

AdMob Adapter Installation

Android Admob Adapter Installation

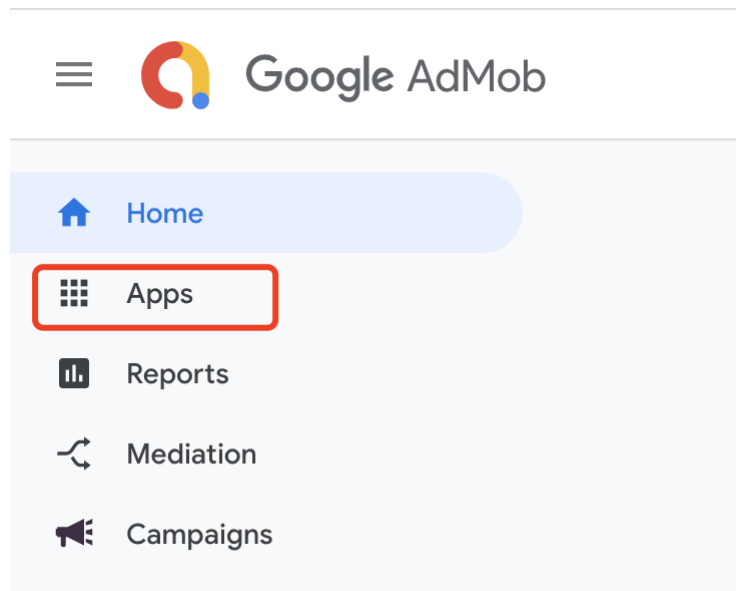
Adding AdView to AdMob

Step 1: Login to your AdMob account and configure AdView

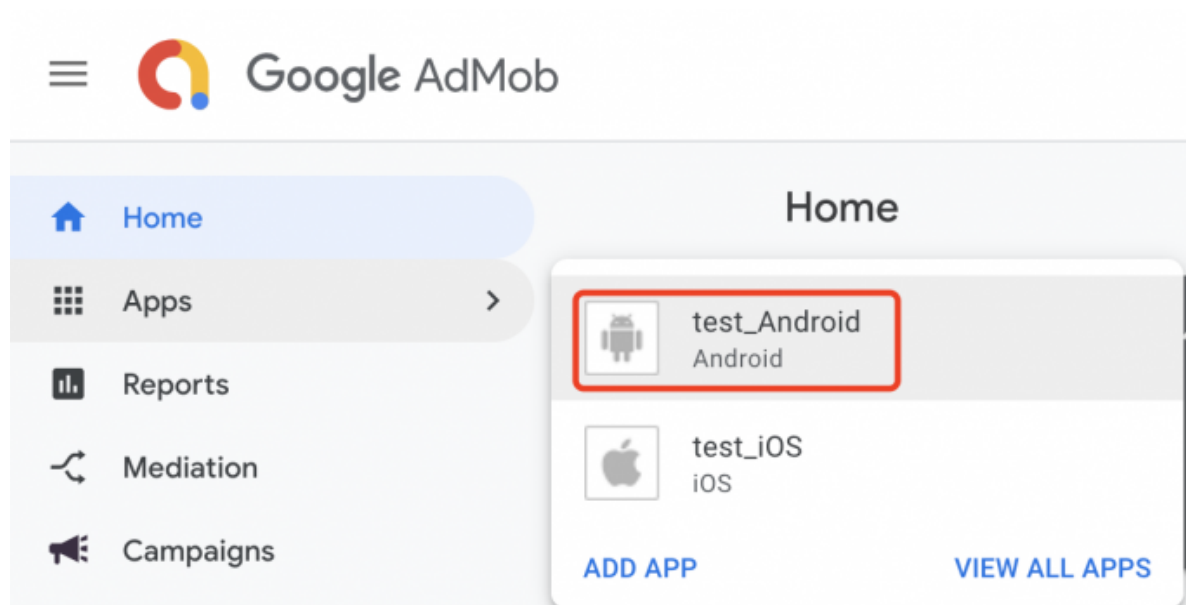
As explained above, you must have a AdMob account, and an associated app within this account, in order to integrate AdView.

Step 2: Select your app and click to monetize it

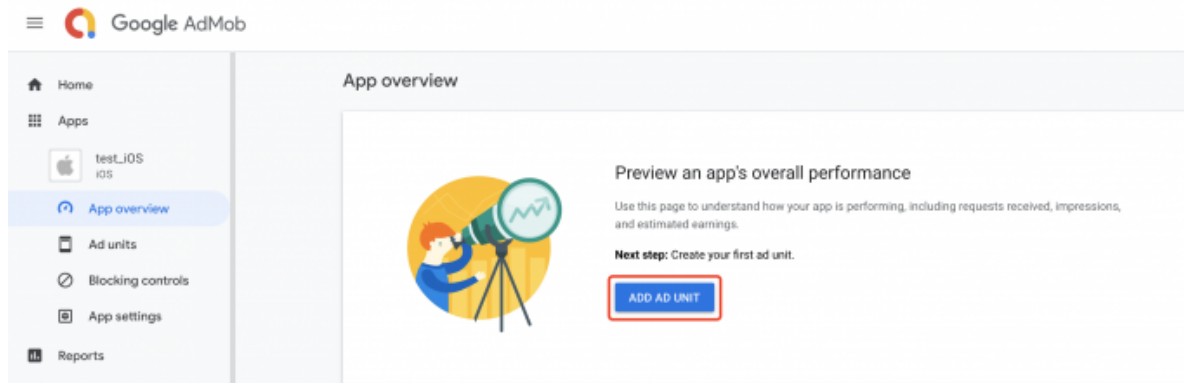
Within your AdMob account, press the 'Apps' tab



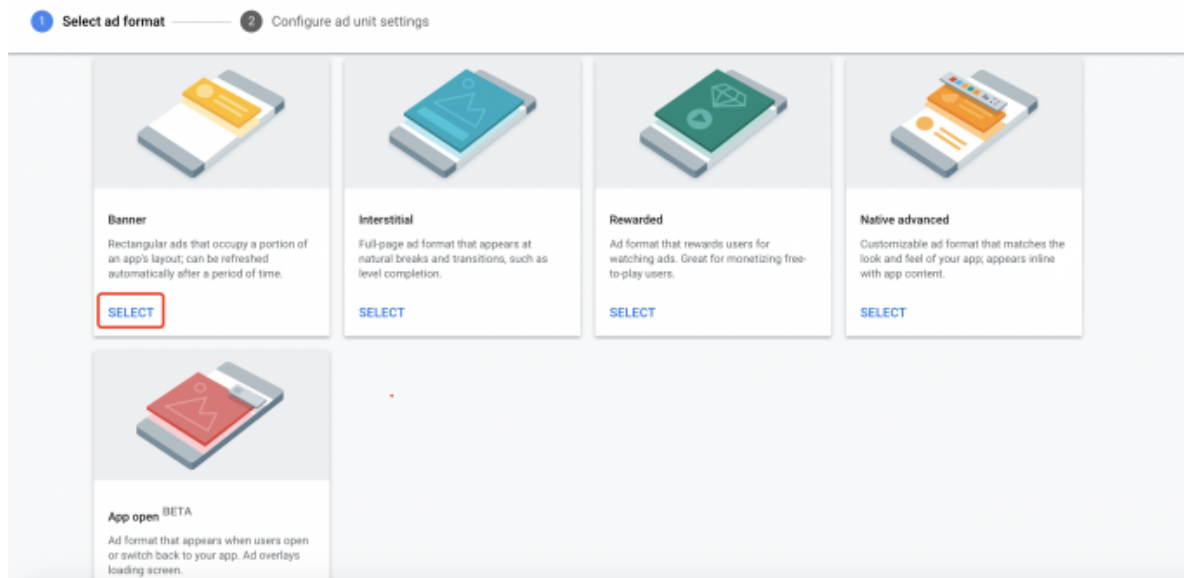
AdView must be integrated as a custom event. If your app does not have a ad unit yet, you should create a new ad unit by clicking on your app's name:



After clicking on your app's name, you will need to click the "New ad unit" button



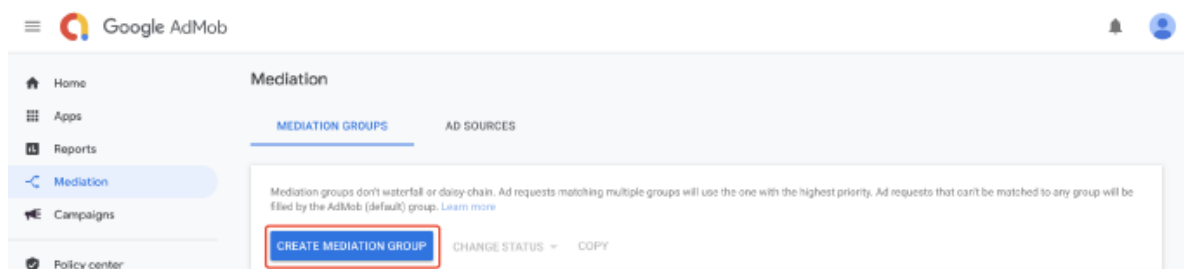
Provide a name for your new ad unit. After making the device type selection, you should choose a format that you need to installation:



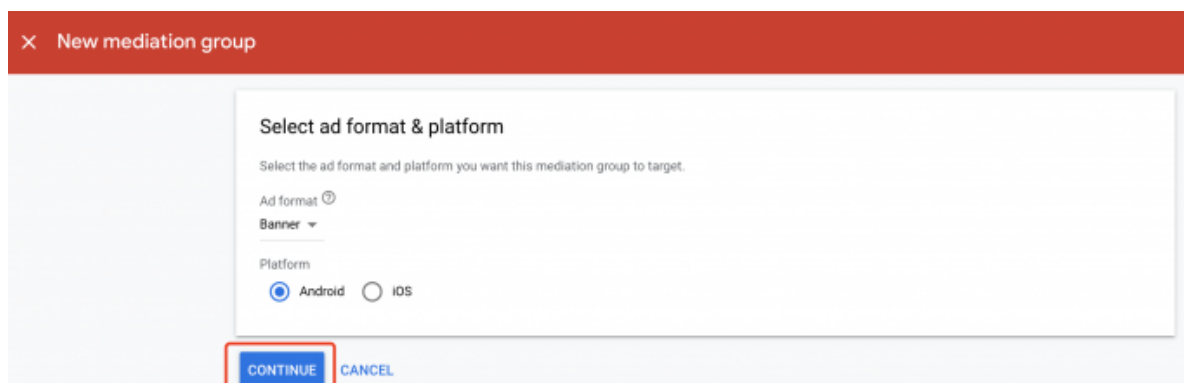
note: create MREC AD unit, you need select "Banner". The configuration is the same as Banner.

Step 3: Create mediation group for your ad placement

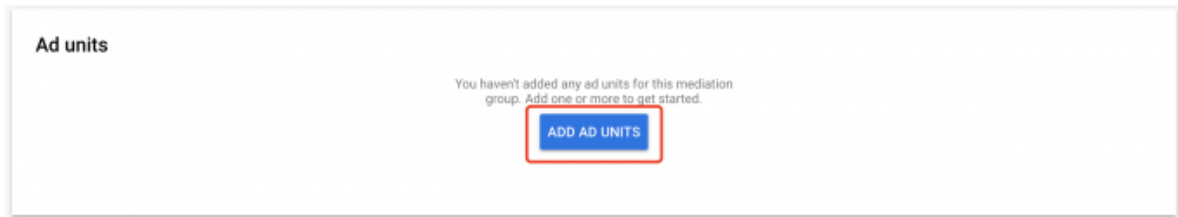
Within your AdMob account, press the 'Mediation' tab, and click "Create Mediation group".



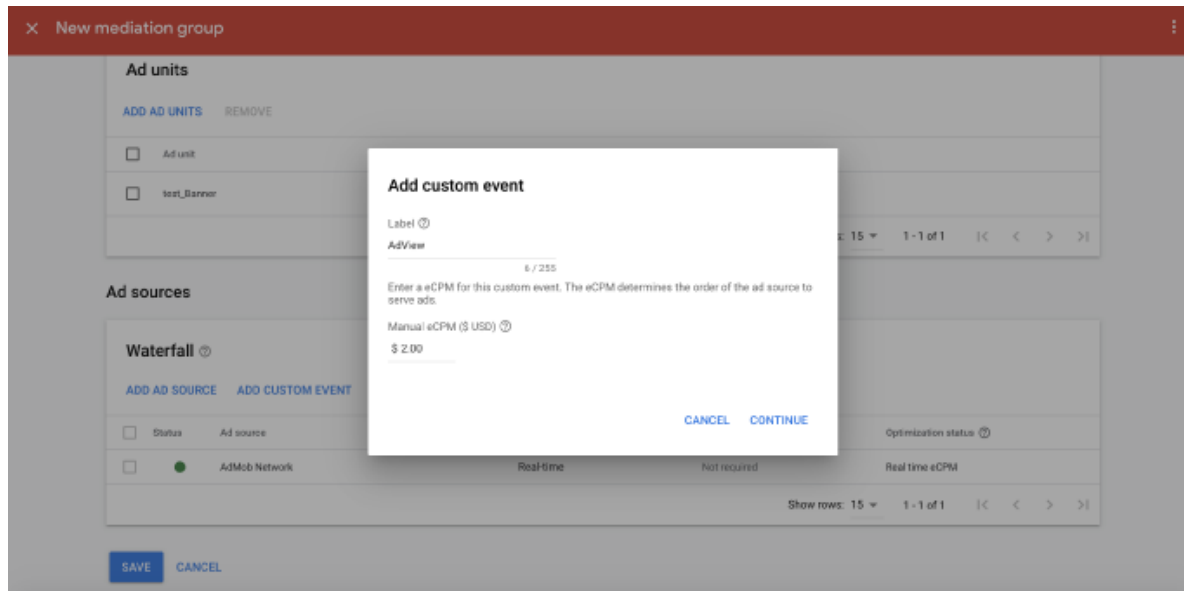
Then click "CONTINUE"



At this time, please provide a Mediation group name for your new Mediation group, then you need click"ADD AD UNITS"



AdView must be integrated as a custom event,so please click Add Custom Event.When you finish write Label and eCPM click continue



Add the following class to the "Class name" section for your ad unit

Map ad units: test_banner_maven

Map your ad units to this custom event. [?](#)

AdMob	test_banner_maven
 test_Android Android	Class Name <code>com.google.ads.mediation.adview.customevent.AdVGCustomEven</code>
test_banner  ca-app-pub-720948430...	Parameter (optional) SDP...

CANCEL DONE

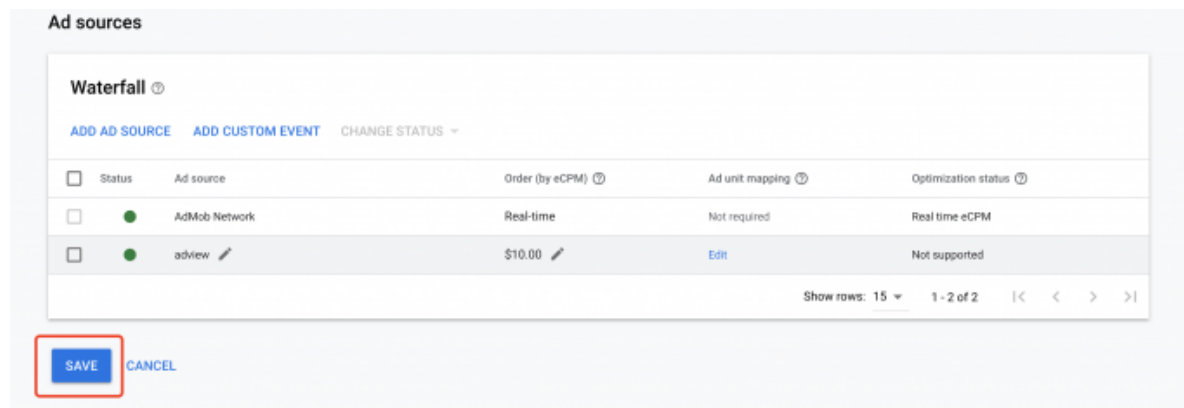
Class Name

```
com.google.ads.mediation.adview.customevent.AdVGCustomEvent
```

It is mandatory to add your AdView appld (SDK-KEY) and placement ID (posId) to the "Parameter(optional)" section.

```
AdView appId (SDK-KEY);AdView placement ID (posId)
```

Now click 'DONE' and 'SAVE'



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

AdMob Adapter Installation

Android AdMob Adapter Installation

Step 1: Get the file with the adapter

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

- AdViewSDK_AdMobAdapter
- AdViewSDK_AdMob_Mediation_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 workfl.

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-admob-adapter:4.4.0'
```

Option 2: Adding the custom adapter Library to app module

add advview's sdk adapter lib in app path.

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

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

Step 3: Use AdMob sdk in Application framework

You only need use method same as AdsMob sdk handling the customevent for AdView sdk.

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

1.Banner

You need set adunit id and set adlistener to handle the ad events.

Note: adsize is *AdSize.BANNER*.

```
Adview advview = new Adview(view.getContext());  
advview.setAdSize(AdSize.BANNER);  
advview.setAdUnitId(getBannerAdUnitId());  
advview.setAdListener(new AdListener() {  
    @Override  
    public void onAdFailedToLoad(@NonNull LoadAdError loadAdError) {  
        ...  
    }  
});  
advview.loadAd(new AdRequest.Builder().build());
```

2.Mrec

Mrec is a special banner, so just need set the advview size with banner usage.

Note: adsize is *AdSize.MEDIUM_RECTANGLE*.

```
Adview admrecview = new Adview(view.getContext());  
admrecview.setAdSize(AdSize.MEDIUM_RECTANGLE);  
admrecview.setAdUnitId(getMrecAdUnitId()); //use mrec id  
...  
advview.loadAd(new AdRequest.Builder().build());
```

3.Interstitial

For interstitial ad, need load and show ad 2 steps.

```
private InterstitialAd interstitial;
```

3.1 load ad

demo codes, load ad and set InterstitialAdLoadCallback() .

```

InterstitialAd.load(MainActivity.this, getInterstitialAdUnitId(),
    new AdRequest.Builder().build(), new InterstitialAdLoadCallback() {
        @Override
        public void onAdLoaded(@NonNull InterstitialAd interstitialAd) {
            interstitial = interstitialAd;
            interstitial.setFullScreenContentCallback(
                new FullScreenContentCallback() {
                    ...
                });
            ...
        }
        @Override
        public void onAdFailedToLoad(@NonNull LoadAdError loadAdError) {
            interstitial = null;
            ...
        }
    });

```

3.2 show ad

after ad is loaded, you can show it anytime.

```

if (interstitial != null) {
    interstitial.show(MainActivity.this);
}

```

4.Reward

Reward ads also need 2 steps: load and show.

```

private RewardedAd rewardedAd;

```

4.1 load reward

load ad and set RewardedAdLoadCallback() to handle events.

```

RewardedAd.load(MainActivity.this, getRewardedAdUnitId(), new
AdRequest.Builder().build(),
    new RewardedAdLoadCallback() {
        @Override
        public void onAdLoaded(@NonNull RewardedAd ad) {
            rewardedAd = ad;
            rewardedAd.setFullScreenContentCallback(new
FullScreenContentCallback() {
                @Override
                public void onAdFailedToShowFullScreenContent(@NonNull
AdError error) {
                    ...
                }
                @Override
                public void onAdDismissedFullScreenContent() {
                    ...
                }
            });
        }
        @Override
        public void onAdFailedToLoad(@NonNull LoadAdError loadAdError) {

```

```

        rewardedAd = null;
        ...
    }
});

```

4.2 show reward

Piece of demo codes.

```

if (rewardedAd != null) {
    rewardedAd.show(MainActivity.this, new OnUserEarnedRewardListener() {
        @Override
        public void onUserEarnedReward(@NonNull RewardItem rewardItem) {
            ...
        }
    });
}

```

5.Native

Native ad, need a NativeAdView to hold NativeAd. after ad has been loaded. native ad view should be added in Container.

5.1 loader and listener

```

// The ad loader.
private AdLoader adLoader;

```

demo codes, here in onNativeAdLoaded(), we put the NativeAdView into nativeContainer to show it.

```

adLoader = new AdLoader.Builder(view.getContext(), getNativeAdUnitId())

    .forNativeAd(new NativeAd.OnNativeAdLoadedListener() {
        @Override
        public void onNativeAdLoaded(@NonNull NativeAd nativeAd) {
            FrameLayout nativeContainer =
findViewById(R.id.native_container);
            NativeAdView adView = (NativeAdView) getLayoutInflater()
                .inflate(R.layout.native_ad, null);
            populateNativeAdView(nativeAd, adView);
            nativeContainer.removeAllViews();
            nativeContainer.addView(adView);
        }
    })
    .withAdListener(new AdListener() {
        @Override
        public void onAdFailedToLoad(@NonNull LoadAdError error) {
            ...
        }
    })
    .build();

```

5.2 load native ad

```
adLoader.loadAd(new AdRequest.Builder().build());
```

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, adview 's custom adapter class's package path is :

```
package com.google.ads.mediation.adview.customevent.AdVCustomEvent;
```

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

```
-keep public class com.google.ads.mediation.adview.customevent { *; }
```

Enable debug tracking

Starting September 2022, google will start limiting ad serving to apps that haven't been reviewed and approved, so begin the review process, pls Check your 'Apps to confirm' page. check 'Apps to confirm' for apps that require additional setup. If there aren't any apps on your 'Apps to confirm' page, no action is needed.

All Apps

Apps **Apps to confirm** 1 app-ads.txt

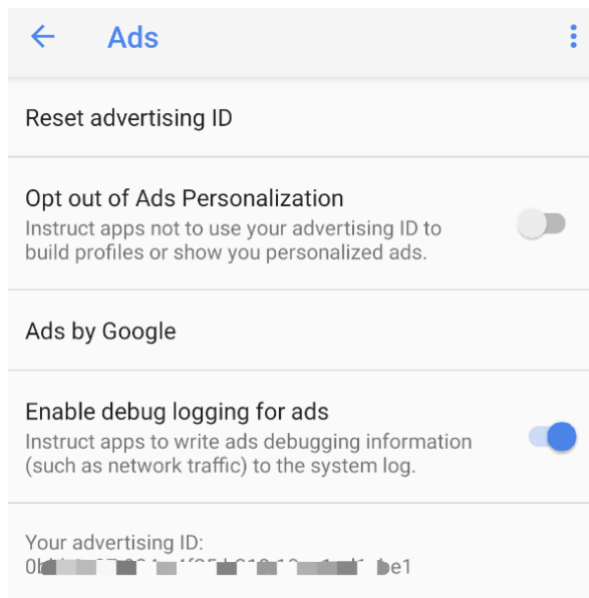
These apps are showing ads, but haven't been added to your AdMob apps list yet. To avoid ad limits, finish setting up all the apps you want to monetize. [Learn more about apps in this list](#)

Filter by **Requires setup** Ads blocked

App	Package name or store ID	Status	Ad requests - last 7 days	Action
 -- Android	com.google.ads.mediation.sample...	Setup incomplete	29	Finish setup Not my app

Show rows: 15 1 - 1 of 1 |< < > >|

Also, if you don't want confirm app yet, you can test you app adapter with debug mode. To enable network tracing, you will need to enable developer options for your device. Launch the Google Settings app and select **Google > Ads > Enable debug** logging for ads. A more detailed guide is available for both AdMob and Google Ad Manager publishers.



You're all done!

Your AdMob SDK should start showing AdView ads immediately.

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