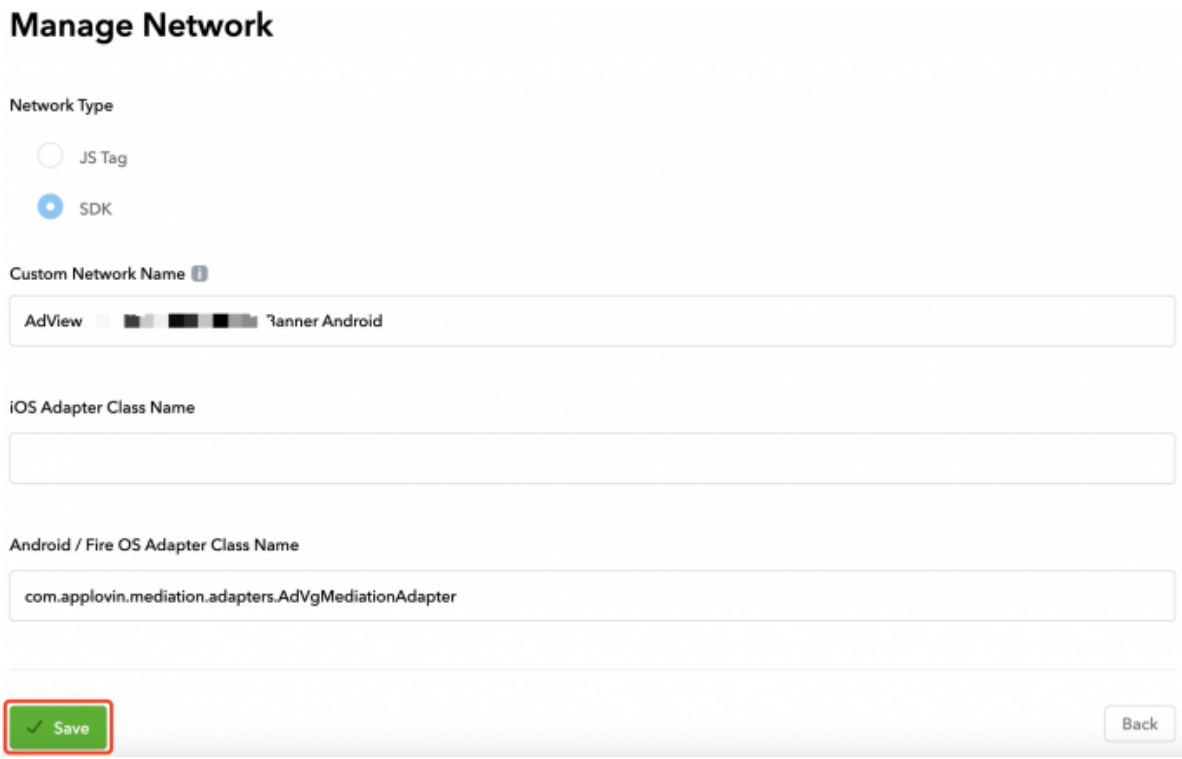
for more native detail mode, pls refer to following 5.0 .

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”

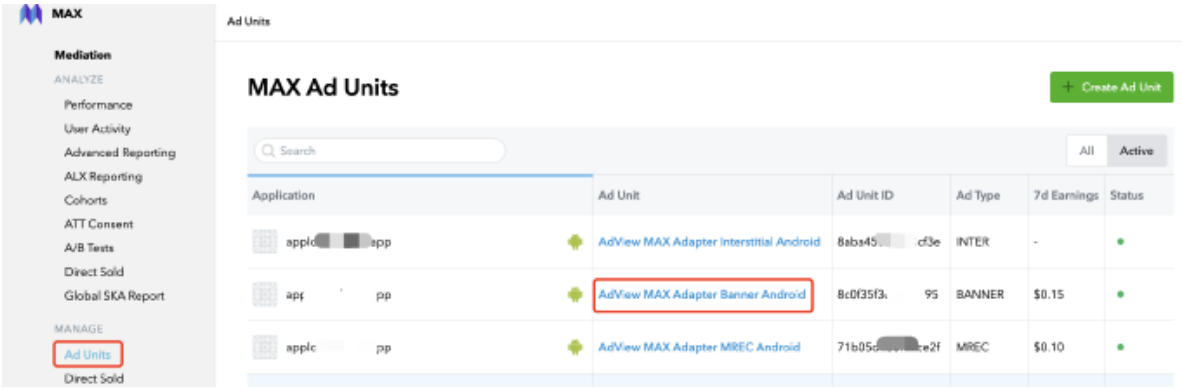


(pic 3-1)

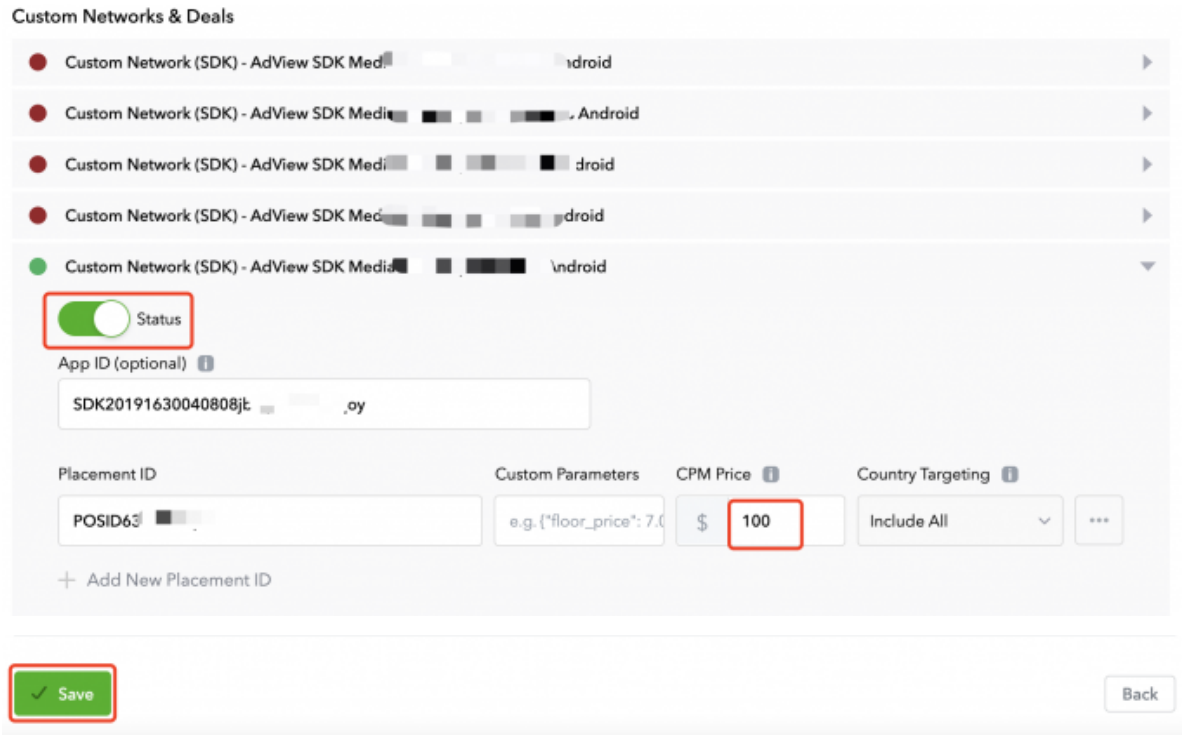
Android / Fire OS Adapter Class Name

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

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



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



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:

- AdView SDK
- AdViewSDK_MAXAdapter_demo

Otherwise please contact your AM or partner@adview.com.

Step 2: Add the AdView SDK into your Project

You can have 2 options to handle adview sdk workful.

Option 1: Pulling the latest custom adapter 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 app module's `build.gradle`.

```
api 'com.adview:android-max-adapter:4.x.x'
```

Option 2: Adding the custom adapter Library to app module

add adview's sdk adapter lib in app path.

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

Sync your gradle project to ensure that the dependency is handled by the build system.

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 interstitilal

after ad was loaded, show it anytime.

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

3.3 interstitilal 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 Menual mode

in this mode, you can do you self customize view with binder. with this mode, you should MaxNatvieView 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 "Menual" 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();
}
```

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 { *; }
```

Enable debug logout

For max sdk, you can use the following codes to open debug info , more detail pls refer to *GlobalApplication.java* in demo app.

```
AppLovinSdkSettings settings = AppLovinSdk.getInstance(this).getSettings();
settings.setVerboseLogging(true); //debug logout
```

You’re all done!

Your Max mediation SDK should start showing AdView ads immediately.

Otherwise please contact your AM or partner@adview.com.