Introduction

This guide is intended for developers who are integrating the AdView Android SDK. After you have integrated the SDK, you must choose the relevant ad format and follow the steps for implementing that format.

Prerequisites

- Android 4.1 (API Level 16)+
- Google Play Services 11.4.0+

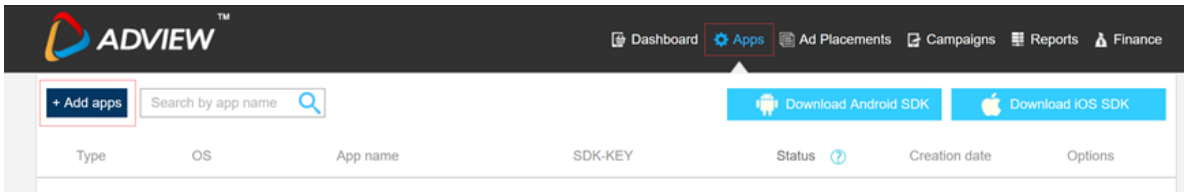
Step 1: Sign up and acquire app id

Before integration, you need to sign up on **AdView Mobile Marketplace** first and acquire a specific AdView App id **SDK-KEY** for each app you would like to integrate with AdView SDK.

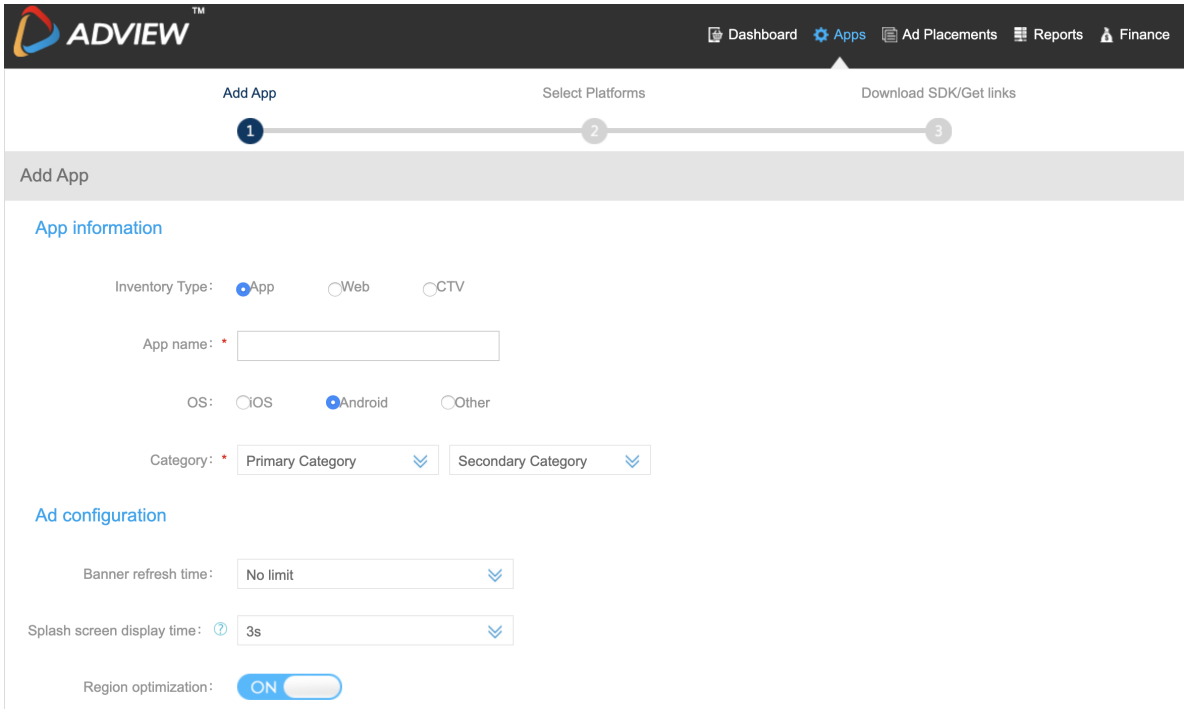
1. Sign up your publisher account at AdView Official Website:

   https://www.adview.com/register

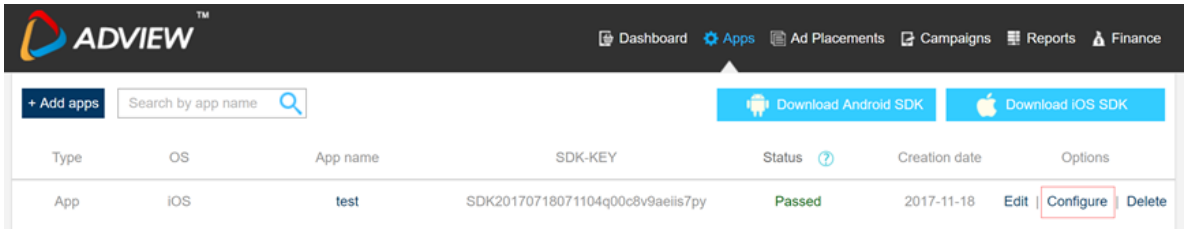
2. Sign in with your user name and password, AdView Publisher Dashboard opens
3. On the above menu bar, click **Apps**
4. On the left-up corner, click **+Add apps**



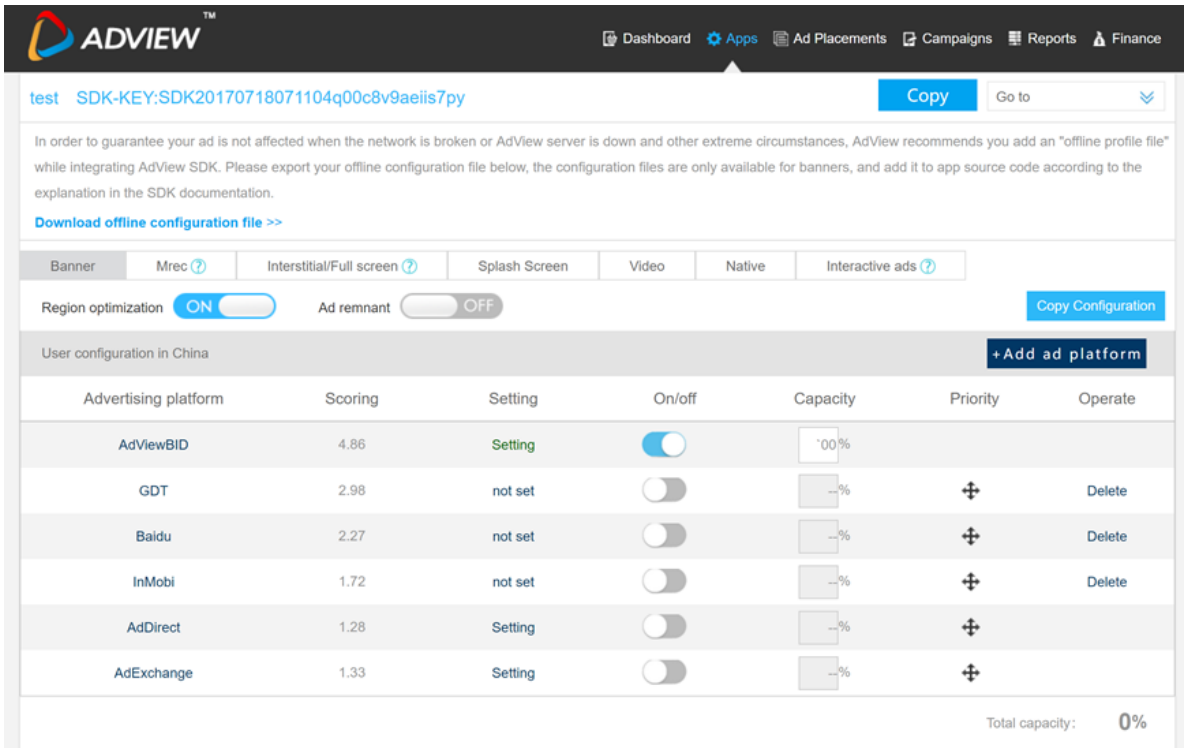
5. Fill in the app information step by step, please select the correct ad format, otherwise you might not get any ad fill. After filling all the necessary app information, please submit the app information for review. The review process generally takes one business day, please contact your account manager if any questions.



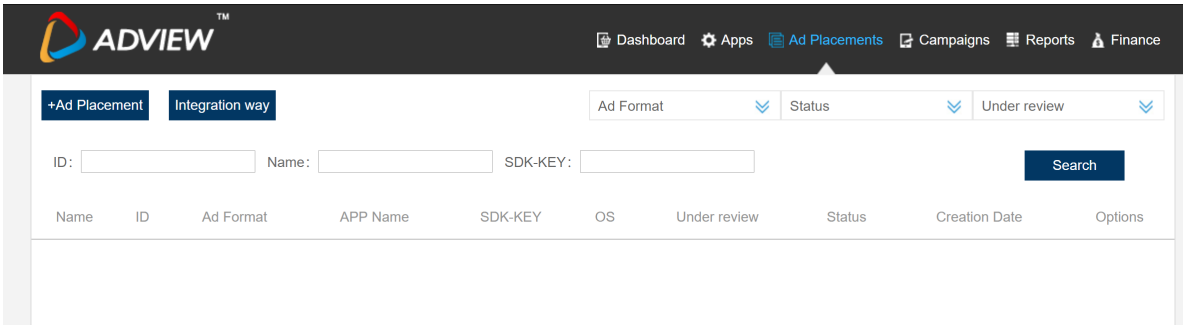
6. After the app has passed the review, the app **Status** will become **Passed** as below, please click **Configure** to config the ad network mediation configuration.



7. If you are using AdView's ad network mediation feature, please click **Add ad platform** to add the ad networks you would like to support. If you are already using other ad mediation solutions, please simply turn the Advertising platform **AdViewBID** on and set the capacity to 100%. You can also find the specific AdView app id **SDK-KEY** here, which you will need for the SDK integration.



8. If your app's ad format is Video or Native, you need to create **ad placement** for each ad place in your app. Please select **Ad Placement** on the above menu bar and click **+Ad Placement**, then add the necessary ad placement information accordingly.



Step 2: Integration

Manual SDK Download

1. Download the latest Version SDK here:

   https://www.adview.com/productAndTech/sdk#download-sdk

   Put aar file into your project's app/libs , and add the dependency in *build.gradle*.

```
dependencies {
    //here import sdk jar in <libs> folder
    api fileTree(include: ['*.aar'], dir: 'libs')

    api 'com.android.support:support-v4:26.0.0'
    api 'com.android.support:appcompat-v7:26.1.0'
    api 'com.google.android.gms:play-services-ads:15.+'
}
```

Update Your Android Manifest

Update your AndroidManifest.xml in order to complete the SDK integration. Add the following permissions and activity declarations according to the bundle you are integrating.

1. Declare the following permissions in your ：

```
    <!-- AdView SDK mandatory or important permissions -->
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

And above Android 6.0 , the dynamic permission should be applied in app start codes, more details of such codes pls refer to Demo app, see SDK package.

```
    if (Build.VERSION.SDK_INT >= 23) {
        permissions = new String[]{
                Manifest.permission.INTERNET

        };
        for (String permission : permissions) {
            if (ContextCompat.checkSelfPermission(this, permission) !=
                    PackageManager.PERMISSION_GRANTED) {
```

```
                    //no permission, ask for permissions
                    ActivityCompat.requestPermissions(this, permissions, 1);
                    return;
                }
            }
            //all granted
            onPermissionGranted();
        }else {
            //all granted
            onPermissionGranted();
        }
    }
```

2. Declare the following activities in your :

AdViewVideoActivity is used by Rewarded & Interstitial Video , and *AdViewLandingPage* is used for landingpage of Advisement click up, more details see section **Video**. also need add AdActivity for interstitial Ads.

```
    <!-- Must declare it for Adivew SDK -->
    <activity
        android:name="com.kuaiyou.video.AdViewVideoActivity"
        android:configChanges="keyboardHidden|orientation|screenSize"
        android:hardwareAccelerated="true" >
    </activity>
    <activity android:name="com.kuaiyou.utils.AdActivity"
        android:theme="@android:style/Theme.Translucent.NoTitleBar" />
    <activity android:name="com.kuaiyou.utils.AdViewLandingPage" />
```

3. Add this tag to your *build.gradle* to use Google Play Services & Huawei OAID:

   The play service's version may different with your app's usage and support, and although it can run ok.

   3.1 add repositories:

```
allprojects {
    repositories {
        jcenter()
        google()    //google support
        maven { url 'http://developer.huawei.com/repo/' } //huawei hms
    }
}
```

   3.2 add gradle dependency:

```
dependencies {
    ...
    api 'com.huawei.hms:hms-ads-identifier:3.4.+'      //huawei hms
    api 'com.google.android.gms:play-services-ads:15.+'
}
```

Add a Network Security Configuration File

Android 9.0 (API 28) blocks cleartext (non-HTTPS) traffic by default, which can prevent ads from serving correctly. To mitigate that, publishers whose apps run on Android 9.0 or above should ensure to add a network security config file. Doing so whitelists cleartext traffic and allows non-HTTPS ads to serve.

1. In your AndroidManifest.xml file, add *android:networkSecurityConfig* tag in ：

```
<application
      android:allowBackup="true"
      android:icon="@drawable/ic_lancher"
      ...
      android:networkSecurityConfig="@xml/network_security_config"
</application>
```

2. In your *network_security_config.xml* file, add a base-config that sets *cleartextTrafficPermitted* to true, and put the xml file into *res/xml* folder, xml file contents as following :

```
<?xml version="1.0" encoding="utf-8"?>
<network-security-config>
    <base-config cleartextTrafficPermitted="true" />
</network-security-config>
```

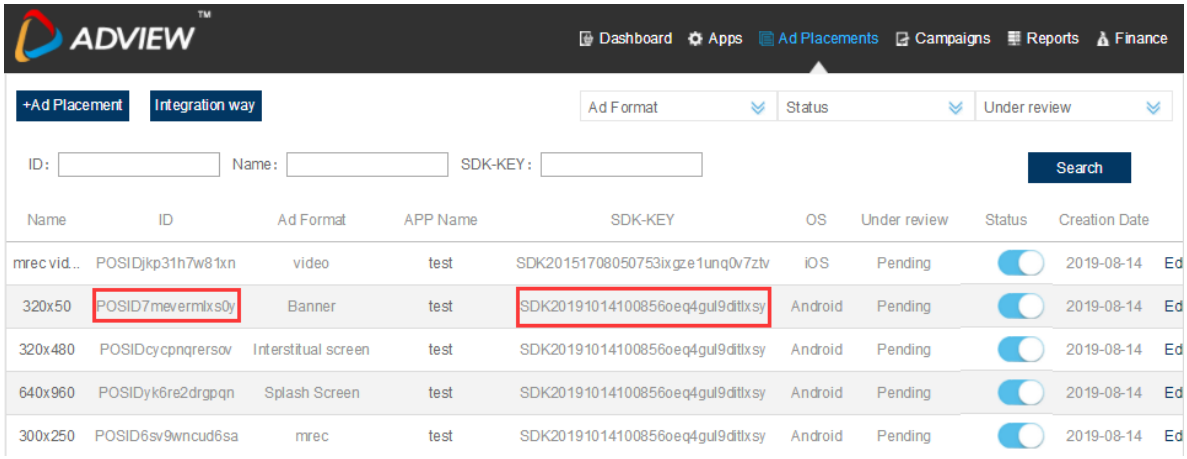
Step 3: Configure Ad Units in Your App

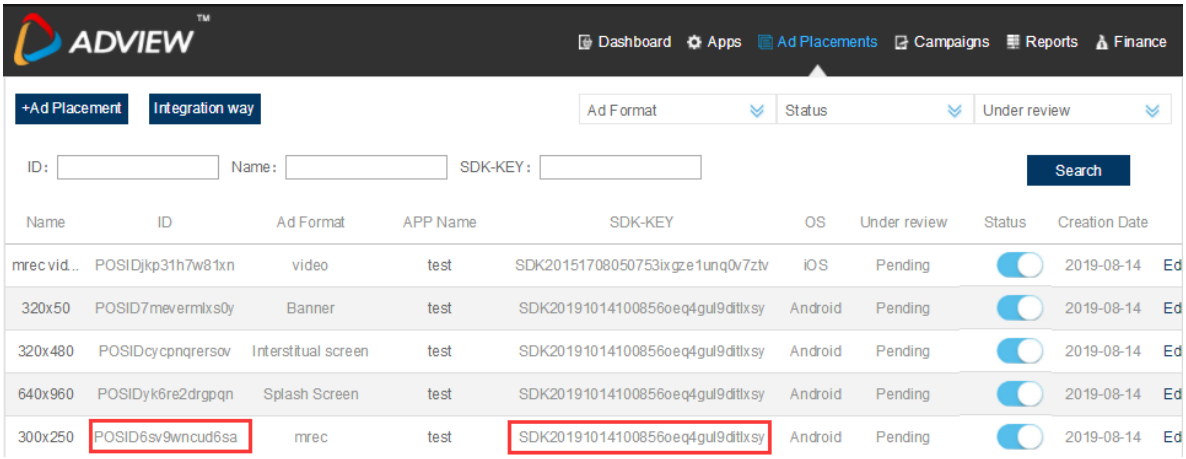
Banner

Banner ads usually appear at the top or bottom of your app's screen. Adding one to your app takes just a few lines of code.

Create an APP in the Publisher UI for instructions. You will need that Ad Placement ID and SDK-KEY.

Usually when you add an app, you have the option of automatically ad banner or mrec ad placement.





1. Define a Slot for Your Banner Ad in Your XML Layout

The AdView SDK provides a custom View subclass *AdViewBannerManager*, which handles requesting and loading ads. We provide the following methods for you to set up this subclass:

prototype:

```
public AdViewBannerManager(Context context, String key, String vPosID, int
adSize,
                           boolean canClosed)
```

example:

```
adViewBIDView = new AdViewBannerManager( AdBannerActivity.this,
                    appID,
                    posID,
                    adSize,
                    true);
```

income SDK-KEY and Ad Placement ID, only MREC mode banner need Placement ID.

```
    public static final String appID = "SDK201916270406508oj5ykxwe3xrjyx";
//test id
    public static final String posID = "POSIDxaal0vy6fstf";
```

Choose the ad size you need, we recommend 728x90 for tablet traffic and 320x50 for mobile traffic. The default size of MREC is 300x250, more define as following, all definitions are in *AdViewBannerManager*. pls note the corresponding value. In addition, MREC currently supports the display of video advertisements, which will be described in detail below.

```
    public static int BANNER_AUTO_FILL = 0; //auto fit your layout width, height
is 50
    public static int BANNER_MREC = 1;      //300x250
    public static int BANNER_480X75 = 2;
    public static int BANNER_728X90 = 3;
    public static int BANNER_SMART = 5;     //320x50
```

Select whether you need to close the button and it can be shown or not .

```
    adViewBIDView.setShowCloseBtn(true);
```

Set refresh time, normal is 15 sec to change to next ad. if you want to show it always until you can change to next one by yourself, you can also put sec to -1.

```
    adViewBIDView.setRefreshTime(15); //-1 means not auto refresh
```

2. Load an Ad into the Banner Slot

After create a object of *AdViewBannerManager*, it will start ad request and to get advertisement, and now , you should add the banner view in your view with *AdViewBannerManager.getAdViewLayout(),* such as following (layout means your ad holder view, it is banner view's parent view):

```
if (null != layout)
    layout.addView(adViewBIDView.getAdViewLayout());
```

3. Using the Delegate

AdViewBannerManager provides a listener interface *setOnAdViewListener*(),

```
adViewBIDView.setOnAdViewListener(this);
```

and the prototype is following:

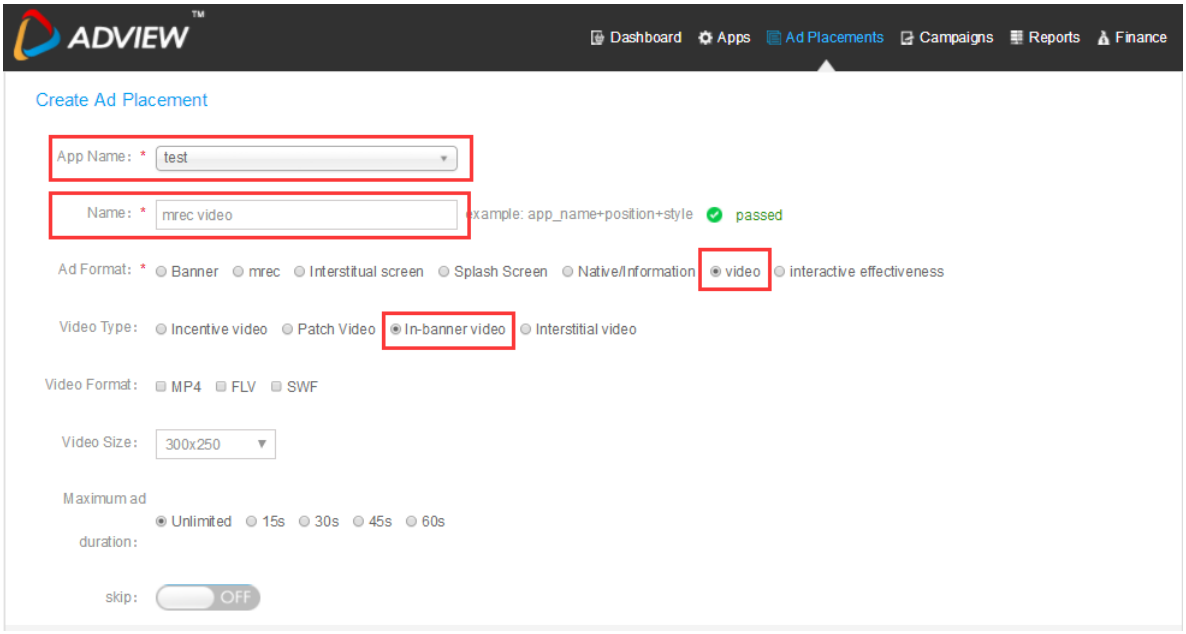
```
public void setOnAdViewListener(AdViewBannerListener bannerListener)
```

which includes a variety of optional callbacks you can use to be notified of events; for example, when an ad has successfully loaded, or when the ad is about to present or dismiss a modal view. *AdViewBannerListener* is a interface includes the following notifications:

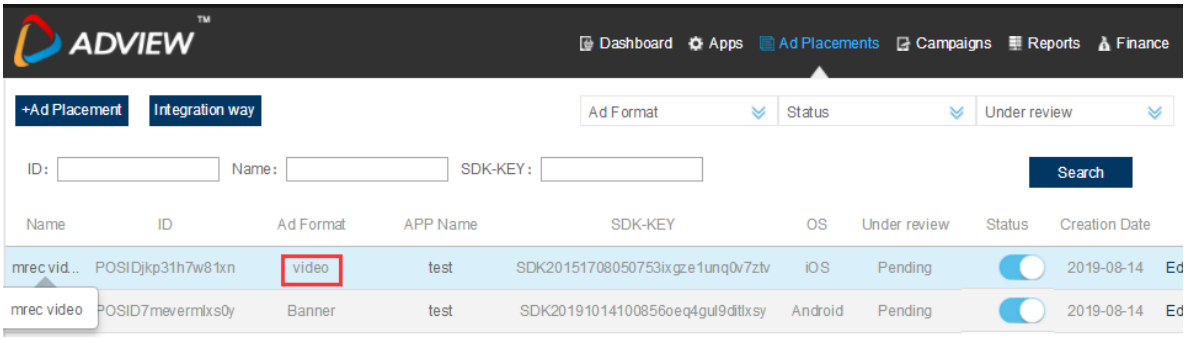
```
public interface AdViewBannerListener {
    void onAdClicked();

    void onAdDisplayed();

    void onAdReceived();        //ad received, before shown

    void onAdFailedReceived(String var1);   //error occur, var1: error message

    void onAdClosed();

    void onAdReady();
}
```

MREC Video

MREC supports the display of video advertisements, but for MREC video, you need to create a video ad placement:



And when you incoming the ad placement id ,create **AdViewBannerManager** and use **setVideoMode**(true) to enable Mrec video supported.



example:

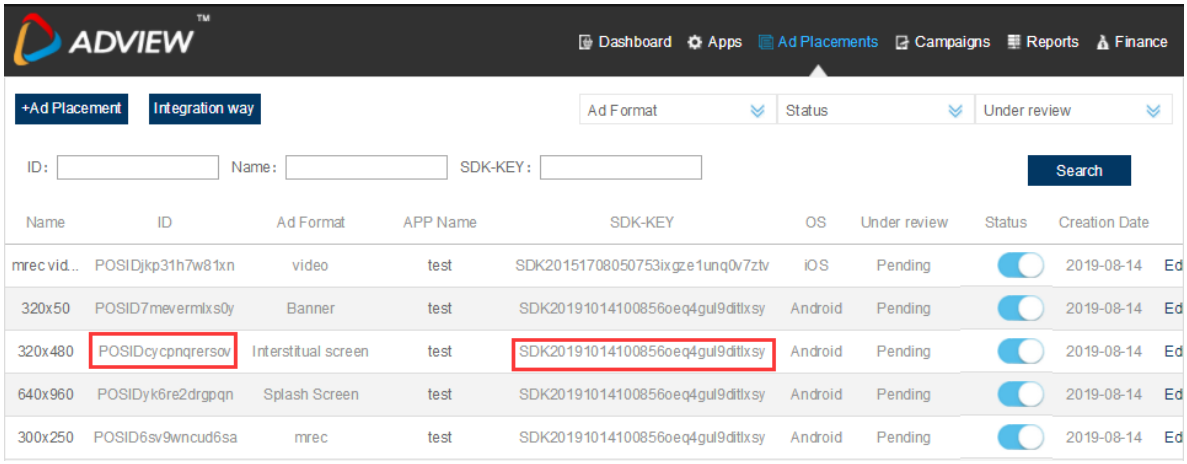
```
adViewBIDView = new AdViewBannerManager(
                    AdBannerActivity.this,
                    appID,
                    posID,
                    adSize,
                    true);

adViewBIDView.setVideoMode(true); //set MRec video mode
```

Interstitial

Interstitial ads provide full-screen experiences, commonly incorporating rich media to offer a higher level of interactivity compared to banner ads. Interstitials are typically shown during natural transitions in your app; for example, after completing a game level, or while waiting for a new view to load.

Create an APP in the Publisher UI for instructions. You will need that Ad Placement ID and SDK-KEY.

Usually when you add an app, you have the option of automatically interstitial ad placement.



1. Create an Interstitial Ad

Income the following parameters and complete interstitial initialization:

income SDK-KEY :

```
public AdViewInstlManager(Context context, String key, String posID, boolean
canClosed)
```

The argument ***canClosed*** means the advisement can be closed by closed button or not , like this:

```
adInstlBIDView = new AdViewInstlManager(
                this,
                appID,
                posID,
                true );
```

2. Display an Interstitial Ad

After initialization, you need to set up a monitoring callback:

```
adInstlBIDView.setOnAdViewListener(this);
```

prototype is:

```
public void setOnAdViewListener(AdViewInstlListener instlListener);
```

When the advertisement is ready, you will receive the following callbacks:

```
public interface AdViewInstlListener {
    ...
    void onAdReady();
}
```

regularly , in *onAdReady()*, it means ad is ready and can be shown as you need, you can call the show method, then ad will pop up to screen:

```
@Override
public void onAdReady() {
    Log.i("AdInstlActivity", "onAdReady");
    /**
     * 1.normal show mode
     */
    adInstlBIDView.showInstl(this);
}
```

anyway, after *onAdReady*() state, you can call *showInstl*() any time during app alive.

```
public boolean showInstl(Context ctx);
```

If you want use a custom show view, such as the back-screen ads , and you can use the following methods to get the ad view, custom display show.

Notice: if you use custom display instead of using *showInstl()*, you must report impression and click event by yourself, use the following methods.

```
public void reportImpression();
public void reportClick();
```

the whole sample (just a reference, more can see demo app) :

```
/**
 * 2.custom show
 */
// if want to use custom show, can use the following references codes
if (adInstlBIDView.getDialogView() != null) {
    new AlertDialog.Builder(AdInstlActivity.this).
         setView(adInstlBIDView.getDialogView()).
         setPositiveButton("View", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                    adInstlBIDView.reportClick();   //report click
                    adInstlBIDView.closeInstl();
            }
         }).setNegativeButton("Exit", new DialogInterface.OnClickListener() {
            @Override
            public void onClick(DialogInterface dialog, int which) {
                adInstlBIDView.closeInstl();
            }
        }).show();
    adInstlBIDView.reportImpression();  //report impression
}
```

3. Using the Delegate

AdViewInstlListener provides a listener interface, which includes a variety of optional callbacks you can use to be notified of events; for example, when an ad has successfully loaded, or when the ad is about to present or dismiss a modal view. *AdViewInstlListener* includes the following methods:

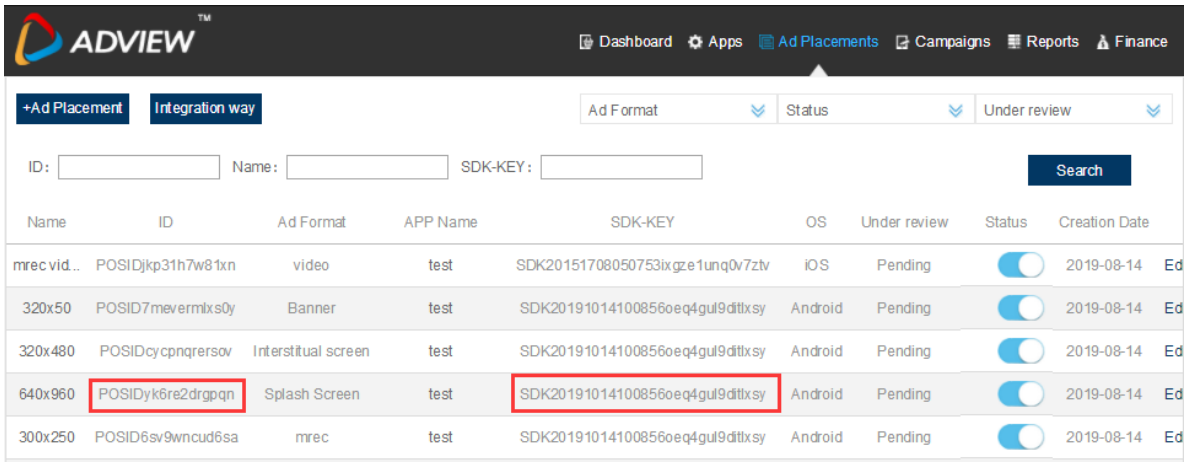
```
public interface AdViewInstlListener {
    void onAdClicked();

    void onAdDisplayed();

    void onAdReceived();    //ad received, before shown

    void onAdFailedReceived(String var1); //error occur, var1: error message

    void onAdClosed();

    void onAdReady();       //ad is ready, can be shown
}
```

Spread

Spread is usually an advertisement displayed at the start of app and usually has higher visibility than banner and insterstitial.

Create an APP in the Publisher UI for instructions. You will need that Ad Placement ID and SDK-KEY.

Usually when you add an app, you have the option of automatically interstitial ad placement.



1. Create an spread Ad

Income the following parameters and complete interstitial initialization:

```
public AdViewSpreadManager(Context context, String key, String posid, ViewGroup
view)
```

also can set background color and logo as you wish:

```
adSpreadBIDView = new AdViewSpreadManager(
                this,
                appID,
                posID,
                (RelativeLayout) findViewById(R.id.spreadlayout));

adSpreadBIDView.setLogo(R.drawable.logo);
adSpreadBIDView.setBackgroundColor(Color.WHITE);
```

Set the top countdown notification method, the default does not notify:

```
adSpreadBIDView.setSpreadNotifyType(AdViewSpreadManager.NOTIFY_COUNTER_CUSTOM);
```

countdown notification type can be following define:

int SpreadNotifyType	Display Items
NOTIFY_COUNTER_NULL = 0	nothing
NOTIFY_COUNTER_NUM = 1	Countdown number
NOTIFY_COUNTER_TEXT = 2	Countdown number & " Skip " text
NOTIFY_COUNTER_CUSTOM = 3	all items by custom design

and then you should set callback for *AdViewSpreadManager*:

```
adSpreadBIDView.setOnAdViewListener(this);
```

prototype:

```
public void setOnAdViewListener(AdViewSpreadListener spreadListener);
```

2. Display spread Ad

Usually spread notify type such as 0-2, it will show spread automatically , if you use **NOTIFY_COUNTER_CUSTOM**, then you must implement the *onAdNotifyCustomCallback()* :

```
public void onAdNotifyCustomCallback( final int ruleTime,final  int delayTime);
```

sample code, also you need add your custom view (more can see demo app) :

```
@Override
public void onAdNotifyCustomCallback( final int ruleTime,final  int delayTime) {
    final TextView tv1 = new TextView(this);
    final Button btn1 = new Button(this);
    final LayoutParams btnLp = new LayoutParams(LayoutParams.WRAP_CONTENT,
            LayoutParams.WRAP_CONTENT);
    final LayoutParams tvLp = new LayoutParams(LayoutParams.WRAP_CONTENT,
            LayoutParams.WRAP_CONTENT);

    tv1.setTextSize(TypedValue.COMPLEX_UNIT_SP, 20);
    btn1.setId(123123);
    tv1.setBackgroundColor(Color.WHITE);
    btn1.setText("Skip");
    tv1.setText(ruleTime + delayTime + "");

    btnLp.addRule(RelativeLayout.ALIGN_PARENT_RIGHT);
    tvLp.addRule(RelativeLayout.LEFT_OF, btn1.getId());
    adSpreadBIDView.getParentLayout().postDelayed(new Runnable() {
        @Override
        public void run() {
            btn1.setVisibility(View.VISIBLE);
            btn1.setOnClickListener(new OnClickListener() {
                @Override
                public void onClick(View v) {
```

```
                    adSpreadBIDView.cancelAd();
                }
            });
        }
    }, ruleTime * 1000);

    adSpreadBIDView.getParentLayout().postDelayed(new Runnable() {
        @Override
        public void run() {
            if (ruleTime + delayTime - count >= 1) {
                tv1.setBackgroundColor(Color.WHITE);
                tv1.setText(ruleTime + delayTime - count + "");
                count++;
                adSpreadBIDView.getParentLayout().postDelayed(this, 1000);
            }
        }
    }, 1000);

    adSpreadBIDView.getParentLayout().addView(btn1, btnLp);
    adSpreadBIDView.getParentLayout().addView(tv1, tvLp);
    btn1.setVisibility(View.INVISIBLE);
}
```

3. Using the Delegate

also, ***AdViewSpreadListener**** provides a listener interface, which includes a variety of optional callbacks you can use to be notified of events.

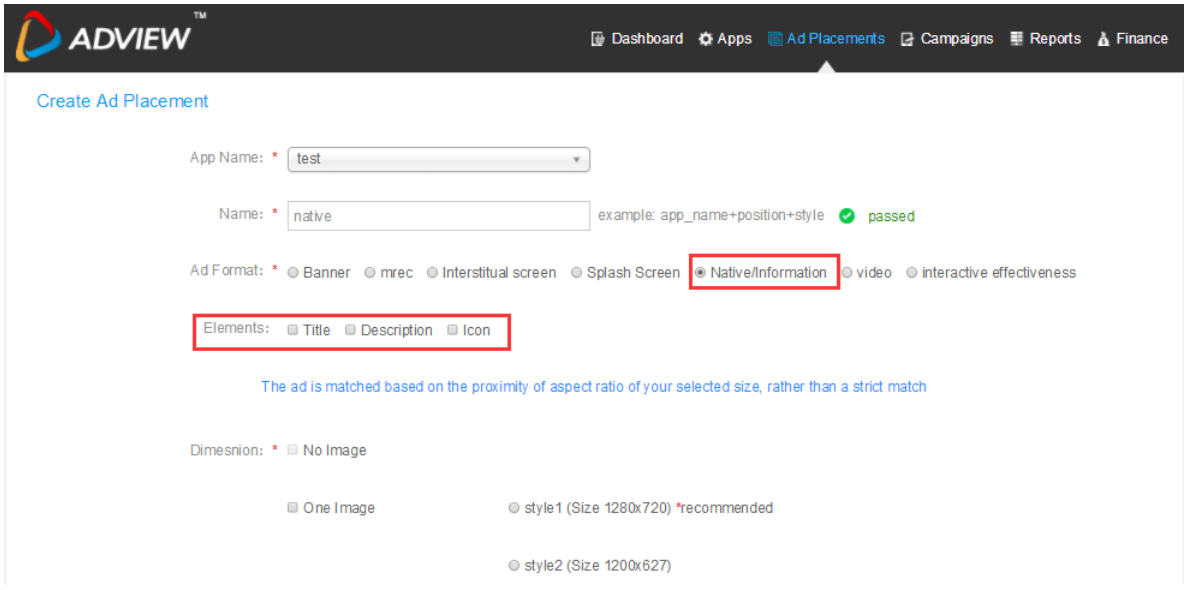
```
public interface AdViewSpreadListener {
    void onAdClicked();

    void onAdDisplayed();

    void onAdReceived();                    //ad received, before shown

    void onAdFailedReceived(String var1);   //error occur, var1:error message

    void onAdClosed();

    void onAdSpreadPrepareClosed();

    void onAdClosedByUser();

    void onAdNotifyCustomCallback(int ruleTime,int delayTime); //custom
countdown
}
```

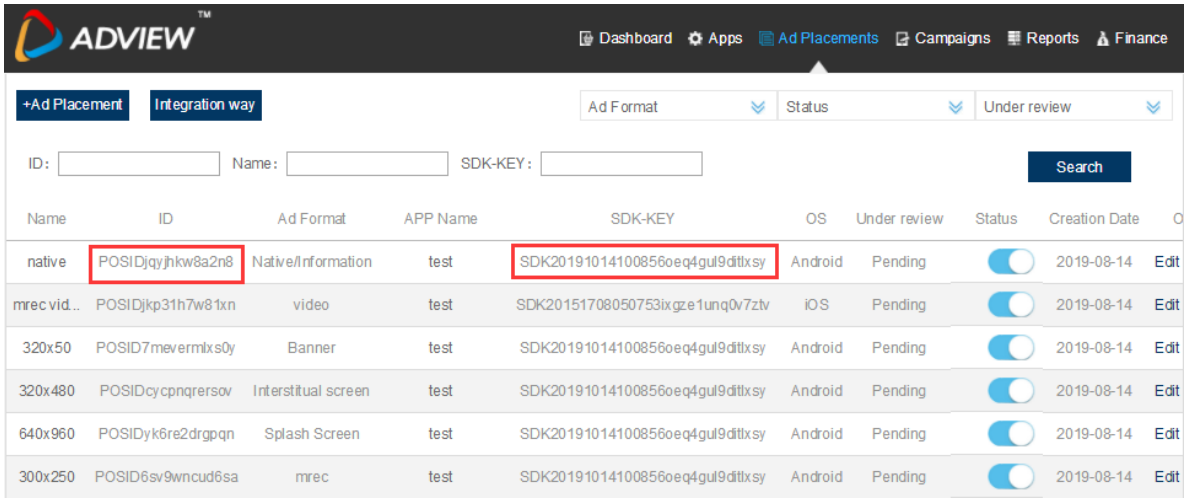
Native

Native ads make it easy for you to monetize your app in a way that's consistent with its existing design. The AdView SDK gives you access to an ad's individual assets so you can design the ad layout to be consistent with the look and feel of your app.

Create an APP in the Publisher UI for instructions. You will need that Ad Placement ID and SDK-KEY.

You need to choose the right creative and image size when creating native ad placement.





1. Create a adview instance, which will be used to register ad listeners, and ad renderers, and to make ad requests. To instantiate your adviewNative instance, specify a Context object, SDK-KEY and ad placement of type String, and a adviewNativeNetworkListener object that will be used to listen to ad load successes/failures, also fill gdpr string if you have gdpr ：

```
public AdViewNativeManager(Context context, String appId, String posId,
                           AdViewNativeListener nativeListener)
```

2. Create an XML layout, specifying how ad assets should be organized, this step is customize by developer.

```
<LinearLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center_horizontal"
        android:orientation="horizontal">
        <ImageView
            android:id="@+id/image"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:adjustViewBounds="true"
            android:scaleType="fitXY"
            app:srcCompat="@drawable/ic_lancher" />
</LinearLayout>
<RelativeLayout
```

```
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:padding="5dp">
        <ImageView
            android:id="@+id/adicon"
            android:layout_width="20dp"
            android:layout_height="10dp"
            android:layout_alignParentLeft="true"
            android:adjustViewBounds="true"
            android:scaleType="fitXY"
            app:srcCompat="@drawable/ic_lancher" />
        <ImageView
            android:id="@+id/adlogo"
            android:layout_width="30dp"
            android:layout_height="10dp"
            android:layout_alignParentRight="true"
            android:adjustViewBounds="true"
            android:scaleType="fitXY"
            app:srcCompat="@drawable/ic_lancher" />
 </RelativeLayout>
```

3. Call *requestAd*() to request an ad from the server and download associated images asynchronously.

```
public void requestAd();
```

4. Display native Ad, after request ad , *onNativeAdReceived*() callback is called when native ad is downloaded from server. Upon failure, *onNativeAdReceiveFailed*() callback is called with a corresponding error code message.

```
@Override
public void onNativeAdReceived(List nativeAdList) {
    Log.i("AdNativeActivity", "onNativeAdReceived");
    native_AdList = nativeAdList;
    //send message to ui thread
    Message message = new Message();
    Bundle bundle = new Bundle();
    message.what = NATIVE_AD_RECEIVED;
    message.setData(bundle);
    this.updateHandler.sendMessage(message);
}
```

according to **nativeAdList** is a list contains items as following:

Normal native mode:

Figure native4.1

Name	Value Type	Description
adId	String	id
adFlagIcon	String	icon url for adview icon
adFlagLogo	String	logo url for adview logo
description	String	description
title	String	ad title string
adImage	String	large image for ad resource
imageWidth	int	image width
imageHeight	int	image height
adIcon	String	icon for ad resource
iconWidth	int	icon width
iconHeight	int	icon height
privacy_image	String	url for privacy information icon
privacy_click	String	url for click privacy information

Vast Video native mode:

Figure native4.2

Name	Value Type	Description
videoUrl	String	url for ad video resource
iconUrl	String	icon url
title	String	title string
description	String	description
duration	Integer	video play duration
adId	String	ad id
preImgUrl	String	image for pre-video
endHtml	String	end html URL for video play done
endImgUrl	String	end html
privacy_image	String	url for privacy information icon
privacy_click	String	url for click privacy information

5. Report Impression & Click , after received Ad's resources, you should call reportImpression() when you want to show the contents in your layout, it will send impression report. and also, in user click event , you should call reportClick() to handle a popup landing-page and report click event.

```
public void reportImpression(View view, String adi);
public void reportClick(View view, String adi, int x, int y);
```

Also interfaces function parameters' definition is below :

Name	Value Type	Description
view	View	view that contains Ad resource
adi	String	Ad id, came from nativeAdList, see Figure native4.1
x	int	click position x , can be 0
y	int	click position y, can be 0

6. Error events, Using the Delegate, *onNativeAdReceived*() is called with native Ad's resources list or other parameters. Upon failure, *onNativeAdReceiveFailed()* method is called with a corresponding error message.

```
public interface AdViewNativeListener {
    void onNativeAdReceived(List<HashMap> var1);

    void onNativeAdReceiveFailed(String var1);

    void onDownloadStatusChange(int var1);

    void onNativeAdClosed(View var1);
}
```

Opt-out

In line with new industry self-regulatory guidance provided by the DAA, AdView now requires all native ads to display the privacy information icon and click region. The icon must be at least 20x20px and can be placed in any one of the four corners of the ad.
You can use your own icon and privacy information page, or use the corresponding icons and privacy information pages provided by AdView, You can get them with **privacy_image** and **privacy_click**, see Figure native4.1 & Figure native4.2.

demo codes for got privacy image :

```
if (null != nativeAd.get("privacy_image")) {
    String url = ((CharSequence) nativeAd.get("privacy_image")).toString();
    privacyImg = getHttpBitmap(url);
}
```

demo codes for privacy click URL, if you don't want to use *adViewNative.showPrivacyInfo()*, you can got click url with *nativeAd.get("privacy_click")* and show it by yourself.

```
//display privacy information
privacyView.setOnClickListener(
    new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            //show privacy information
            if (null != privacyImg) {
                adViewNative.showPrivacyInfo();
            }
        }
    }
);
```

Video

Similar to Interstitial Ads, Video Ads are shown in fullscreen mode and can be placed before, in-between, or after the app content.
First, the Video Player requires activity declaration in your AndroidManifest.xml:

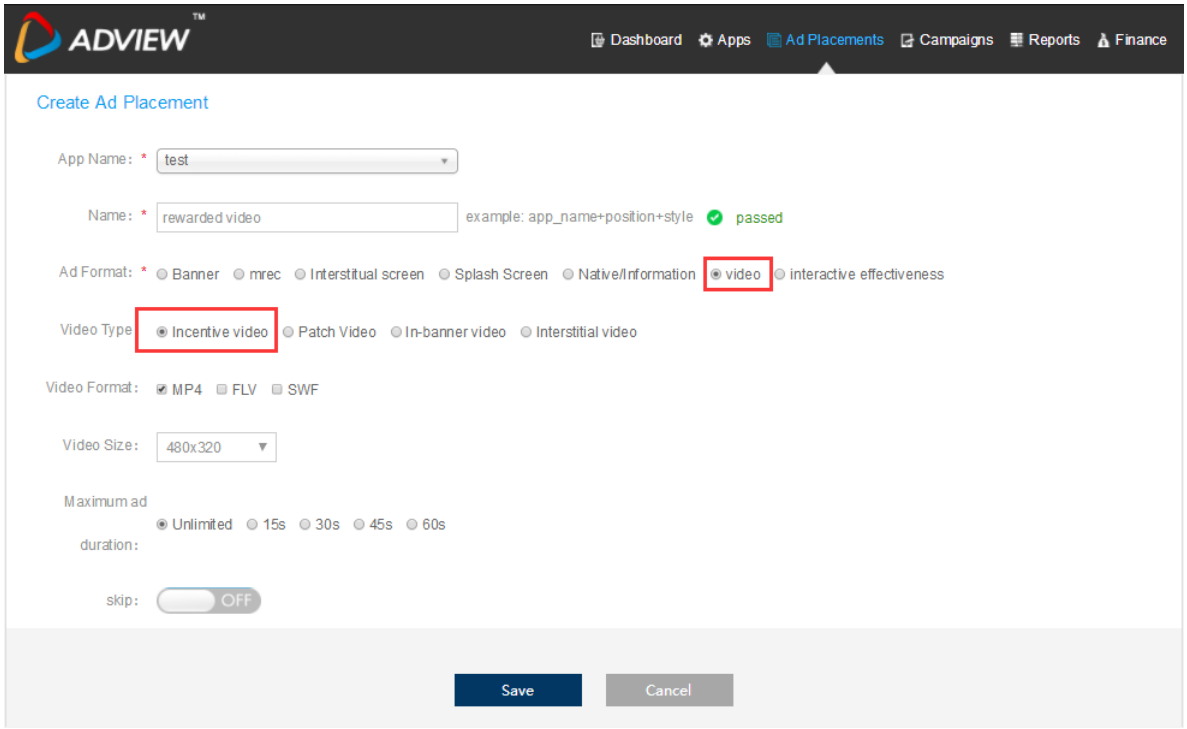
```
<activity
            android:name="com.kuaiyou.video.AdViewVideoActivity"
            android:configChanges="keyboardHidden|orientation|screenSize"
            android:hardwareAccelerated="true" >
</activity>
```

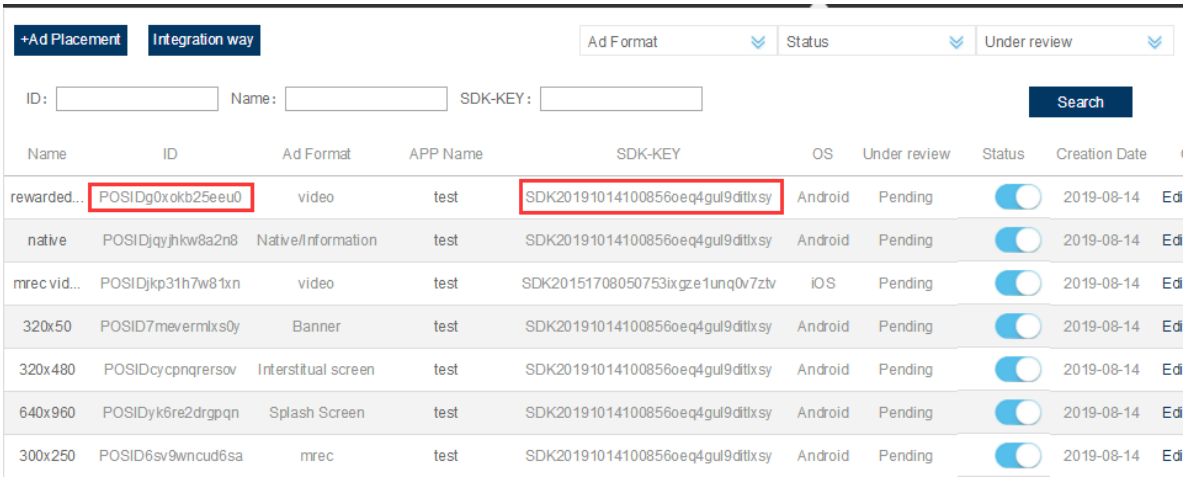
Here recommend to enable *android:hardwareAccelerated* to make video play smooth and better.

Rewarded & Interstitial video

Similar to interstitial ad units, rewarded video ads are shown in full-screen mode and can be placed before, in-between or after the app content.

Create an APP in the Publisher UI for instructions. You will need that Ad Placement ID and SDK-KEY.





1. Create an Rewarded video Ad:

Income the following parameters and complete interstitial initialization:

```
public AdViewVideoManager(Context context, String appId, String posId,
                          AdViewVideoListener appListener, boolean isPaster)
```

income SDK-KEY and Ad Placement ID as **appId** & **posId**. and **isPaster** must be false in Rewarded & Interstitial Video mode, like this:

```
videoManager = new AdViewVideoManager(
                this,
                appID,
                posID,
                advListener,
                false);
// screen orientation , pls refer to ActivityInfo.SCREEN_XXXXXX definition
videoManager.setVideoOrientation(ActivityInfo.SCREEN_ORIENTATION_LANDSCAPE);
```

callback **AdViewVideoListener appListener** is call capture events from video SDK process.

```
public interface AdViewVideoListener extends Serializable {
    void onReceivedVideo(String var1);
    void onFailedReceivedVideo(String var1);
    void onVideoReady();
    void onVideoStartPlayed();
    void onVideoFinished();
    void onVideoClosed();
    void onPlayedError(String var1);
}
```

AdView provides a response function **onVideoReady()* to let you know if the advertisement was successfully requested:

This function is called when the ad request is successful:

```
@Override
public void onReceivedVideo(String arg0) {
    Log.i(TAG, "onRecievedVideo:" + arg0);
}
```

This function is called when an ad request fails:

```
@Override
public void onFailedReceivedVideo(String error) {
    Log.i(TAG, "onFailedRecievedVideo:"+error);
}
```

3. Called when the video ad is ready, after which video shows can be made:

```
@Override
public void onVideoReady() {
    Log.i(TAG, "onVideoReady");
    videoManager.playVideo(this);
}
```

4. You can also monitor the following play status to determine rewards for users:

Called when video ads start playing

```
@Override
public void onVideoStartPlayed() {
    Log.i(TAG, "onVideoStartPlayed");
}
```

Called when video ads have ended

```
@Override
public void onVideoFinished() {
    Log.i(TAG, "onVideoFinished");
}
```

Called when the video ad is closed:

```
@Override
public void onVideoClosed() {
    Log.i(TAG, "onVideoClosed");
}
```

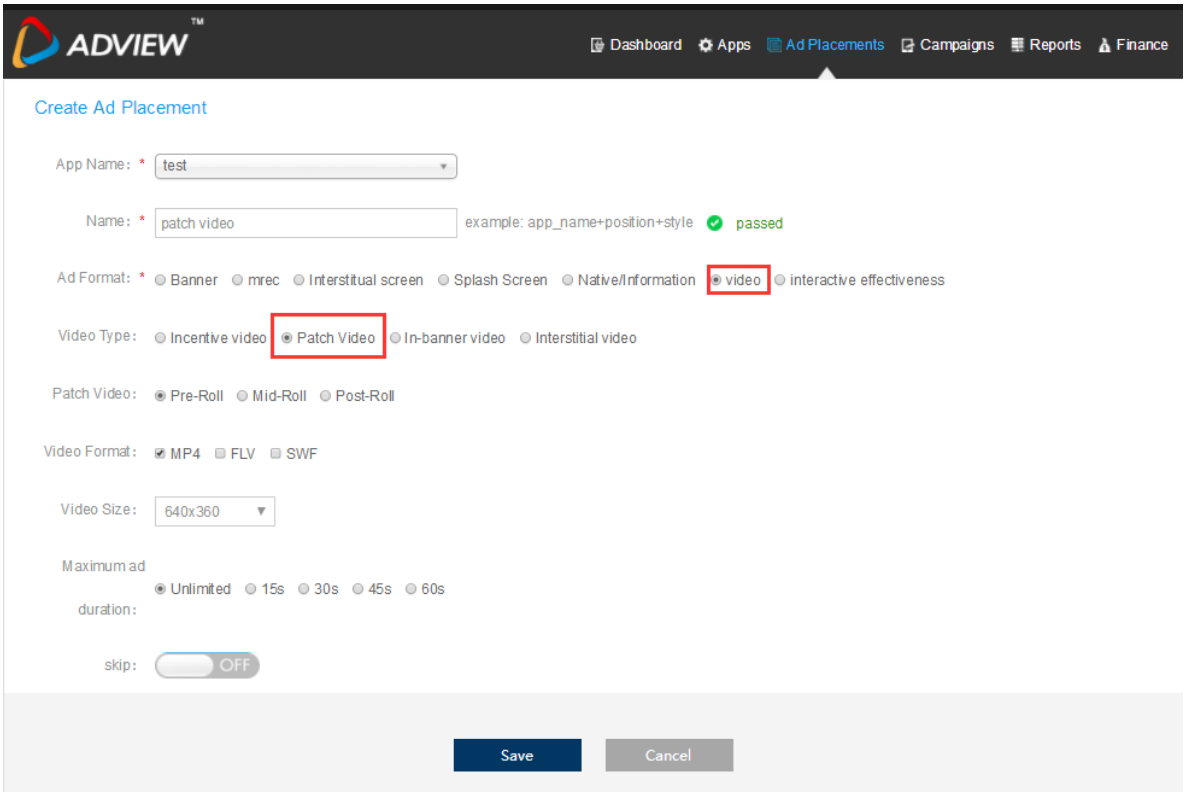
Called when the video ad is playing incorrectly:

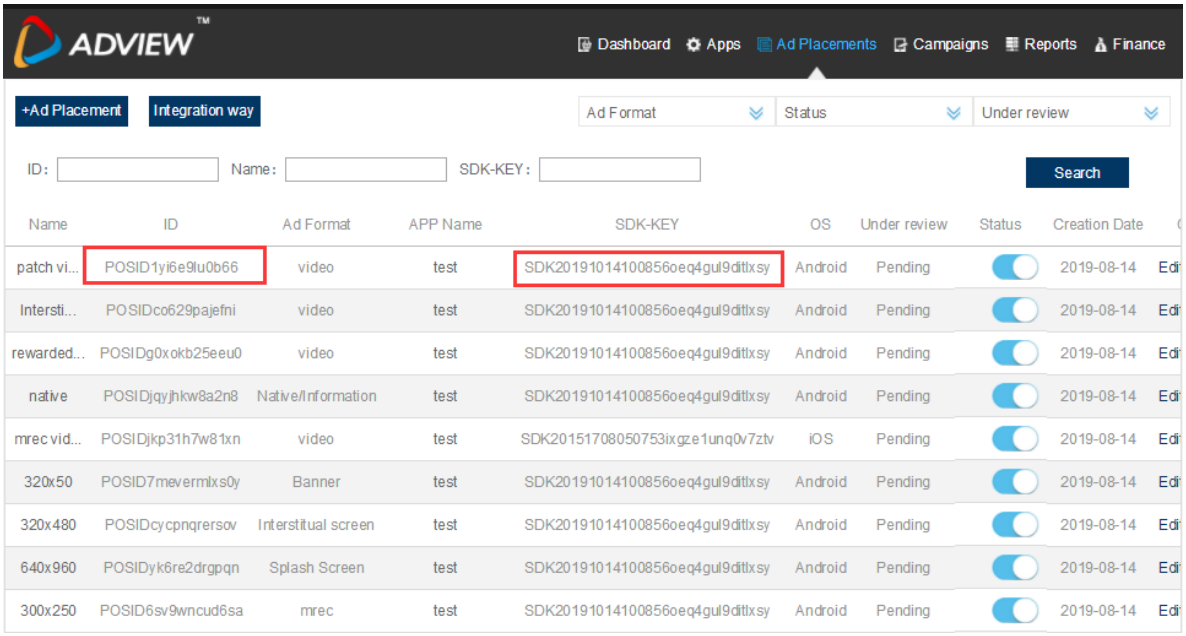
```
@Override
public void onPlayedError(String arg0) {
    Log.i(TAG, "onPlayedError:"+arg0);
}
```

Paste video

Paste video can provide video source (means VAST xml content) to app , and SDK will not show it . though app can parse vast itself and play it as it wish.

Create an APP in the Publisher UI for instructions. You will need that Ad Placement ID and SDK-KEY.





1. Create an paste video ad:

```
videoManager = new AdViewVideoManager(this,
                    APPID,
                    POSID,
                    this,   //AdViewVideoListener listener
                    true,   //this means paste video
                    MainActivity.Gdpr);
```

2. Response for video ad received:

   with paste video mode, after ad video is received, the video contents will be pass through to app with callback *onReceivedVideo()*, in parameter string:

```
@Override
public void onReceivedVideo(String arg0) {
    //arg0 is <Vast> content of video
    Log.i(TAG, "onRecievedVideo:"+arg0);
}
```

3. Play & other process:

   same as rewarded video.

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 *setGDPR()* method to provide gdpr subjects.

prototype:

```
public void setGDPR(boolean cmpPresent,
                    String subjectToGDPR,
                    String consentString,
                    String parsedPurposeConsents,
                    String parsedVendorConsents);
```

for example:

```
adViewBIDView.setGDPR(true,
                          "subject001",
                          MainActivity.Gdpr,
                          "p0001",
                          "vendor001");
```

if you don't use this function, it means not support GDPR.

CCPA

The California Consumer Privacy Act (CCPA) was created to provide California consumers with greater transparency and control over their personal information. In many ways, the CCPA is a first of its kind regulation in the United States that seeks to create broad privacy and data protection rules that apply to all industries doing business in the jurisdiction of California, rather than focusing on a single sector or specific data collection and use practices.

For Publishers with California-Based Users

As a publisher, you need to make sure to request consent from California-based users (to give or refuse consent / to opt-out or opt-in) about private data transfer. This answer should be saved in `SharedPreferences` with key "`IABUSPrivacy_String`" in the US Privacy String format (CCPA Opt-Out Storage Format).

Sample of US Privacy String Saving in SharedPreferences:

```
SharedPreferences sharedPref =
PreferenceManager.getDefaultSharedPreferences(context)
SharedPreferences.Editor editor = sharedPref.edit();
editor.putString("IABUSPrivacy_String", "1YNN"); // for example "1YNN"
editor.commit();
```

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.kuaiyou.**
-keep public class com.kuaiyou.**.* { *; }
-keep public class com.AdVG.**.* { *; }
```

and also, all SDK listeners should not be obfuscated, for a listener callback example :

```
package com.abc.adviewdriver;
class MyListener implements
com.kuaiyou.loader.loaderInterface.AdViewInstlListener {
        public void onAdClicked() {
                ...
        }
        ...
}
```

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

```
-keep com.abc.adviewdriver.MyListener { public void on*(); }
```