Documentation of JioAds SDK 3.16 for Android

Index

An Introduction –JioAds SDK for Android

The JioAds Android SDK is designed to help you integrate Ads in your application and fast track your way to monetizing your Android applications. It provides you with various mechanisms to request for Ads in your Android application.

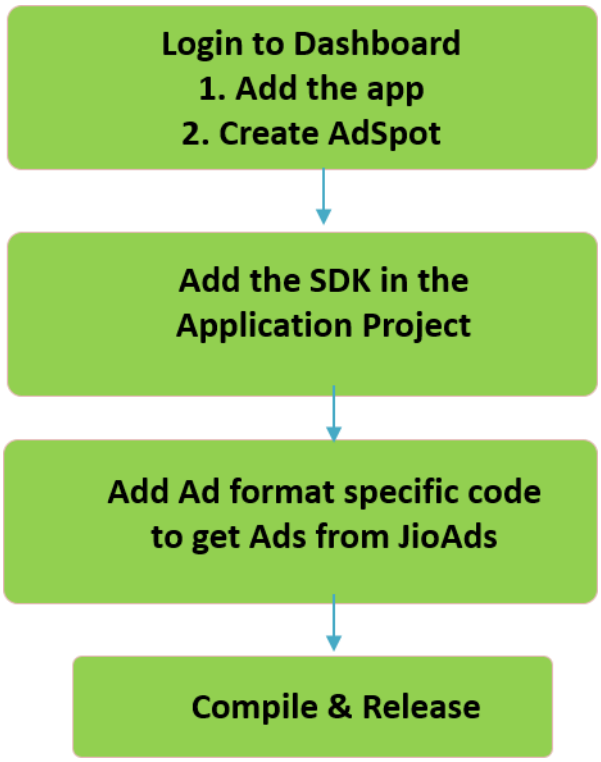
It's a very lightweight, modular & feature-rich SDK. The JioAds SDK delivers highest yield ads for app publishers from JioAds partners.

The SDK have both simple fully-managed Ad request methods. The Cache & show Ad methods to enable prefetching of ads. Our SDK lets you quickly and easily integrate with multiple ad partners.

Prerequisite:

1. Signup JioAds Dashboard
2. Download latest JioAds Android SDK

Flowchart:



JioAds SDK for Android

Step 1: Download the SDK

Before proceeding, ensure that you have downloaded the latest version of the JioAds SDK for Android.

Step 2: Set up the SDK

Extract all the files from the SDK archive into a separate folder. It consists of the following folders and files:

Android Studio/JioAdsSDK/

This is the JioAds SDK for Android Studio packaged as an AAR Project.

Step 3: Reference SDK in project

1. Copy the file vmax.aar from Android Studio/JioAdsSDK / to your application libs folder.
2. Include following dependency in build.gradle

```
compile 'com.vmax.android.ads.aar:vmax@aar'
```

3. [Highly recommended] Add the Google Mobile Ads entry in build.gradle

```
compile 'com.google.android.gms:play-services-ads:17.0.0'
```

4. If you want to fetch Location targeted ads, follow below steps:
    1. Include below dependency in Gradle

```
compile  'com.google.android.gms:play-services-location:16.0.0'
```

    2. Include below permissions to AndroidManifest.xml

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION"/>
 <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION"/>
```

5. If below optional dependency is present then SDK will enhance your ad click experience

```
compile  'com.android.support:customtabs:26.1.0'
```

6. For better monetization, below recommended permissions should be added.

```
<uses-permission android:name="android.permission.BLUETOOTH" />
```

7. At any point of time if you get MultiDex issue then configure your app for multidex by following this documentation.

Step 4: Setup Viewability SDK

Viewability is a metric for ensuring that an ad has the opportunity to be seen. For an ad to be considered viewable, the ad content must be within the viewport of App for a minimum period of time.

Advertisers these days are always careful about viewability metrics for ads and ensure that if they have been actually viewed by their valued targeted audience and not by Bots. viewability measurement has became necessary requirement for many Big brands and advertiser.

To measure viewability for In-App environment, your App needs to incorporate SDKs of viewability partners. MOAT & IAS are leading viewability vendors which advertiser and publisher use in Industry.

Advertiser are willing to pay premium price for viewable ad inventory. App publishers who don't integrate viewability SDK may miss out on ad spending of premium brand & advertiser who's prerequisite is viewable ad inventory. Hence We strongly recommend to use viewblity SDK to inccrease your revenue. To integrate viewability SDK follow the below steps.

1.Copy the file REL-moat-mobile-app-kit-2.4.2.aar from Android Studio/JioAdsSDK /ViewAblitySDK to your application libs folder.

2.Include following dependency in build.gradle

```
compile(name:'REL-moat-mobile-app-kit-2.4.2', ext:'aar')
```

3.Add Moat Adapter

Create a package com.vmax.android.ads.mediation.partners in your app and add the file VmaxMOAT.java to it.

Custom Player Integration for MOAT to track VMAP

If the Ad playback is done by some other sdk and needs to be tracked by MOAT, app publisher needs to make use of below mentioned API's. This guide is customized especially for ads played using Google IMA using Exoplayer.

Note: Copy VmaxMOAT.java from "/Custom Player-IMA" folder (for above step 3)

1. API to dispatch ad progress event

It will track the ad's playback progress. JIO Ads SDK will notify MOAT for ad progress events and MOAT will internally track events like STARTED, FIRST_QUARTILE, MIDPOINT, THIRD_QUARTILE, PAUSED, SKIPPED, COMPLETED

```
public void dispatchEvent(Object adEvent, View adContainer, int currentPositionMillis)
```

It takes object of type AdEvent, reference to the view where the ad is rendered and current position of player in milliseconds.

Following is the usage of the API

```
//To Track the events
if (adsLoader != null) {
    adsLoader.getAdsLoader().addAdsLoadedListener(new AdsLoader.AdsLoadedListener() {
        @Override
        public void onAdsManagerLoaded(AdsManagerLoadedEvent adsManagerLoadedEvent) {
            mAdsManager = adsManagerLoadedEvent.getAdsManager();
            if (mAdsManager != null) {

                mAdsManager.addAdEventListener(new AdEvent.AdEventListener() {
                    ViewabilityTracker viewabilityTracker = vmaxRequest.getViewabilityTracker();
                    @Override
                    public void onAdEvent(AdEvent adEvent) {

                        if(viewabilityTracker!=null)

                            {
                            viewabilityTracker.dispatchEvent(adEvent, (View) simpleExoPlayerView, (int)
(mAdsManager.getAdProgress().getCurrentTime() * 1000));

                             }

                    }
                });
            }
        }
    });
}
```

2. API to track volume change events

If the player has a custom volume control, such as a mute or un-mute button, please notify the tracker about player volume changes with the following API, where playerVolume is the volume of the player normalized on a 0 to 1 scale

```
viewabilityTracker.setPlayerVolume(adEvent, player.getVolume());
```

You can also notify about mute and unmute events

```
viewabilityTracker.setPlayerVolume(adEvent,
ViewabilityTracker.VOLUME_EVENTS.MUTED); // or UNMUTED
```

3. API to change the target player view

If the View where the ad is rendered changes at any point during playback, you should let the tracker know about the new View

```
viewabilityTracker.changeTargetView(View view);
```

Step 5: ProGuard Configuration

If you are running ProGuard on your Android application, ensure that the following directive is added in your ProGuard configuration file:

```
-keepattributes
*Annotation*,JavascriptInterface,Exceptions,InnerClasses,Signature,*Annotation*,EnclosingMethod
,*Annotation*,Signature

-dontwarn com.google.**
-dontwarn com.google.firebase.**
-dontwarn com.google.android.gms.**

-keep public class com.vmax.android.ads.util.UrlConstants {
   public <fields>;
   public <methods>;
}

-keep public class com.vmax.android.ads.util.UrlConstants$** {
   public <fields>;
   public <methods>;
}

-keep public class com.vmax.android.ads.api.VmaxAdView {
   public <fields>;
   public <methods>;
}

-keep public class com.vmax.android.ads.api.NativeAd {
```

```
    public <fields>;
    public <methods>;
}

-keep public class com.vmax.android.ads.api.AdContainer {
    public <fields>;
    public <methods>;
}



-keep public class com.vmax.android.ads.api.VmaxSdk {
    public <fields>;
    public <methods>;
}

-keep public class com.vmax.android.ads.api.ImageLoader {
    public <fields>;
    public <methods>;
}

-keep public class com.vmax.android.ads.api.NativeImageDownload {
    public <fields>;
    public <methods>;
}

-keep public class com.vmax.android.ads.api.NativeImageDownloadListener {
    public <fields>;
    public <methods>;
}

-keep public class com.vmax.android.ads.api.VmaxAdPartner {
    public <fields>;
    public <methods>;
}

-keep public class com.vmax.android.ads.common.VmaxAdListener {
    <fields>;
    <methods>;
}

-keep public class com.vmax.android.ads.api.VmaxRequest {
    public <fields>;
    public <methods>;
}

-keep public class com.vmax.android.ads.common.AdCustomizer {
    public <fields>;
    public <methods>;
}

-keep public class com.vmax.android.ads.common.AdCustomizer$Builder {
   <fields>;
   public <methods>;
```

```
  public static **[] values();
  public static ** valueOf(java.lang.String);
}

-keep public class com.vmax.android.ads.exception.VmaxRequestError {
   public <fields>;
   public <methods>;
}


-keep public class com.vmax.android.ads.exception.VmaxAdError {
   public <fields>;
   public <methods>;
}

-keep public class com.vmax.android.ads.mediation.** {
   public <fields>;
   public <methods>;
   }

-keep class com.vmax.android.ads.mediation.partners.** {
   public <fields>;
   public <methods>;
}

-keep public class com.vmax.android.ads.nativeads.** {
   public <fields>;
   public <methods>;
}

-keep public class com.vmax.android.ads.util.Utility {
   <fields>;
   <methods>;
}

-keep public class com.vmax.android.ads.util.Constants {
   public <fields>;
   public <methods>;
}

-keep public class com.vmax.android.ads.util.Constants$** {
   public <fields>;
   public <methods>;
}


-keep public class com.vmax.android.ads.util.IntentUtils {
   <fields>;
   <methods>;
}

-keep public class com.vmax.android.ads.common.VmaxRequestListener {
   <fields>;
   <methods>;
```

```
}

-keep class com.vmax.android.ads.util.gifsupport.** {
   <fields>;
   <methods>;
}

#Custom show Ad

-keep public class com.vmax.android.ads.common.IVmaxAdQueue {
   public <fields>;
   public <methods>;
}
-keep public class com.vmax.android.ads.common.IVmaxAdEvents {
   public <fields>;
   public <methods>;
}
-keep public class com.vmax.android.ads.api.VmaxAd {
   public <fields>;
   public <methods>;
}
-keep public class com.vmax.android.ads.common.VmaxTracker {
    public <fields>;
    public <methods>;
 }



 -keep public class com.google.android.gms.** {
   <fields>;
  <methods>;
}

-keep public class com.google.ads.** {
   public <fields>;
   public <methods>;
}

#Google IMA
-keep class com.google.** { *; }
-keep interface com.google.** { *; }
-keep class com.google.ads.interactivemedia.v3.api.** { *;}
-keep interface com.google.ads.interactivemedia.** { *; }

-dontwarn com.google.ads.interactivemedia.v3.api.**

-keep public class com.google.android.gms.common.internal.safeparcel.SafeParcelable {
   public static final *** NULL;
}

# Keep names - Native method names. Keep all native class/method names.
-keepclasseswithmembers class * {
   native <methods>;
}
```

```
-keep public enum  com.vmax.android.ads.api.VmaxAdView$AdState {
   <fields>;
   public static **[] values();
   public static ** valueOf(java.lang.String);
}

-keep public enum  com.vmax.android.ads.api.VmaxAdView$MediaQuality{
  <fields>;
   public static **[] values();
   public static ** valueOf(java.lang.String);
}

-keep public enum  com.vmax.android.ads.api.VmaxAdView$AdOrientation{
  <fields>;
   public static **[] values();
   public static ** valueOf(java.lang.String);
}


-keep public enum  com.vmax.android.ads.api.VmaxAdView$AdViewState {
   <fields>;
   public static **[] values();
   public static ** valueOf(java.lang.String);
}


-keep public enum  com.vmax.android.ads.api.VmaxAdView$AdPodVariant {
   <fields>;
   public static **[] values();
   public static ** valueOf(java.lang.String);
}

-keep public enum  com.vmax.android.ads.api.VmaxAdView$AdspotSize {
   <fields>;
   public static **[] values();
   public static ** valueOf(java.lang.String);
}

-keep public enum  com.vmax.android.ads.api.AddOns$Environment {
   <fields>;
   public static **[] values();
   public static ** valueOf(java.lang.String);
}

-keep public enum  com.vmax.android.ads.api.VmaxSdk$LogLevel {
   <fields>;
   public static **[] values();
   public static ** valueOf(java.lang.String);
}

-keep public enum  com.vmax.android.ads.api.VmaxSdk$Gender {
   <fields>;
   public static **[] values();
```

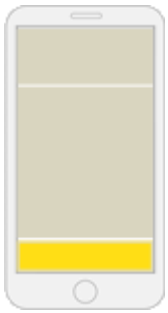
```
  public static ** valueOf(java.lang.String);
}

-keep public enum  com.vmax.android.ads.api.VmaxSdk$CacheMode {
  <fields>;
  public static **[] values();
  public static ** valueOf(java.lang.String);
}

-keep public enum  com.vmax.android.ads.api.VmaxSdk$ContentVideoPlayer {
  <fields>;
  public static **[] values();
  public static ** valueOf(java.lang.String);
}

-keep public enum  com.vmax.android.ads.api.VmaxSdk$ContentVideoHandler {
  <fields>;
  public static **[] values();
  public static ** valueOf(java.lang.String);
}

-keep public enum  com.vmax.android.ads.api.VmaxSdk$ViewabilityPartner {
  <fields>;
  public static **[] values();
  public static ** valueOf(java.lang.String);
}

-keep public enum  com.vmax.android.ads.api.VmaxSdk$RequestType {
  <fields>;
  public static **[] values();
  public static ** valueOf(java.lang.String);
}

-keep public enum  com.vmax.android.ads.api.VmaxSdk$MediaType {
  <fields>;
  public static **[] values();
  public static ** valueOf(java.lang.String);
}

-keep public class com.vmax.android.ads.api.Section {
   public <fields>;
   public <methods>;
}

-keep public enum  com.vmax.android.ads.api.Section$** {
  <fields>;
  public static **[] values();
  public static ** valueOf(java.lang.String);
}

-keep public class com.vmax.android.ads.common.vast.VmaxTrackingEventInterface {
   <fields>;
   <methods>;
}
```

```
-keep public class com.vmax.android.ads.api.VmaxAdEvent {
   <fields>;
   <methods>;
}

-keep public class com.google.android.gms.location.FusedLocationProviderClient {
   public <fields>;
   public <methods>;
}
-keep public class com.google.android.gms.tasks.OnSuccessListener {
   <fields>;
   <methods>;
}

-keep public class com.vmax.android.ads.common.DataReceiver {
   public <fields>;
   public <methods>;
}

-keep public class com.vmax.android.ads.common.AppInstallReceiver {
   public <fields>;
   public <methods>;
}

-keep public class com.vmax.android.ads.common.AdsSPCListener {
   <fields>;
   <methods>;
}



#Chrome Custom Tab
-keep public class android.support.customtabs.CustomTabsIntent {
   public <fields>;
   public <methods>;
}

#GDPR Proguard
-keep public class com.vmax.android.ads.common.RegionCheckListener {
   <fields>;
   <methods>;
}

#MOAT Custom Plyer integration
-keep public class com.vmax.android.ads.api.ViewabilityTracker {
   public <fields>;
   public <methods>;
}

#OM Custom Plyer integration
-keep public class com.iab.omid.library.vmax.** { *; }
-dontwarn com.iab.omid.library.vmax.**
```

```
-keep public enum  com.vmax.android.ads.api.ViewabilityTracker$VOLUME_EVENTS {
   <fields>;
   public static **[] values();
   public static ** valueOf(java.lang.String);
}

-keep,allowshrinking @com.google.android.gms.common.annotation.KeepName class *


-keep public enum  com.vmax.android.ads.api.VmaxSdk$AdChoicePlacement {
   <fields>;
   public static **[] values();
   public static ** valueOf(java.lang.String);
}
```

Enabling Ads - Android SDK

Banner

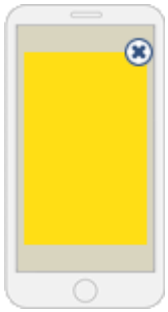


A banner adspot typically covers up to 20% of the app's visible area. You can use banners that are displayed at the top or bottom of the app/game content. Banner ads are display / richmedia ads which can either be sticky on in-line with your app.

Billboard

A billboard adspot typically between 20% and 70% of the app's visible area. You can use billboards that are displayed at the top or bottom of the app/game content. Static or animated ads displayed over a portion of the app's or game's existing content.

Interstitial

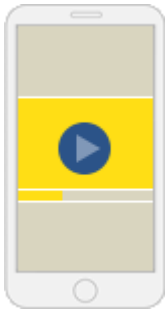


A Interstitial adspot typically covers up the app's entire visible area. You can either use interstitial ads (static, rich media and video) as well as content stream ads. Full-screen image, rich-media, or video ads that are displayed at natural pauses in the app or game

Rewarded video

Ads that reward users with virtual currency or premium content for viewing video trailers. Ads that reward users with virtual currency or premium content for viewing video trailers. You can use this adspot to show rewarded video ads that can be used to reward users with in-app currency on view or completing actions.

In-Stream Video



Video commercials displayed in the same area where you will play the video content selected by a user.Video commercials displayed just before (pre-roll), in-between (mid-roll) or after (post-roll) playing the video content selected by a user

In-Stream Audio

Audio ads played seamlessly into the users listening experience before, during or after audio content with accompanied by ad image with call to action

Native

Ads that are consistent with the form and function of the existing app design. Native Ads can be of the following types: in-feed, content stream.

Integrating Banner Ads(XML Approach-Recommended) – JioAds SDK for Android

Supported Banner Sizes

Size (WxH)	Description	Availability
320×50	Standard Banner	Phones
728×90	IAB Leaderboard	Tablets / STB

The SDK will automatically request the ad-size based on the user's device.There isn't a need for the developer to pass ad-size explicitly during integration.

Enabling Banner Ads

The JioAds SDK provides a custom **View** subclass, **VmaxAdview**, that handles requesting and showing ads.
Modify your **layout** file so that it includes the **VmaxAdview** . This method is preferred over programmatically adding things to an **Activity** using Java code as it offers better separation of presentation and behaviour. However, if you want to programmatically show ads , you need to request for banner ads programmatically.

Add VmaxAdview to your layout

In your **layout** file (for example: **/res/layout/activity_main.xml**), you need to add **VmaxAdview** that will act as a container for the banner ads. Include this XML block in your **Activity layout**.

```
<com.vmax.android.ads.api.VmaxAdView

        xmlns:vmax="http://schemas.android.com/apk/res-auto"

        android:id="@+id/xml_container"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:layout_gravity="center"

        android:background="@android:color/transparent"

        android:gravity="center"

        vmax:adspotKey="<ADSPOT_KEY_GOES_HERE>"
```

```
        vmax:uxType="@integer/vmax_ux_banner"

        />
```

Initializing VmaxAdView object

Place this code snippet in the **Activity/fragment** class to initialize the **VmaxAdView object** to manage it for **Activity life-cycle** changes

```
VmaxAdView vmaxAdViewBanner=(VmaxAdView)findViewById(R.id.xml_container);
```

Requesting Strict Banner Size(Optional)

You can request strict banner size using below api which optional and is not recommended since it is likely to induce lesser fill

```
//For STANDARD BANNER OF SIZE 320x50

vmax:strictSize ="@string/vmax_standard_banner"



//For LEADERBOARD OF SIZE 728x90

vmax:strictSize ="@string/vmax_leaderboard"
```

Handling Activity life-cycle changes

Lastly, add the following code to destroy **vmaxAdView** when **Activity** is no longer available

```
@Override

   protected void onDestroy() {

        if (vmaxAdViewBanner != null) {



           vmaxAdViewBanner.onDestroy();



        }

        super.onDestroy();

   }
```

FAQs

Q: Do I need to manage refresh?
A: No, JioAds Sdk will automatically handle refreshing of Ads.

Q: Can I customize the refresh rate for banner ads?
A: The default refresh rate for banner ads is set at 30 seconds to ensure a good ad yield without compromising on the user experience. You can manage the the frequency at which the Banner Ads are refreshed using the **vmax:refreshRate**attribute. Use the following code to set a custom refresh rate:

```
<com.vmax.android.ads.api.VmaxAdView

       xmlns:vmax="http://schemas.android.com/apk/res-auto"

       android:id="@+id/xml_container"

       android:layout_width="wrap_content"

       android:layout_height="wrap_content"

       android:layout_gravity="center"

       android:background="@android:color/transparent"

       android:gravity="center"

       vmax:adspotKey="<ADSPOT_KEY_GOES_HERE>"

       vmax:refreshRate="<REFRESH RATE IN SECONDS>"

       vmax:uxType="@integer/vmax_ux_banner"

       />
```

As the minimum refresh rate is 30 seconds; setting a lower value will default it to 30 seconds, unlike the value 0 which indicates that refresh is disabled.

Integrating Banner Ads (Programatic Approach) – JioAds SDK for Android

Enabling Banner Ads

This section explains rendering banner ads programatically within the same **Activity**. if you want to cache a banner ad in one **Activity** and show it in a different **Activity** you can refer Integrating Banner Ads (Advanced).

Create a Container For Ads

Place this code snippet in the **layout** in which you want to show your Ad.

```
    <FrameLayout

      android:id="@+id/adview"

      android:layout_width="wrap_content"

      android:layout_height="wrap_content"

      android:gravity="center"

      android:background="@android:color/transparent"

      android:layout_gravity="center"

      android:visibility="gone" />
```

Show banner ads

Place this code snippet in an **Activity** class to show banner ad.

```
 final VmaxAdView vmaxAdViewBanner = new VmaxAdView(this, <ADSPOT_KEY_GOES_HERE>,
VmaxAdView.UX_BANNER);

 final FrameLayout bannerContainer=(FrameLayout)findViewById(R.id.adview);




  vmaxAdViewBanner.setAdListener(new VmaxAdListener() {

             @Override

            public void onAdReady(VmaxAdView adView) {

//              wont be triggrred
```

```
            }



            @Override
            public void onAdError(VmaxAdError Error, VmaxAdView adView) {
            }



            @Override
            public void onAdClose(VmaxAdView adView) {
            }



            @Override
            public void onAdMediaEnd(boolean isMediaCompleted, long reward,
VmaxAdView adView) {
            }
        });
   bannerContainer.removeAllViews();
   bannerContainer.addView(vmaxAdViewBanner);
 /*Requesting Strict Banner Size(Optional)
You can request strict banner size using below api which optional and is not
recommended since it is likely to induce lesser fill*/
//For STANDARD BANNER OF SIZE 320x50
adView.setStrictSize(VmaxAdView.AdspotSize.STANDARD_BANNER_320x50);
//For LEADERBOARD OF SIZE 728x90
adView.setStrictSize(VmaxAdView.AdspotSize.LEADERBOARD_728x90);
 vmaxAdViewBanner.showAd();
```

Handling Activity life-cycle changes

Lastly, add the following code to destroy **vmaxAdView** when **Activity** is no longer available

```
@Override

   protected void onDestroy() {

       if (vmaxAdViewBanner != null) {



          vmaxAdViewBanner.onDestroy();



       }

       super.onDestroy();

   }
```

FAQs

Q: Do I need to manage refresh?
A:No, JioAds Sdk will automatically handle refreshing of Ads.

Q: Can I customize the refresh rate for banner ads?
A: The default refresh rate for banner ads is set at 30 seconds to ensure a good ad yield without compromising on the user experience. You can manage the the frequency at which the Banner Ads are refreshed using the **setRefreshRate()**method. Use the following code to set a custom refresh rate:

```
vmaxAdViewBanner.setRefreshRate(<REFRESH_RATE_IN_SECONDS>);
```

As the minimum refresh rate is 30 seconds; setting a lower value will default it to 30 seconds, unlike the value 0 which indicates that refresh is disabled.

Integrating Banner Ads (Advanced) – JioAds SDK for Android

JioAds SDK allows you to cache a banner ad in one **Activity** and show it in a different Activity.You can also use this approach in a single **Activity** if you wish to do so.

Cache banner ad in one Activity

In order for the SDK to maintain Ad-states across **Activity** you need to create a mutable instance of **VmaxAdView** class Place this code snippet in an **Activity** class to initialize the SDK and cache the ad.

```
 VmaxAdView vmaxAdViewBanner =
VmaxAdView.getMutableInstance(this,"<ADSPOT_KEY_GOES_HERE>",VmaxAdView.UX_BANNER);

 /*Requesting Strict Banner Size(Optional)
You can request strict banner size using below api which optional and is not
recommended since it is likely to induced lesser fill*/

//For STANDARD BANNER OF SIZE 320x50

adView.setStrictSize(VmaxAdView.AdspotSize.STANDARD_BANNER_320x50);

//For LEADERBOARD OF SIZE 728x90

adView.setStrictSize(VmaxAdView.AdspotSize.LEADERBOARD_728x90);

vmaxAdViewBanner.cacheAd();
```

Create a container for banner ad

Place this code snippet in the **layout** in which you want to show your ad.

```
    <FrameLayout

      android:id="@+id/adview"

      android:layout_width="wrap_content"

      android:layout_height="wrap_content"

      android:gravity="center"

      android:background="@android:color/transparent"

      android:layout_gravity="center"
```

```
      android:visibility="gone" />
```

Show banner ad in another Activity

Place this code snippet in the second **Activity** class to show a cached ad. Modifying the **Activty context** through the method **setBaseContext()** is necessary before calling **showAd()**.

```
 AdContainer singleton = AdContainer.getInstance();

 VmaxAdView vmaxAdViewBanner= singleton.getAdView("<ADSPOT_KEY_GOES_HERE>");

 vmaxAdViewBanner.setAdListener(new VmaxAdListener() {

     @Override

    public void onAdError(VmaxAdError error, VmaxAdView adView) {

    }

    @Override

    public void onAdReady(VmaxAdView adView) {

    }

    @Override

    public void onAdClose(VmaxAdView adView) {

    }

    @Override

    public void onAdMediaEnd(boolean isMediaCompleted, long reward, VmaxAdView
adView) {

    }

  });

  FrameLayout bannerContainer=(FrameLayout) findViewById(R.id.adview);

  if (vmaxAdViewBanner.getContext() != null) {

   ((MutableContextWrapper) vmaxAdViewBanner.getContext()).setBaseContext(this);

    }

    bannerContainer.removeAllViews();

    bannerContainer.addView(vmaxAdViewBanner);

 if (vmaxAdViewBanner.getAdState() == VmaxAdView.AdState.STATE_AD_READY){

    vmaxAdViewBanner.showAd();

    bannerContainer.setVisibility(View.VISIBLE);
```

```
}
```

FAQs

Q: Can I Refresh Ads?
A:The option to Refresh ads isn’t available when you use the advanced approach to show ads.

Integrating Billboard Ads (XML Approach-Recommended) – JioAds SDK for Android

Supported Billboard Sizes

Size (WxH)	Description	Availability
320×250	Medium Rectangle	Phones / Tablet / STB

The SDK will automatically request the ad-size based on the user's device.There isn't a need for the developer to pass ad-size explicitly during integration.

Enabling Billboard Ads

The JioAds SDK provides a custom **View subclass**, **VmaxAdview**, that handles requesting and showing ads.
Modify your **layout** file so that it includes the **VmaxAdview** . This method is preferred over programmatically adding things to an **Activity** using Java code as it offers better separation of presentation and behaviour.

Add VmaxAdview to your layout

In your layout file (for example: **/res/layout/activity_main.xml**), you need to add **VmaxAdview** that will act as a container for the billboard ads. Include this XML block in your **Activity layout**.

```
<com.vmax.android.ads.api.VmaxAdView

      xmlns:vmax="http://schemas.android.com/apk/res-auto"

      android:id="@+id/xml_container"

      android:layout_width="wrap_content"

      android:layout_height="wrap_content"

      android:layout_gravity="center"

      android:background="@android:color/transparent"

      android:gravity="center"
```

```
        vmax:adspotKey="<ADSPOT_KEY_GOES_HERE>"

        vmax:uxType="@integer/vmax_ux_billboard"

        />
```

Initializing VmaxAdView object

Place this code snippet in the **Activity class** to initialize the **VmaxAdView** object to manage it for **Activity life-cycle** changes

```
VmaxAdView vmaxAdViewBillboard=(VmaxAdView)findViewById(R.id.xml_container);
```

Handling Activity life-cycle changes

Lastly, add the following code to destroy **vmaxAdView** when **Activity** is no longer available

```
@Override

   protected void onDestroy() {

        if (vmaxAdView != null) {



           vmaxAdView.onDestroy();

        }

        super.onDestroy();

   }
```

FAQs

Q:Can I run video ads in Billboard adspace?
A: Yes, availability of video ads depends on campaign and network

Q: Do I need to manage refresh?
A: No, JioAds Sdk will automatically handle refreshing of Ads.

Q: Can I customize the refresh rate for banner ads?
A: The default refresh rate for banner ads is set at 30 seconds to ensure a good ad yield without compromising on the user experience. You can manage the the frequency at which the Billboard Ads are refreshed using the **vmax:refreshRate**attribute. Use the following code to set a custom refresh rate:

```
 <com.vmax.android.ads.api.VmaxAdView

        xmlns:vmax="http://schemas.android.com/apk/res-auto"

        android:id="@+id/xml_container"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"
```

```
android:layout_gravity="center"

android:background="@android:color/transparent"

android:gravity="center"

vmax:refreshRate="<REFRESH_RATE_IN_SECONDS>"

vmax:adspotKey="<ADSPOT_KEY_GOES_HERE>"

vmax:uxType="@integer/vmax_ux_billboard"

/>
```

As the minimum refresh rate is 30 seconds; setting a lower value will default it to 30 seconds,unlike the value 0 which indicates that refresh is disabled.

Integrating Billboard Ads (Programatic Approach) – JioAds SDK for Android

Enabling Billboard Ads

This section explains rendering billboard ads programatically within the same **Activity**

Create a container for ad

Place this code snippet in the **layout** in which you want to show your Ad.

```
    <FrameLayout

      android:id="@+id/adview"

      android:layout_width="wrap_content"

      android:layout_height="wrap_content"

      android:gravity="center"

      android:background="@android:color/transparent"

      android:layout_gravity="center"

      android:visibility="gone" />
```

view rawProgrammatic Container.xml

Show billboard ad

Place this code snippet in your **Activity** class to show billboard ads.

```
 final VmaxAdView vmaxAdViewBillboard = new VmaxAdView(this, <ADSPOT_KEY_GOES_HERE>,
VmaxAdView.UX_BILLBOARD);

 final FrameLayout billboardContainer=(FrameLayout)findViewById(R.id.adview);




  vmaxAdViewBillboard.setAdListener(new VmaxAdListener() {

             @Override

            public void onAdReady(VmaxAdView adView) {

              // Wont Be Triggerd

              }
```

```
            @Override
            public void onAdError(VmaxAdError Error, VmaxAdView adView) {
            }


            @Override
            public void onAdClose(VmaxAdView adView) {
            }


            @Override
            public void onAdMediaEnd(boolean isMediaCompleted, long reward,
VmaxAdView adView) {
            }
        });
   billboardContainer.removeAllViews();
   billboardContainer.addView(vmaxAdViewBillboard);
   vmaxAdViewBillboard.showAd();
```

Handling Activity life-cycle changes

Lastly, add the following code to destroy **vmaxAdView** when **Activity** is no longer available

```
@Override
    protected void onDestroy() {
        if (vmaxAdViewBillboard != null) {

            vmaxAdViewBillboard.onDestroy();
        }
```

```
        super.onDestroy();

    }
```

FAQs

Q:Can I run video ads in Billboard adspace?
A: Yes, availability of video ads depends on campaign and network

Q: Do I need to manage refresh?
A: No, JioAds Sdk will Automatically handle refreshing of Ads.

Q: Can I customize the refresh rate for billboard ads?
A: The default refresh rate for billboard ads is set at 30 seconds to ensure a good ad yield without compromising on the user experience. You can manage the the frequency at which the Banner Ads are refreshed using the setRefreshRate() method. Use the following code to set a custom refresh rate:

```
vmaxAdViewBillboard.setRefreshRate(<REFRESH_RATE_IN_SECONDS>);
```

As the minimum refresh rate is 30 seconds; setting a lower value will default it to 30 seconds, unlike the value 0 which indicates that refresh is disabled.

Integrating Billboard Ads (Advanced) – JioAds SDK for Android

JioAds SDK allows you to cache a billboard ad in one **Activity** and show it in a different **Activity**.You can also use this approach in a single **Activity** if you wish to do so.

Cache billboard ads in one Activity

In order for the SDK to maintain Ad-states across **Activity** you need to create a mutable instance of **VmaxAdView** class. Place this code snippet in an **Activity** class to initialize the SDK and cache the ad.

```
VmaxAdView vmaxAdViewBillboard =
VmaxAdView.getMutableInstance(this,"<ADSPOT_KEY_GOES_HERE>",VmaxAdView.UX_BILLBOARD);

vmaxAdViewBillboard.cacheAd();
```

Create a container for billboard ad

Place this code snippet in the **layout** in which you want to show your Ad.

```
    <FrameLayout

      android:id="@+id/adview"

      android:layout_width="wrap_content"

      android:layout_height="wrap_content"

      android:gravity="center"

      android:background="@android:color/transparent"

      android:layout_gravity="center"

      android:visibility="gone" />
```

Show billboard ad in another Activity

Place this code snippet in the second **Activity** class to show a cached ad. Modifying the **Activty context** through the method **setBaseContext()** is necessary before calling **showAd()**.

```
AdContainer singleton = AdContainer.getInstance();

VmaxAdView vmaxAdViewBillboard= singleton.getAdView("<ADSPOT_KEY_GOES_HERE>");

vmaxAdViewBillboard.setAdListener(new VmaxAdListener() {

   @Override

   public void onAdError(VmaxAdError error, VmaxAdView adView) {
```

```
    }
    @Override
    public void onAdReady(VmaxAdView adView) {
    }
    @Override
    public void onAdClose(VmaxAdView adView)
    }
    @Override
    public void onAdMediaEnd(boolean isMediaCompleted, long reward, VmaxAdView
adView) {
    }
  });
  FrameLayout billboardContainer=(FrameLayout) findViewById(R.id.adview);
  if (vmaxAdViewBillboard.getContext() != null) {
   ((MutableContextWrapper) vmaxAdViewBillboard.getContext()).setBaseContext(this);
   }
    billboardContainer.removeAllViews();
    billboardContainer.addView(vmaxAdViewBillboard);

  if (vmaxAdViewBillboard.getAdState() == VmaxAdView.AdState.STATE_AD_READY){
    vmaxAdViewBillboard.showAd();
    billboardContainer.setVisibility(View.VISIBLE);
  }
```

Video controls customization

You can customize video controls by following this link Video layout Customization

FAQs

Q:Can I run video ads in Billboard adspace?
A: Yes, availability of video ads depends on campaigns availability.

Q: Can I Refresh Ads?
A:The option to Refresh ads isn't available when you use the advanced approach to show ads.

Integrating Interstitial Ads – JioAds SDK for Android

Enabling Interstitial Ads

This section shows how to cache and show interstitial in the same **Activity**. If you want to cache a interstitial ad in one **Activity** and show it in a different **Activity**.

Cache interstitial ads

Place this code snippet in an **Activity** class to cache interstitial ads.

```
 VmaxAdView vmaxAdViewInterstitial = new
VmaxAdView(this,"<ADSPOT_KEY_GOES_HERE>",VmaxAdView.UX_INTERSTITIAL);



          vmaxAdViewInterstitial.setAdListener(new VmaxAdListener() {

            @Override

            public void onAdReady(VmaxAdView vmaxAdView) {



            }




            @Override

            public void onAdError(VmaxAdError vmaxAdError, VmaxAdView adView) {

            }




            @Override

            public void onAdClose(VmaxAdView adView) {

            }



            @Override
```

```
            public void onAdMediaEnd(boolean isMediaCompleted, long reward,
VmaxAdView adView) {

            }



        });

        vmaxAdViewInterstitial.cacheAd();
```

Show interstital ads

Place this code snippet in an **Activity** class to show a cached ad. If you are placing **vmaxAdViewInterstitial.showAd()** inside **onAdReady()** checking for **VmaxAdView.AdState.STATE_AD_READY** isn't necessary.

```
if(vmaxAdViewInterstitial.getAdState()== VmaxAdView.AdState.STATE_AD_READY)

        {

            vmaxAdViewInterstitial.showAd();

        }
```

Customize Ad background

The below mentioned API helps you customize the background colour for interstitial ads.

```
vmaxAdView.setAdviewBackgroundColor(#COLOUR_CODE);
```

Handling VmaxAdView object for Activity life-cycle changes

Lastly, add the following code to destroy **vmaxAdView** when **Activity** is no longer available.

```
@Override

    protected void onDestroy() {

        if (vmaxAdViewInterstitial != null) {

       /** To Destroy vmaxAdView when Activity Is No Longer Available  */

            vmaxAdViewInterstitial.onDestroy();

        }

        super.onDestroy();

    }
```

FAQs

Q: How can I ensure that ads sounds don't interfere with app sounds?
A: You can pause all the app sounds before calling **showAd()** API and resume the sounds when you receive the callback **onAdClose()**.

Q: Why I am able to see buffering effect while showing ads?
A: There are two approaches to render ads one of which is to call **showAd()** API directly and another approach is to cache the ad using **cacheAd()** prior to showing the ad. The buffering effect might be observed when **showAd()** API is called directly. Hence it is recommended to use the cache + show approach and maintain some interval between them. Also a low bandwidth connection could be responsible for such a buffering effect.

Integrating Interstitial Ads (Advanced) – JioAds SDK for Android

JioAds SDK allows you to cache an Interstitial ad in one **Activity** and show it in a different **Activity**. You can also use this approach in a single **Activity** if you wish to do so.

Cache an Interstitial ad in one Activity

In order for the SDK to maintain Ad-states across **Activity** you need to create a mutable instance of **VmaxAdView** class. Place this code snippet in an **Activity** class to initialize the SDK and cache the ad.

```
 VmaxAdView vmaxAdViewInterstitial  =
VmaxAdView.getMutableInstance(this,"<ADSPOT_KEY_GOES_HERE>",VmaxAdView.UX_INTERSTITIAL);

 vmaxAdViewInterstitial.cacheAd();
```

Show an Interstitial ads in another Activity

Place this code snippet in the second **Activity** class to show a cached ad. Modifying the **Activty context** through the method **setBaseContext()** is necessary before calling **showAd()**.

```
AdContainer singleton = AdContainer.getInstance();

VmaxAdView vmaxAdViewInterstitial= singleton.getAdView("your adspot id");




vmaxAdViewInterstitial.setAdListener(new VmaxAdListener() {


   @Override

   public void onAdError(VmaxAdError error, VmaxAdView adView) {

   }

   @Override

   public void onAdReady(VmaxAdView adView) {



   }
```

```
    @Override
    public void onAdClose(VmaxAdView adView)
    }
    @Override
    public void onAdMediaEnd(boolean isMediaCompleted, long reward, VmaxAdView
adView) {
    }
  });



  if (vmaxAdViewInterstitial.getContext() != null) {
    ((MutableContextWrapper)
vmaxAdViewInterstitial.getContext()).setBaseContext(this);



    }


  if (vmaxAdViewInterstitial.getAdState() == VmaxAdView.AdState.STATE_AD_READY) {
      vmaxAdViewInterstitial.showAd();
    }
```

Customize Dynamic Interstial (Native ad) Layout

To customize the look and feel of the Dynamic Interstitial (Native Ads) follow the below process.

Native ad elements

Primary Elements

1. Icon Layout: Generally contains logo of the brand being advertised.
2. Title: Contains the title of the advertisement.
3. Sponser badge: Contains a highlighted badge which is an indicative that this item is an Ad.

4. Description: Contains the description of the advertisement
5: MediaView: Ads with video/Image content make use of mediaview or composite view (Mandatory if image elements used are other than icon)
6: CTA: A clickable view which redirects to the ad's website on click action.

Secondary Elements
1. Tagline : Additional descriptive text associated with the product or service being advertised.
2. Rating : Rating of the product being offered to the user. For example an app's rating in an app store from 0-5.
3. Downloads: Number of social ratings or "likes" of the product being offered to the user.
4. Sale Price: Sale price that can be used together with price to indicate a discounted price compared to a regular price.
5. Price: Price for product / app / in-app purchase.
6: Phone: Contains contact information.
7: Address: Contains address.
8: Display Url: Display URL for the text ad. To be used when sponsoring entity doesn't own the content.
9.Likes: Number of social ratings or "likes" of the product being offered to the user.

Note: Taggging of Icon layout **/** MediaView / Custom Image Layout **must be** on **ViewGroup** and **not View**. In case it's not tagged on **ViewGroup** then it will be discarded with runtime logs.

Create a Custom Container For Dynamic insterstitial ads

Portrait Layout
Below is a sample layout which will be used to render Dynamic Interstitial (Native ads) in portrait mode

```
<RelativeLayout

    xmlns:android="http://schemas.android.com/apk/res/android"

    android:layout_width="match_parent"

    android:layout_height="match_parent"

    android:id="@+id/vmax_native_interstitial_layout"

    >



    <RelativeLayout

        android:layout_width="match_parent"
```

```
        android:layout_height="match_parent">


        <LinearLayout
            android:id="@+id/fullscreenNativeLayout"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            android:orientation="vertical"
            android:visibility="visible">
            <FrameLayout
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:id="@+id/relativeLayout1"
                android:layout_gravity="top"
                android:layout_weight="35">


                <RelativeLayout
                    android:id="@+id/media_view"
                    android:layout_width="match_parent"
                    android:layout_height="240dp"
                    android:gravity="center"
                    android:visibility="gone"
                    android:tag="@string/vmax_native_media_layout"
                    />


                <RelativeLayout
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:layout_gravity="top"
                    android:orientation="horizontal"
                    >
```

```
                    <FrameLayout

android:tag="@string/vmax_native_adchoice_layout"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_alignParentLeft="true"
                        android:layout_marginLeft="5dp"
                        android:layout_marginTop="5dp"
                        android:id="@+id/adchoice_layout"
                        />

                    <TextView
                        android:id="@+id/vmax_ad_badge"
                        android:layout_width="20dp"
                        android:layout_height="20dp"
                        android:layout_gravity="center"
                        android:layout_marginRight="2dp"
                        android:background="@drawable/vmax_nativebg_ad"
                        android:gravity="center"
                        android:paddingLeft="5dp"
                        android:paddingRight="5dp"
                        android:layout_marginTop="5dp"
                        android:text="Ad"
                        android:textColor="#000000"
                        android:textSize="7sp"
                        android:layout_toRightOf="@id/adchoice_layout"
                        android:visibility="visible" />

                    <TextView
```

```
                        android:id="@+id/iv_close_button"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:layout_alignParentTop="true"
                        android:layout_alignParentRight="true"
                        android:text="You can Skip Ad in SKIP_COUNTER"
android:drawableRight="@drawable/vmax_close_advertisement_video"
                        android:tag="@string/vmax_native_skip_element"
                        android:textColor="@android:color/white"
                        android:layout_marginTop="5dp"
                        android:layout_marginRight="5dp"
android:textAppearance="?android:attr/textAppearanceSmall"
                        android:visibility="gone"
                        />
                </RelativeLayout>
            </FrameLayout>

            <FrameLayout
                android:layout_gravity="center"
                android:id="@+id/vmax_custom_icon"
                android:layout_width="80dp"
                android:layout_height="80dp">
                <FrameLayout
                    android:tag="@string/vmax_native_icon_layout"
                    android:id="@+id/native_icon_layout"
                    android:layout_width="match_parent"
                    android:layout_height="match_parent"/>

            </FrameLayout>
```

```
<TextView
    android:id="@+id/vmax_title"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:ellipsize="end"
    android:textColor="@color/vmax_black"
    android:textSize="15sp"
    android:textStyle="bold"
    android:layout_gravity="center"
    android:gravity="center"
    android:layout_margin="1dp"
    android:text="Title"
    android:layout_weight="5"
    android:tag="@string/vmax_native_title"
    />

<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:gravity="center_horizontal"
    android:layout_weight="20"
    android:id="@+id/linearLayout1"
    >
    <RatingBar
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/vmax_rating_bar"
        style="?android:attr/ratingBarStyleSmall"
```

```
                android:numStars="5"
                android:progressTint="#ffcc00"
                android:scaleX="1.3"
                android:scaleY="1.3"
                android:layout_gravity="center"
                android:layout_margin="6dp"
                android:visibility="gone"
                android:tag="@string/vmax_native_rating"
                />

            <LinearLayout
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:orientation="horizontal"
                android:id="@+id/vmax_like_layout"
                android:layout_margin="4dp"
                android:layout_gravity="center"
                android:visibility="gone"
                >

                <ImageView
                    android:layout_width="40dp"
                    android:layout_height="50dp"
                    android:src="@drawable/vmax_like" />
                <TextView
                    android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:layout_gravity="center"
                    android:textColor="#5d5d5d"
                    android:id="@+id/vmax_like"
```

```
                android:tag="@string/vmax_native_likes"
                />
        </LinearLayout>

        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:id="@+id/vmax_downloads_layout"
            android:layout_margin="4dp"
            android:layout_gravity="center"
            android:visibility="gone"
            >

            <ImageView
                android:layout_width="30dp"
                android:layout_height="45dp"
                android:src="@drawable/vmax_user"
                />
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:textColor="#5d5d5d"
                android:id="@+id/vmax_downloads"
                android:tag="@string/vmax_native_downloads"
                />
        </LinearLayout>

    </LinearLayout>
```

```
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical"
    android:layout_weight="10">
    <TextView
        android:id="@+id/vmax_desc"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:gravity="center"
        android:ellipsize="end"
        android:textColor="@color/vmax_black"
        android:textSize="13sp"
        android:fontFamily="sans-serif-medium"
        android:layout_marginLeft="5dp"
        android:layout_marginRight="5dp"
        android:tag="@string/vmax_native_description"
        />

    <TextView
        android:id="@+id/vmax_desc2"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:gravity="center"
        android:lines="2"
        android:ellipsize="end"
        android:textColor="@color/vmax_black"
```

```
                android:textSize="11sp"
                android:layout_marginLeft="5dp"
                android:layout_marginTop="2dp"
                android:layout_marginRight="5dp"
                android:tag="@string/vmax_native_description2"/>

        </LinearLayout>

        <LinearLayout
            android:id="@+id/linearLayout2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:gravity="center"
            android:orientation="horizontal"
            android:layout_centerHorizontal="true"
            android:layout_margin="5dp"
            android:layout_weight="12">

            <Button
                android:id="@+id/vmax_cta"
                style="@style/style_vmax_button_480"
                android:background="#008c46"
                android:layout_width="320dp"
                android:layout_height="45dp"
                android:layout_gravity="center"
                android:gravity="center"
                android:tag="@string/vmax_native_cta"
                android:text="Install Now" />
        </LinearLayout>
```

```
            <TextView
                android:id="@+id/vmax_display_url"
                android:layout_width="match_parent"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:gravity="center"
                android:ellipsize="end"
                android:textColor="@color/vmax_black"
                android:textSize="12sp"
                android:text="www.xyz.com"
                android:layout_weight="3"
                android:tag="@string/vmax_native_display_url"/>

        </LinearLayout>
    </RelativeLayout>
</RelativeLayout>
```

Landscape Layout

Below is a sample layout which will be used to render Dynamic Interstitial (Native ads) in landscape mode

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/vmax_native_interstitial_layout"
    android:layout_width="match_parent"
```

```
android:layout_height="match_parent"
android:background="@android:color/white">

<RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    >

    <FrameLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent">

        <FrameLayout
            android:layout_width="match_parent"
            android:layout_height="match_parent">

            <RelativeLayout
                android:id="@+id/media_view"
                android:layout_width="match_parent"
                android:layout_height="match_parent"
                android:visibility="gone"
                android:tag="@string/vmax_native_media_layout"
                />
        </FrameLayout>

        <RelativeLayout
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_gravity="top"
            android:gravity="center_vertical"
```

```
android:padding="5dp"

>


<FrameLayout

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:tag="@string/vmax_native_adchoice_layout"

    android:id="@+id/adchoice_layout"

    android:layout_alignParentLeft="true"

    />


<TextView

    android:id="@+id/vmax_ad_badge"

    android:layout_width="20dp"

    android:layout_height="20dp"

    android:layout_gravity="center"

    android:layout_marginRight="2dp"

    android:background="@drawable/vmax_nativebg_ad"

    android:gravity="center"

    android:paddingLeft="5dp"

    android:paddingRight="5dp"

    android:text="Ad"

    android:textColor="#000000"

    android:textSize="7sp"

    android:layout_toRightOf="@id/adchoice_layout"

    android:visibility="visible" />


<TextView

    android:id="@+id/vmax_display_url"

    android:layout_width="wrap_content"
```

```
            android:layout_height="wrap_content"
            android:ellipsize="end"
            android:textColor="@color/vmax_white"
            android:textSize="15sp"
            android:gravity="left"
            android:paddingLeft="5dp"
            android:layout_toRightOf="@id/vmax_ad_badge"
            android:tag="@string/vmax_native_display_url"
            />

        <TextView
            android:id="@+id/iv_close_button"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="You can Skip Ad in SKIP_COUNTER"
            android:drawableRight="@drawable/vmax_close_advertisement_video"
            android:tag="@string/vmax_native_skip_element"
            android:textColor="@android:color/white"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:layout_alignParentRight="true"
            android:visibility="gone" />

    </RelativeLayout>

    <RelativeLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:gravity="center_vertical"
        android:layout_gravity="bottom"
        android:background="@drawable/vmax_gradient_background"
```

```
android:paddingTop="10dp"

android:paddingBottom="10dp"

android:paddingLeft="5dp"

android:paddingRight="5dp"

>


<FrameLayout

    android:layout_gravity="center"

    android:id="@+id/vmax_custom_icon"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:layout_alignParentLeft="true"

    >

    <FrameLayout

        android:tag="@string/vmax_native_icon_layout"

        android:id="@+id/native_icon_layout"

        android:layout_width="40dp"

        android:layout_height="40dp"/>


</FrameLayout>

<LinearLayout

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:layout_alignParentRight="true"

    android:layout_centerVertical="true"

    android:orientation="vertical">

    <LinearLayout

        android:id="@+id/linearLayout2"

        android:layout_width="match_parent"

        android:layout_height="wrap_content"
```

```
android:gravity="center"
android:orientation="horizontal"
>

<RatingBar
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:id="@+id/vmax_rating_bar"
    style="?android:attr/ratingBarStyleSmall"
    android:numStars="5"
    android:progressTint="#ffcc00"
    android:scaleX="1.3"
    android:scaleY="1.3"
    android:layout_gravity="center"
    android:layout_margin="7dp"
    android:visibility="gone"
    android:tag="@string/vmax_native_rating"
    />

<LinearLayout
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:id="@+id/vmax_like_layout"
    android:layout_marginLeft="10dp"
    android:layout_gravity="center"
    android:visibility="gone"
    >
    <ImageView
        android:layout_width="25dp"
```

```
                android:layout_height="25dp"
                android:src="@drawable/vmax_like"/>
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="25dp"
                android:layout_gravity="center_vertical"
                android:gravity="center_vertical"
                android:id="@+id/vmax_like"
                android:textColor="@color/vmax_white"
                android:tag="@string/vmax_native_likes"
                />
        </LinearLayout>
        <LinearLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal"
            android:id="@+id/vmax_downloads_layout"
            android:layout_margin="5dp"
            android:layout_gravity="center"
            android:visibility="gone"
            >
            <ImageView
                android:layout_width="25dp"
                android:layout_height="25dp"
                android:src="@drawable/vmax_user"/>
            <TextView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center_vertical"
                android:gravity="center_vertical"
```

```
                    android:textColor="@color/vmax_white"
                    android:id="@+id/vmax_downloads"
                    android:tag="@string/vmax_native_downloads"
                    />
            </LinearLayout>
        </LinearLayout>
        <Button
            android:id="@+id/vmax_cta"
            android:background="#008c46"
            android:textColor="@color/vmax_white"
            android:textSize="17sp"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:gravity="center"
            android:paddingLeft="65dp"
            android:paddingRight="65dp"
            android:paddingTop="5dp"
            android:paddingBottom="5dp"
            android:text="Install Now"
            android:tag="@string/vmax_native_cta"/>
    </LinearLayout>
    <LinearLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dp"
        android:layout_toRightOf="@id/vmax_custom_icon"
        android:layout_centerVertical="true"
        android:orientation="vertical">
```

```
                <TextView
                    android:id="@+id/vmax_title"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:ellipsize="end"
                    android:textColor="@color/vmax_white"
                    android:textSize="16sp"
                    android:padding="1dp"
                    android:fontFamily="Roboto"
                    android:textStyle="bold"
                    android:gravity="center_vertical"
                    android:tag="@string/vmax_native_title"
                    />
                <TextView
                    android:id="@+id/vmax_desc"
                    android:layout_width="match_parent"
                    android:layout_height="wrap_content"
                    android:ellipsize="end"
                    android:maxWidth="300dp"
                    android:maxLines="1"
                    android:textColor="@color/vmax_white"
                    android:textSize="12sp"
                    android:padding="1dp"
                    android:fontFamily="Roboto"
                    android:gravity="center_vertical"
                    android:tag="@string/vmax_native_description"
                    />
                <TextView
                    android:id="@+id/vmax_desc2"
                    android:layout_width="match_parent"
```

```
                        android:layout_height="wrap_content"

                        android:maxWidth="300dp"

                        android:maxLines="2"

                        android:ellipsize="end"

                        android:textColor="@color/vmax_white"

                        android:textSize="10sp"

                        android:padding="1dp"

                        android:fontFamily="Roboto"

                        android:gravity="center_vertical"

                        android:tag="@string/vmax_native_description2"

                        />

                </LinearLayout>

            </RelativeLayout>

        </FrameLayout>

    </RelativeLayout>

</RelativeLayout>
```

Pass Layout to VmaxAdView Object

```
vmaxAdViewInterstitial.setLayout(R.layout.custom_interstitial_portrait,R.layout.custo
m_interstitial_landscape,Constants.AdCategory.NATIVE);
```

Video controls customization

You can customize video controls by following this link Video layout Customization

Integrating Rewarded Video Ads – JioAds SDK for Android

Enabling Rewarded Video Ads

This section explains rendering rewarded video ads programatically within the same **Activity**

To cache rewarded video ad

Place this code snippet in an **Activity** class to cache rewarded video ad.

```
  VmaxAdView vmaxAdViewRewardedVideo = new
VmaxAdView(this,"<AD_SPOT_KEY_GOES_HERE>",VmaxAdView.UX_INTERSTITIAL);

  vmaxAdViewRewardedVideo.setAdListener(new VmaxAdListener() {

   @Override

    public void onAdError(VmaxAdError error, VmaxAdView adView) {



     }

   @Override

    public void onAdReady(VmaxAdView adView) {



     }

   @Override

   public void onAdClose(VmaxAdView adView) {




    }

   @Override

   public void onAdMediaEnd(boolean isMediaCompleted, long reward, VmaxAdView adView)
{
```

```
   }
  });
  vmaxAdViewRewardedVideo.cacheAd();
```

Show rewarded video ad

Place this code snippet in an **Activity** class to show rewarded video ad. If you are placing **vmaxAdViewInterstitial.showAd()** inside **onAdReady()** checking for **VmaxAdView.AdState.STATE_AD_READY** isn't necessary.

```
if(vmaxAdViewRewardedVideo.getAdState()== VmaxAdView.AdState.STATE_AD_READY)
       {
           vmaxAdViewRewardedVideo.showAd();
       }
```

Handling Activity life-cycle changes

Lastly, add the following code to your **Activity** to enable **VMAXAdView** object to adapt to **Activity-lifecycle** changes

```
 @Override
   protected void onDestroy() {
       if (vmaxAdViewRewardedVideo != null) {
      /* To Destroy vmaxAdView when Activity Is No Longer Available  /
           vmaxAdViewRewardedVideo.onDestroy();
       }
       super.onDestroy();
   }
```

FAQs

Q: How can I ensure that ads sounds don't interfere with app sounds?
A: You can pause all the app sounds when you receive **onAdMediaStart()** callback and resume the sounds when you receive the callback **onAdClose()**.

Q: Why I am able to see buffering effect while showing ads?
A: There are two approaches to render ads one of which is to call **showAd()** API directly and another approach is to cache the ad using **cacheAd()** prior to showing the ad. The buffering effect might be observed when **showAd()** API is called directly. Hence it is recommended to use the cache + show approach and maintain some interval between them. Also a low bandwidth connection could be responsible for such a buffering effect.

Integrating Rewarded Video Ads Advanced

JioAds SDK allows you to cache a Rewarded Video ad in one **Activity** and show it in a different **Activity**.You can also use this approach in a single **Activity** if you wish to do so.

Cache Rewarded Video in one Activity

In order for the SDK to maintain Ad-states across **Activity** you need to create a mutable instance of **VmaxAdView** class Place this code snippet in an **Activity** class to initialize the SDK and cache the ad.

```
VmaxAdView vmaxAdViewRewardedVideo = VmaxAdView.getMutableInstance(this,"your adspot id",VmaxAdView.UX_INTERSTITIAL);

vmaxAdViewRewardedVideo.cacheAd();
```

Show Rewarded Video in another Activity

Place this code snippet in the second **Activity** class to show a cached ad. Modifying the **Activty context** through the method **setBaseContext()** is necessary before calling **showAd()**.

```
AdContainer singleton = AdContainer.getInstance();

VmaxAdView vmaxAdViewRewardedVideo= singleton.getAdView("<AD_SPOT_KEY_GOES_HERE>");

vmaxAdViewRewardedVideo.setAdListener(new VmaxAdListener() {

    @Override

   public void onAdError(VmaxAdError error, VmaxAdView adView) {

   }

   @Override

   public void onAdReady(VmaxAdView adView) {

   }

   @Override
```

```
    public void onAdClose(VmaxAdView adView) {

    }

    @Override

    public void onAdMediaEnd(boolean isMediaCompleted, long reward, VmaxAdView
adView) {

    }

 });

 if (vmaxAdViewRewardedVideo.getContext() != null) {

   ((MutableContextWrapper)
vmaxAdViewRewardedVideo.getContext()).setBaseContext(this);

 }

if (vmaxAdViewRewardedVideo.getAdState() == VmaxAdView.AdState.STATE_AD_READY){

    vmaxAdViewRewardedVideo.showAd();

}
```

Handling VmaxAdView object for **Activity life-cycle** changes
Lastly, add the following code to your **Activity** to enable VMAXAdView object to adapt to **Activity-lifecycle** changes

```
  @Override

    protected void onPause() {

        if (vmaxAdView != null) {

            /** To Pause Refresh Of The Ad While Activity Isn't in Foreground */

            vmaxAdView.onPause();

        }




        super.onPause();

    }
```

```
    @Override
    protected void onResume() {
        if (vmaxAdView != null) {
            /** To Resume Refresh Of The Ad While Activity Comes Back To Foreground
*/



            vmaxAdView.onResume();
        }
        super.onResume();
    }
    @Override
    protected void onDestroy() {
        if (vmaxAdView != null) {
       /** To Destroy vmaxAdView when Activity Is No Longer Available  */
            vmaxAdView.onDestroy();
        }
        super.onDestroy();
    }



configu


    }
```

Video controls customization

You can customize video controls by following this link Video layout Customization

Integrating Video In-Stream Ads – JioAds SDK for Android

Enabling In-Stream Video Ads

This section shows how to integrate instream video ads within the same activity

---

Cache In-Stream Video ads

Place this code snippet in an **Activity** class to cache in-stream video ads.

```
VmaxAdView vmaxAdViewInstreamVideo = new
VmaxAdView(this,"<ADSPOT_KEY_GOES_HERE>",VmaxAdView.UX_INSTREAM_VIDEO);

 vmaxAdViewInstreamVideo.setAdListener(new VmaxAdListener() {

     @Override

    public void onAdError(VmaxAdError error, VmaxAdView adView) {

    }

    @Override

    public void onAdReady(VmaxAdView adView) {

    }

    @Override

    public void onAdClose(VmaxAdView adView) {

    }

    @Override

    public void onAdMediaEnd(boolean isMediaCompleted, long reward, VmaxAdView
adView) {

    }

 });

 vmaxAdViewInstreamVideo.cacheAd();
```

Show In-Stream Video ads

Place this code snippet in an **Activity** class to show a cached ad.

```
Framelayout instreamAdContainer =(FrameLayout)findViewById(R.id.instreamAdContainer)



if(vmaxAdViewInstreamVideo.getAdState()== VmaxAdView.AdState.STATE_AD_READY)
        {
            vmaxAdViewInstreamVideo.setVideoPlayerDetails(instreamAdContainer);
            vmaxAdViewInstreamVideo.showAd();
        }
```

1. Use the below APIs to hide or show all of the overlay controls.

```
//to hide all the video controls
 vmaxAdViewInstreamVideo.hideControls();
 //to show all the video controls
 vmaxAdViewInstreamVideo.showControls();
```

2. Use below APIs to Expand or Collapse the In-Stream video ads.

```
//to expand the video ad
 vmaxAdViewInstreamVideo.expandAd();
 //to collapse the video ad
 vmaxAdViewInstreamVideo.collapseAd();
```

Note:

- It is recommended to use above APIs in cases like Orientation change handling, Gesture based expand / collapse
- Make sure to call above APIs in ConfigurationChanged until video ad is completed, which can be checked using below API

```
vmaxAdViewInstreamVideo.isMediaInProgress();
```

Following code snippet specifies the usage of exapandAd(), collapseAd() and isMediaInProgress() APIs in case of Orientation change

```
@Override
 public void onConfigurationChanged(Configuration newConfig) {
     if(vmaxAdViewInstreamVideo!=null &&
                     vmaxAdViewInstreamVideo.isMediaInProgress()) {
         if (newConfig.orientation == Configuration.ORIENTATION_LANDSCAPE) {
             vmaxAdViewInstreamVideo.expandAd();
         } else if (newConfig.orientation == Configuration.ORIENTATION_PORTRAIT) {
             vmaxAdViewInstreamVideo.collapseAd();
         }
     }
     super.onConfigurationChanged(newConfig);
 }
```

Use expandAd() API when the orientation is LANDSCAPE & collapseAd() API in PORTRAIT orientation by checking the media progress using isMediaInProgress() API

Handling VmaxAdView object for Activity life-cycle changes

Lastly, add the following code to pause, resume and destroy **vmaxAdView** object according to **Activity** life-cycle changes.

```
@Override

  protected void onDestroy() {

      if (vmaxAdViewInstreamVideo != null) {

          vmaxAdViewInstreamVideo.onDestroy();

      }

      super.onDestroy();

  }

@Override

  protected void onPause() {

      if (vmaxAdViewInstreamVideo != null) {

          vmaxAdViewInstreamVideo.pauseInstreamAd();

      }

      super.onPause();

  }

@Override

  protected void onResume() {

      if (vmaxAdViewInstreamVideo != null) {

          vmaxAdViewInstreamVideo.resumeInstreamAd();

      }

      super.onResume();

  }
```

Integrating Video In-Stream Ads (Advanced) – JioAds SDK for Android

JioAds SDK allows you to cache an In-Stream Video ad in one **Activity** and show it in a different **Activity**.You can also use this approach in a single **Activity** if you wish to do so.

Cache In-Stream Video in the first Activity

In order for the SDK to maintain Ad-states across **Activity** you need to create a mutable instance of **VmaxAdView** class Place this code snippet in an **Activity** class to initialize the SDK and cache the ad.

```
VmaxAdView vmaxAdViewInstreamVideo =
VmaxAdView.getMutableInstance(this,"<AD_SPOT_KEY_GOES_HERE>",VmaxAdView.UX_INSTREAM_VIDEO);

vmaxAdViewInstreamVideo.cacheAd();
```

Show In-Stream Video in another Activity

Place this code snippet in the second **Activity** class to show a cached ad. Modifying the **Activty context** through the method **setBaseContext()** is necessary before calling **showAd()**.

```
AdContainer singleton = AdContainer.getInstance();

VmaxAdView vmaxAdInstreamVideo= singleton.getAdView("<AD_SPOT_KEY_GOES_HERE>");

vmaxAdInstreamVideo.setAdListener(new VmaxAdListener() {

    @Override

   public void onAdError(VmaxAdError error, VmaxAdView adView) {

   }

   @Override

   public void onAdReady(VmaxAdView adView) {

   }

   @Override

   public void onAdClose(VmaxAdView adView) {

   }
```

```
    @Override
    public void onAdMediaEnd(boolean isMediaCompleted, long reward, VmaxAdView
adView) {
    }
 });
 if (vmaxAdInstreamVideo.getContext() != null) {
   ((MutableContextWrapper) vmaxAdInstreamVideo.getContext()).setBaseContext(this);
 }
if (vmaxAdInstreamVideo.getAdState() == VmaxAdView.AdState.STATE_AD_READY){
    vmaxAdInstreamVideo.setVideoPlayerDetails(<Your_ad_UI_container>,false);
    vmaxAdInstreamVideo.showAd();
}
```

You can choose to hide the default ad overlay controls.
Note: The Skip button in overlay controls cannot be hidden.

To hide overlay controls

```
//pass false to show the default UI controls
vmaxAdView.setVideoPlayerDetails(<Your_ad_UI_container>,true);
```

Video controls customization

You can customize video controls by following this link Video layout Customization

Note: AdCustomizer class will be deprecated soon.

Alternative approach of scheduling Ads using playlist:

The process of placing ads into video content involves two basic steps:

1) Defining the structure of the ad inventory such as the opportunities within the video content for servingads (or ad breaks), their quantity, their position in the content timeline, the number of ads allowed, and so on

2) Defining the ads to fill the ad inventory

JioAds Android SDK allows you to achieve this using IAB VMAP. VMAP defines the structure for the ad inventory (1), and VAST defines the ads that fill the inventory (2).

Currently JioAds SDK supports VMAP if used in conjuction with Google IMA SDK.

Requesting VMAP

The IAB Video Multiple Ad Playlist (VMAP) specification is an XML template that video content owners can use to describe the structure for ad inventory insertion. In order to effectively monetize video content with in-stream insertion advertising, video content owners must carefully manage the structure and use of ad inventory opportunities available within their content.

With VMAP, video content creators can define the following:

- Ad breaks within their content
- Timing for each break
- How many breaks are available

VMAP was designed to be used in conjunction with VAST and is well-suited for video content creators who have no control over the video player, but want to control the ad experience within the videos. This method of ad scheduling is supported via VAST and Google IMA plugins.

Requesting VMAP through JioAds

There are two methods to get VMAP through JioAds sdk
1) Fetch VMAP URL
2) Fetch VMAP XML

Fetch VMAP URL

Use the below code snippet to fetch VMAP URL

```
 VmaxRequest vmaxRequest = new VmaxRequest(getApplicationContext()
,"<REQUEST_KEY_GOES_HERE>",VmaxSdk.RequestType.REQUEST_TYPE_VMAP);
// It is mandatory to pass Application Context
// REQUEST_KEY is same as your VMAP id

 // helps ad server to schedule appropriate ad breaks according to the content length
 vmaxRequest.setContentVideoDuration(20); //eg. content duration set to 20 seconds


  //informs ad server regarding which video player would be used to play content video
 vmaxRequest.setContentVideoPlayer(<Content Video Player Name>,<Content Video player
```

```
version>);
   //possible values for <Content Video Player Name>
   //VmaxSdk.ContentVideoPlayer.EXO_PLAYER
   //VmaxSdk.ContentVideoPlayer.JW_PLAYER
   //VmaxSdk.ContentVideoPlayer.OTHER
   //eg. vmaxRequest.setContentVideoPlayer(VmaxSdk.ContentVideoPlayer.EXO_PLAYER,"2.5");

   //informs ad server regarding which SDK would be used to handle VMAP
  vmaxRequest.setContentVideoHandler(VmaxSdk.ContentVideoHandler.IMA_SDK,"3.7.4");



  //helps ad server to understand whether content video handler support of
   //viewability partners in order to deliver viewability enabled ads
  vmaxRequest.setSupportedViewabilityPartners(VmaxSdk.ViewabilityPartner.IAS,
VmaxSdk.ViewabilityPartner.MOAT);
  vmaxRequest.setListener(new VmaxRequestListener() {
                @Override
                public void onSuccess() {
                  String vmapUrl = vmaxRequest.getVMAPUrl();
                }



                @Override
                public void onFailure(VmaxRequestError vmaxRequestError) {
                     Log.d("vmax","Request Failure"
+"Title:"vmaxRequestError.getErrorTitle()

+"Description:"vmaxRequestError.getErrorDescription()+"ErrorCode"+vmaxRequestError.getErro
rCode());
                }
            });
```

```
String vmapURL = vmaxRequest.fetchVMAPUrl();
```

Fetch VMAP XML

Use the below code snippet to fetch VMAP raw XML

```
final VmaxRequest vmaxRequest = new VmaxRequest(getApplicationContext(),"<REQUEST_KEY_GOES_HERE>", VmaxSdk.RequestType.REQUEST_TYPE_VMAP);

// It is mandatory to pass Application Context

// REQUEST_KEY is same as your VMAP id


// helps ad server to schedule appropriate ad breaks according to the content length

vmaxRequest.setContentVideoDuration(20); //eg. content duration set to 20 seconds



//informs ad server regarding which video player would be used to play content video

vmaxRequest.setContentVideoPlayer(<Content Video Player Name>,<Content Video player version>);

 //possible values for <Content Video Player Name>

 //VmaxSdk.ContentVideoPlayer.EXO_PLAYER

 //VmaxSdk.ContentVideoPlayer.JW_PLAYER

 //VmaxSdk.ContentVideoPlayer.OTHER

 //eg. vmaxAdView.setContentVideoPlayer(VmaxSdk.ContentVideoPlayer.EXO_PLAYER,"2.5");


 //informs ad server regarding which SDK would be used to handle VMAP

vmaxRequest.setContentVideoHandler(VmaxSdk.ContentVideoHandler.IMA_SDK,"3.7.4");



 //helps ad server to understand whether content video handler support of

 //viewability partners in order to deliver viewability enabled ads
```

```
vmaxRequest.setSupportedViewabilityPartners(VmaxSdk.ViewabilityPartner.IAS,VmaxSdk.Viewabi
lityPartner.MOAT);



            vmaxRequest.setListener(new VmaxRequestListener() {
                @Override
                public void onSuccess() {
                   String vmapXML= vmaxRequest.getVMAPXml();


                }



                @Override
                public void onFailure(VmaxRequestError vmaxRequestError) {
                    Log.d("vmax","Request Failure"
+"Title:"vmaxRequestError.getErrorTitle()

+"Description:"vmaxRequestError.getErrorDescription()+"ErrorCode"+vmaxRequestError.getErro
rCode());
                }
            });


            vmaxRequest.fetchVMAPXml();
```

Advanced Settings

You can make use of api's mentioned in the below section for better ad targetting

1) Request Settings

Use the api's from the below code snippet

```
    HashMap<String,String> customData=new HashMap<>();

    customData.put("key1","value1local");

    customData.put("key2","value2local");

    vmaxRequest.setCustomData(customData);

    vmaxRequest.setSectionCategory(Section.SectionCategory.CAREERS);

    vmaxRequest.setPageCategory(Section.BUSINESS.CONSTRUCTION);

    vmaxRequest.setLanguageOfArticle("<LANGUAGE>");

    vmaxRequest.setKeyword("<KEYWORD>");//You can pass on keyword information during
ad request to have keyword targeted ads
```

2) Global Ad Settings

To know more about global ad settings follow later section of document

Note: Include code for adding Global ad settings
before **fetchVMAPXml()/fetchVMAPUrl()** api's to ensure that the ad settings data gets included in the request.

Integrating In-Stream Audio Ads – JioAds SDK for Android

Enabling In-Stream Audio Ads

Cache In-Stream Audio ads

Place this code snippet in an **Activity** class to cache in-stream audio ads.

```
       VmaxAdView vmaxAdViewInstreamAudio = new VmaxAdView(this,<YOUR_ADSPOT_KEY>,
VmaxAdView.UX_INSTREAM_AUDIO);

    vmaxAdViewInstreamAudio.setAdListener(new VmaxAdListener() {

        @Override

        public void onAdReady(VmaxAdView vmaxAdViewInstreamAudio)

        {


        }


        @Override

        public void onAdError(VmaxAdError Error, VmaxAdView adView) {


        }


        @Override

        public void onAdClose(VmaxAdView adView) {


        }


        @Override

        public void onAdMediaEnd(boolean isVideoCompleted, long reward, VmaxAdView
adView) {


        }

    });

   vmaxAdViewInstreamAudio.setRequestedBitRate(128);
```

```
vmaxAdViewInstreamAudio.setContainer(<COMPANION_CONTAINER>,<WIDTH>,<HEIGHT>);

vmaxAdViewInstreamAudio.enableMediaCaching(VmaxSdk.CacheMode.AUDIO);

vmaxAdViewInstreamAudio.cacheAd();
```

API's:

1) public void setRequestedBitRate(int bitRate)

- In order to request media file with desired Bitrate, this API can be used.
- It can be called multiple times whenever there is change in bitrate

2) public void setContainer(ViewGroup container, int width, int height)

container	Developers container where companion ad will be rendered by SDK
width	Required companion width
height	Required companion height

- SDK performs companion selection logic basis on width & height parameter passed via this API and renders appropriate companion to container
- This API can be called multiple times in a single cache-show cycle, whenever there is change in container or required sizes

Note: set width & height to -1 if size is unknown

3) public void enableMediaCaching(VmaxSdk.CacheMode cacheMode)

- In order to get selected media file downloaded and keep it ready during cacheAd() call, developer can make use of this API

Show In-Stream Audio ads

In order to achieve SDK defined audio ad experience, you can make use of below API. If you are placing **vmaxAdViewInstreamAudio.showAd()** inside **onAdReady()** checking for **VmaxAdView.AdState.STATE_AD_READY** isn't necessary.

```
if(vmaxAdViewInstreamAudio.getAdState()== VmaxAdView.AdState.STATE_AD_READY)

       {

           vmaxAdViewInstreamAudio.showAd();

       }
```

Handling VmaxAdView **object for** Activity life-cycle **changes**

Lastly, add the following code to destroy **vmaxAdView** when **Activity** is no longer available.

```
@Override

    protected void onDestroy() {

        if (vmaxAdViewInstreamAudio != null) {



            vmaxAdViewInstreamAudio.onDestroy();

        }

        super.onDestroy();

    }
```

Note:

SDK will auto pause and resume the Instream Audio Ads. In order to manually control the pause and resume of the Instream Audio *pauseInstreamAd()* and *resumeInstreamAd()* API of *VmaxAdView* needs to be called. Kindly note that when manual approach is used to pause the ad then manual resume needs to be done.

Integrating In-Stream Audio Ads Custom approach – JioAds SDK for Android

Implement Tracking Interface on your content player

For SDK to track events it requires you to implement **VmaxTrackingEventInterface** on your player object and override methods **getAdCurrentPosition ()** and **getAdDuration ()** return current position and the total audio duration respectively for eg.below snippet shows how **VmaxTrackingEventInterface** is implemented on Exoplayer.

```
import com.vmax.android.ads.common.vast.VmaxTrackingEventInterface;

public class MyExoPlayer extends SimpleExoPlayer implements VmaxTrackingEventInterface{



    protected MyExoPlayer(Context context, TrackSelector trackSelector, LoadControl
loadControl, DrmSessionManager<FrameworkMediaCrypto> drmSessionManager, int
extensionRendererMode, long allowedVideoJoiningTimeMs) {

        super(context, trackSelector, loadControl, drmSessionManager,
extensionRendererMode, allowedVideoJoiningTimeMs);

    }



    public MyExoPlayer(Context context, TrackSelector trackSelector, LoadControl
loadControl){

        this(context,trackSelector,loadControl,null,
SimpleExoPlayer.EXTENSION_RENDERER_MODE_OFF

        , ExoPlayerFactory.DEFAULT_ALLOWED_VIDEO_JOINING_TIME_MS);

    }



    @Override

    public int getAdCurrentPosition() {

        int position = (int)getCurrentPosition();

        return position;

    }
```

```
    @Override
    public int getAdDuration() {
        int duration = (int)getDuration();
        return duration;
    }
}
```

Cache In-Stream Audio ads

Place this code snippet in an **Activity** class to cache in-stream audio ads. Setting VmaxAdListener allows you to listen to **onAdReady()** event after which you can use the api **getAssets()** which returns a **JSON** object containing the assets to render the ad.

```
VmaxAdEvent vmaxCustomVastAdEvent;
       VmaxAdView vmaxAdViewInstreamAudio = new VmaxAdView(this,<YOUR_ADSPOT_KEY>,
VmaxAdView.UX_INSTREAM_AUDIO);
    vmaxAdViewInstreamAudio.setAdListener(new VmaxAdListener() {
        @Override
        public void onAdReady(VmaxAdView vmaxAdViewInstreamAudio)
        {
            JSONObject adAssets = vmaxAdViewInstreamAudio.getAssets();
            //Ad Media Url to be passed to your Audio Player
            String mediaUrl=adAssets.optString(Constants.AdElement.MEDIA_URL);
            //URL of static image resource to be attached to the Container (Object
which extends View)
            String staticResourceUrl =
              adAssets.optString(Constants.AdElement.STATIC_RESOURCE);
                                        //OR
            //URL of HTML resource to be attached to WebView
            String htmlResource =
              adAssets.optString(Constants.AdElement.HTML_RESOURCE);
            //AdEvent object to be used for notifying Ad-start and Ad-end events
            vmaxCustomVastAdEvent =
               (VmaxAdEvent)adAssets.opt(Constants.AdElement.AD_EVENTS_NOTIFIER);
// You can make use of these elements to define your own ad experience & ignore SDK
showAd() API
//Most importantly you need to use setPlayer() api for SDK to handle Ad rendition
```

```
       //Set the player where the Media will be played so that ad can be tracked
         vmaxAdViewInstreamAudio.setPlayer(obj_of_MyExoPlayer);
        }

        @Override
        public void onAdError(VmaxAdError Error, VmaxAdView adView) {

        }

        @Override
        public void onAdClose(VmaxAdView adView) {

        }

        @Override
        public void onAdMediaEnd(boolean isVideoCompleted, long reward, VmaxAdView
adView) {

        }
    });
    vmaxAdViewInstreamAudio.cacheAd();
```

Ad Playback API's

Use the **VmaxAdEvent**'s object retrived from **getAssets()** to indicate Ad Start and End events. For example, below snippet shows how to notify Ad SDK for Ad start and Ad End events inside Exoplayer's **STATE_READY**, **STATE_ENDED** playback states.

```
private ExoPlayer.EventListener eventListener = new ExoPlayer.EventListener() {

        @Override
        public void onTimelineChanged(Timeline timeline, Object manifest) {
        }
        @Override
        public void onTracksChanged(TrackGroupArray trackGroups, TrackSelectionArray
trackSelections) {}
```

```
    @Override
    public void onLoadingChanged(boolean isLoading) {}
    @Override
    public void onPlayerStateChanged(boolean playWhenReady, int playbackState) {
        switch (playbackState){
            case ExoPlayer.STATE_ENDED:
                vmaxCustomVastAdEvent.onAdEnd();
                break;
            case ExoPlayer.STATE_READY:
                vmaxCustomVastAdEvent.onAdStart();
                break;
        }
    }
    @Override
    public void onPlayerError(ExoPlaybackException error) {}
    @Override
    public void onPositionDiscontinuity() {}
};
```

Ad click event:

Use the **VmaxAdEvent** object retrived from **getAssets()** to indicate Companion click event. Pass a **View** object where companion is rendered (e.g WebView if HTMLResource is rendered or ImageView if StaticResource is rendered)

```
vmaxCustomVastAdEvent. onAdClicked(<COMPANION_VIEW>);
```

Integrating Native Ads Using Predefined Layouts – JioAds SDK for Android

Integrating Native Ads through Native Predefined layouts is a recommended method to request Native ads. We have also included an alternative method which helps you build a custom layout according to your app's components which you can find after this section.

Native Predefined Ad Layouts

1.ContentStream(300 x 250)
2.Infeed (320 x 50)
3.Native Interstitial (Fullscreen)

Add VmaxAdview to your layout

In your layout file (for example: **/res/layout/activity_main.xml**), you need to add **VmaxAdView** that will act as a container for the native ad. Include this XML block in your **layout**.

```
<com.vmax.android.ads.api.VmaxAdView

      xmlns:vmax="http://schemas.android.com/apk/res-auto"

      android:id="@+id/xml_container"

      android:layout_width="wrap_content"

      android:layout_height="wrap_content"

      android:layout_gravity="center"

      android:background="@android:color/transparent"

      android:gravity="center"

      vmax:adspotKey="<ADSPOT_KEY_GOES_HERE>"

      vmax:uxType="<UX_TYPE_GOES_HERE>"

      />
```

You can modify **vmax:uxType** to render the desired native ad format

1. ContentStream: **vmax:uxType="@integer/vmax_ux_native_content_stream"**
2.Infeed: **vmax:uxType="@integer/vmax_ux_infeed"**

3.Icon: **vmax:uxType=”@integer/vmax_ux_icon”**
4.Native Interstitial vmax:uxType=”@integer/vmax_ux_native_interstitial”

Initializing VmaxAdView object

Place this code snippet in your **Activity** class to initialize the **VmaxAdView** object to manage it for **Activity life-cycle** changes

```
VmaxAdView vmaxAdViewNative=(VmaxAdView)findViewById(R.id.xml_container);
```

Handling Activity life-cycle changes

Lastly, add the following code to destroy **vmaxAdView** when **Activity** is no longer available.

```
@Override

  protected void onDestroy() {

      if (vmaxAdViewNative != null) {



          vmaxAdViewNative.onDestroy();

      }

      super.onDestroy();

  }
```

FAQs

Q: Do I need to manage refresh?
A: No, JioAds Sdk will automatically handle refreshing of Ads.

Q: Can I customize the refresh rate for native ads?
A: The default refresh rate for native ads is set at 30 seconds to ensure a good ad yield without compromising on the user experience. You can manage the the frequency at which the native Ads are refreshed using the **vmax:refreshRate**attribute. Use the following code to set a custom refresh rate:

```
<com.vmax.android.ads.api.VmaxAdView

        xmlns:vmax="http://schemas.android.com/apk/res-auto"

        android:id="@+id/xml_container"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:layout_gravity="center"

        android:background="@android:color/transparent"

        android:gravity="center"

        vmax:adspotKey="<ADSPOT_KEY_GOES_HERE>"

        vmax:refreshRate="<REFRESH_RATE_IN_SECONDS>"

        vmax:uxType="@integer/vmax_ux_native"

        />
```

As the minimum refresh rate is 30 seconds; setting a lower value will default it to 30 seconds, unlike the value 0 which indicates that refresh is disabled.

Note:

On STB while loading native ads inside application project Activity component, back button key handling needs to be done in publisher app. Below is example usage.

```
  @Override

    public boolean dispatchKeyEvent(KeyEvent event) {

        try{

            if(Utility.getCurrentModeType(this) == Configuration.UI_MODE_TYPE_TELEVISION)
{

                int keyCode = event.getKeyCode();

                if (keyCode == KeyEvent.KEYCODE_BACK) {

                    finish();

                }

            }

        }

        catch (Exception ex)

        {}
```

```
        return super.dispatchKeyEvent(event);

    }
```

Integrating Native Ads(Programmatic Approach) – JioAds SDK for Android

Enabling Native Ads

Follow this process if you wish to request for Native ads programmatically.

Create a Container For Ads

Place this code snippet in the **layout** in which you want to show your Ad.

```
    <FrameLayout

      android:id="@+id/adview"

      android:layout_width="wrap_content"

      android:layout_height="wrap_content"

      android:gravity="center"

      android:background="@android:color/transparent"

      android:layout_gravity="center"

      android:visibility="gone" />
```

Show Native ads

Place this code snippet in an **Activity** class to show native ads.

```
 final VmaxAdView vmaxAdViewNative = new VmaxAdView(this, <ADSPOT_KEY_GOES_HERE>,
<UX_TYPE_GOES_HERE>);

 final FrameLayout NativeContainer=(FrameLayout)findViewById(R.id.adview);




  vmaxAdViewNative.setAdListener(new VmaxAdListener() {

              @Override

            public void onAdReady(VmaxAdView adView) {



            }
```

```
            @Override
            public void onAdError(VmaxAdError Error, VmaxAdView adView) {
            }



            @Override
            public void onAdClose(VmaxAdView adView) {
            }



            @Override
            public void onAdMediaEnd(boolean isMediaCompleted, long reward,
VmaxAdView adView) {
            }
        });
  NativeContainer.removeAllViews();
  NativeContainer.addView(vmaxAdViewNative);
  vmaxAdViewNative.showAd();
```

You can modify the **UX_TYPE** parameter to fetch the desired native ad layout.

1.ContentStream: **VmaxAdView.UX_NATIVE_CONTENT_STREAM**
2.Infeed: **VmaxAdView.UX_NATIVE_INFEED**
3.Icon: **VmaxAdView.UX_NATIVE_ICON**
4.Native Interstitial : **VmaxAdView.UX_NATIVE_INTERSTITIAL**

Customization

You can customize the behavior of the video element in native ad by using the below listed APIs. These are optional APIs and should be used only if your want to override the default values.

```
  //To enable or disable mute state in non fullscreen mode- default value false

  vmaxAdView.setNativeMuteStateForNonFullscreen(boolean muteState)







  //To enable or disable autoplay functionality in non fullscreen mode - default
value true

  vmaxAdView.setNativeMediaViewAutoPlayMode(boolean autoPlay)




  //to play video if setNativeMediaViewAutoPlayMode(boolean flag) is set to false

  vmaxAdView.playVmaxNativeMediaView()


 For XML Approach Use the Attribute

 vmax:videoMuteStateForNonFullscreen="<true/flase>"
```

Handling Activity life-cycle changes

Lastly, add the following code to your **Activity** to enable **VMAXAdView** object to adapt to **Activity-lifecycle** changes.

```
@Override

   protected void onDestroy() {

       if (vmaxAdViewNative != null) {



           vmaxAdViewNative.onDestroy();

       }
```

```
        super.onDestroy();

    }
```

FAQs

Q: Do I need to manage refresh?
A:No, JioAds Sdk will automatically handle refreshing of Ads.

Q: Can I customize the refresh rate for native ads?
A: The default refresh rate for native ads is set at 30 seconds to ensure a good ad yield without compromising on the user experience. You can manage the the frequency at which the native Ads are refreshed using the **setRefreshRate()**method. Use the following code to set a custom refresh rate:

```
vmaxAdViewNative.setRefreshRate(<REFRESH_RATE_IN_SECONDS>);
```

view rawNative Refresh.xml

As the minimum refresh rate is 30 seconds; setting a lower value will default it to 30 seconds, unlike the value 0 which indicates that refresh is disabled.

Note:

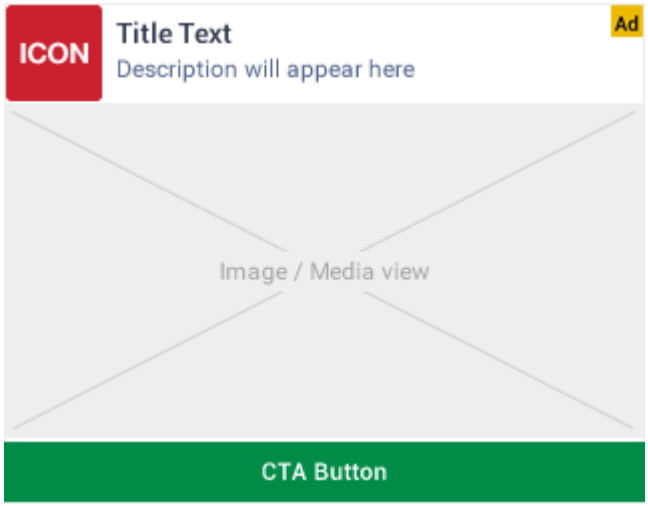
On STB while loading native ads inside application project Activity component, back button key handling needs to be done in publisher app. Below is example usage.

```
  @Override

    public boolean dispatchKeyEvent(KeyEvent event) {

        try{

            if(Utility.getCurrentModeType(this) == Configuration.UI_MODE_TYPE_TELEVISION)
{

                int keyCode = event.getKeyCode();

                if (keyCode == KeyEvent.KEYCODE_BACK) {

                    finish();

                }

            }

        }
```

```
        catch (Exception ex)

        {}

        return super.dispatchKeyEvent(event);

    }
```

Integrating Native Ads Using Custom Layout – JioAds SDK for Android

It is recommended to integrate Native Ads Using Predefined Layouts. Alternatively, you can add child views to **VmaxAdView** to customize the look and feel as desired.



Native ad elements

Primary Elements

1. Icon layout: Generally contains logo of the brand being advertised.
2. Title: Contains the title of the advertisement.
3. Sponser badge: Contains a highlighted badge which is an indicative that this item is an Ad.
4. Description: Contains the description of the advertisement
5: MediaView: Contains either video or main-image of the ad

Note: The **Image-Main** element that was used in earlier versions to display the poster image of ad is now removed and the poster image of the ad would be populated in the **MediaView** element itself.

6: CTA: A clickable view which redirects to the ad's website on click action.

Secondary Elements

1. Tagline : Additional descriptive text associated with the product or service being advertised.

2. Rating : Rating of the product being offered to the user. For example an app's rating in an app store from 0-5.
3. Downloads: Number of social ratings or "likes" of the product being offered to the user.
4. Sale Price: Sale price that can be used together with price to indicate a discounted price compared to a regular price.
5. Price: Price for product / app / in-app purchase.
6: Phone: Contains contact information.
7: Address: Contains address.
8: Display Url: Display URL for the text ad. To be used when sponsoring entity doesn't own the content.
9.Likes: Number of social ratings or "likes" of the product being offered to the user.

10.**Custom Image layout**:Contains Image of custom aspect ratio , which can be added in the 'Add custom image' option on the JioAds dashboard.

Note: Taggging of Icon layout **/** MediaView / Custom Image Layout **must be** on **ViewGroup** and **not View**. In case it's not tagged on **ViewGroup** then it will be discarded with runtime logs.

Adding child views to VmaxAdView

To render native ad you need to add below example layout to your actual ad placement layout.

```
<com.vmax.android.ads.api.VmaxAdView

  xmlns:android="http://schemas.android.com/apk/res/android"

  xmlns:vmax="http://schemas.android.com/apk/res-auto"

  android:layout_width="match_parent"

  android:layout_height="wrap_content"

  android:background="@drawable/vmax_nativeborder"

  android:layout_gravity="center"

  vmax:adspotKey="<ADSPOT_KEY_GOES_HERE>"

  vmax:uxType="@integer/vmax_ux_native"

  android:id="@+id/adView"

  android:paddingBottom="10dp"

  >
```

```
<TextView
    android:layout_marginTop="5dp"
    android:layout_toRightOf="@id/vmax_custom_icon"
    android:id="@+id/vmax_custom_title"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:ellipsize="end"
    android:maxLines="1"
    android:paddingLeft="8dp"
    android:tag="@string/vmax_native_title"
    android:paddingRight="8dp"
    android:text="Title"
    android:textColor="#000000"
    android:textSize="12sp"
    android:textStyle="bold" />


<LinearLayout
    android:id="@+id/ll_sponser"
    android:layout_alignParentRight="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:orientation="horizontal">


    <TextView
```

```
            android:id="@+id/vmax_sponsored"

            android:layout_width="15dp"

            android:layout_height="15dp"

            android:layout_gravity="center"

            android:layout_marginRight="2dp"

            android:background="@drawable/vmax_nativebg_ad"

            android:gravity="center"

            android:paddingLeft="2dp"

            android:paddingRight="2dp"

            android:text="Ad"

            android:textColor="#000000"

            android:textSize="8sp"

            android:visibility="visible" />

        <FrameLayout

            android:id="@+id/vmax_custom_adChoiceView"

            android:layout_width="15dp"

            android:layout_height="15dp"

            android:layout_gravity="center"

            android:tag="@string/vmax_native_adchoice_layout"

            android:orientation="vertical"

            android:gravity="top|right"

            android:paddingLeft="1dp"

            android:paddingRight="1dp"

            android:visibility="visible">



        </FrameLayout>

    </LinearLayout>
```

```
<TextView

    android:layout_toRightOf="@id/vmax_custom_icon"

    android:layout_below="@id/vmax_custom_title"

    android:id="@+id/vmax_custom_desc"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:tag="@string/vmax_native_description"

    android:layout_marginTop="1sp"

    android:ellipsize="end"

    android:paddingLeft="8dp"

    android:paddingRight="8dp"

    android:text="Description"

    android:textColor="#000000"

    android:textSize="8sp" />



<TextView

    android:layout_toRightOf="@id/vmax_custom_icon"

    android:layout_below="@id/vmax_custom_desc"

    android:id="@+id/flurry_advertiser_name"

    android:layout_width="wrap_content"

    android:layout_height="wrap_content"

    android:tag="@string/vmax_advertiser_name"

    android:layout_marginTop="1sp"

    android:ellipsize="end"

    android:paddingLeft="8dp"

    android:paddingRight="8dp"

    android:text="Flurry advertiser name"

    android:textColor="#000000"

    android:textSize="8sp" />
```

```
<RelativeLayout
    android:gravity="center"
    android:id="@+id/rl_media"
    android:layout_below="@id/vmax_custom_icon"
    android:layout_width="match_parent"
    android:layout_height="wrap_content">

    <RelativeLayout
        android:id="@+id/vmax_custom_media_view"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:tag="@string/vmax_native_media_layout"
        android:layout_gravity="bottom|center"
        android:gravity="bottom|center"
        />

</RelativeLayout>
<Button
    android:id="@+id/vmax_custom_cta"
    android:layout_below="@id/rl_media"
    android:layout_width="match_parent"
    android:tag="@string/vmax_native_cta"
    android:layout_height="wrap_content"
    android:layout_gravity="center"
```

```
        android:lines="1"

        android:paddingLeft="1dp"

        android:paddingRight="1dp"

        android:text="Install Now"

        android:textAlignment="center" />



    <ImageView

        android:src="@mipmap/ic_launcher"

        android:id="@+id/vmax_custom_icon"

        android:layout_width="60dp"

        android:layout_height="60dp"

        android:tag="@string/vmax_native_icon"

        android:layout_alignParentTop="true"

        android:layout_alignParentLeft="true"

        android:layout_alignParentStart="true" />



    <!---------Optional Secondary Elements----------->

    <TextView

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:tag="@string/vmax_native_tagline"

        android:textColor="#000000"

        android:textSize="8sp" />

    <TextView

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:tag="@string/vmax_native_rating"

        android:textColor="#000000"
```

```
        android:textSize="8sp" />

    <TextView

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:tag="@string/vmax_native_downloads"

        android:textColor="#000000"

        android:textSize="8sp" />

    <TextView

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:tag="@string/vmax_native_sale_price"

        android:textColor="#000000"

        android:textSize="8sp" />

    <TextView

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:tag="@string/vmax_native_likes"

        android:textColor="#000000"

        android:textSize="8sp" />

    <TextView

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:tag="@string/vmax_native_price"

        android:textColor="#000000"

        android:textSize="8sp" />

    <TextView

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:tag="@string/vmax_native_phone"

        android:textColor="#000000"
```

```
        android:textSize="8sp" />

    <TextView

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:tag="@string/vmax_native_address"

        android:textColor="#000000"

        android:textSize="8sp" />

    <TextView

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:tag="@string/vmax_native_display_url"

        android:textColor="#000000"

        android:textSize="8sp" />


</com.vmax.android.ads.api.VmaxAdView>
```

you can modify the positions of the child views to match your UI.

Attribute usage

1. android:tag : This is mandatory attribute for JioAds SDK to understand what native ad elements to map to tagged subview.

Primary Elements

a. Icon layout: Include attribute **android:tag=”@string/vmax_native_icon_layout”**
b. Title: Include attribute **android:tag=”@string/vmax_native_title”**
c. Sponser badge: Include attribute **android:tag=”@string/vmax_native_adchoice_layout”**
d. Description: Include attribute **android:tag=”@string/vmax_native_description”**
e. Advertiser Name: Include attribute **android:tag=”@string/vmax_advertiser_name”(Required only for Flurry native ads)**
f: MediaView: Include attribute **android:tag=”@string/vmax_native_media_layout”**
g: CTA: Include attribute **android:tag=”@string/vmax_native_cta”**

Secondary Elements

a. Tagline: Include attribute **android:tag="@string/vmax_native_tagline"**.
b. Rating : Include attribute **android:tag="@string/vmax_native_rating"**.
c. Downloads: Include attribute **android:tag="@string/vmax_native_downloads"**.
d. Sale Price: Include attribute **android:tag="@string/vmax_native_sale_price"**.
e. Price: Include attribute **android:tag="@string/vmax_native_price"**.
f: Phone: Include attribute **android:tag="@string/vmax_native_phone"**.
g: Address: Include attribute **android:tag="@string/vmax_native_address"**.
h: Display Url: Include attribute **android:tag="@string/vmax_native_display_url"**.
i.Likes: Include attribute **android:tag="@string/vmax_native_likes"**.

j.Custom Image Layout: Include attribute **android:tag="@string/vmax_native_custom_image_layout"**.

Note: Taggging of Icon layout **/** MediaView / Custom Image Layout **must be** on **ViewGroup** and **not View**. In case it's not tagged on **ViewGroup** then it will be discarded with runtime logs.

2. android:clickable : Optional attribute to specify whether subview should be clickable. In case none of the subviews are marked with this attribute then it will make the subview for call to action(CTA) clickable by default.

3. android:contentDescription : This optional attribute will allow to set fallback element for image assets. For eg. you may choose to fill your ImageView with other native ad image element(imageMedium) in case primary image element(imageMain) fails.

Initializing VmaxAdView object

Place this code snippet in an **Activity** class to initialize the **VmaxAdView** object to manage it for **Activity life-cycle** changes.

```
VmaxAdView vmaxAdViewNative=(VmaxAdView)findViewById(R.id.adView);
```

view rawInitializing native non-helper.java

Handling Activity life-cycle changes

Lastly, add the following code to destroy **vmaxAdView** when **Activity** is no longer available.

```
@Override
  protected void onDestroy() {
      if (vmaxAdViewNative != null) {
```

```
            vmaxAdViewNative.onDestroy();
        }
        super.onDestroy();
    }
```

view rawNative OnDestroy.java

Video controls customization

You can customize video controls by following this link Video layout Customization

FAQs

Q: Do I need to manage refresh?
A: No, JioAds Sdk will automatically handle refreshing of ads.

Q: Can I customize the refresh rate for native ads?
A: The default refresh rate for native ads is set at 30 seconds to ensure a good ad yield without compromising on the user experience. You can manage the the frequency at which the native Ads are refreshed using the **vmax:refreshRate**attribute. Use the following code to set a custom refresh rate:

```
<com.vmax.android.ads.api.VmaxAdView
        xmlns:vmax="http://schemas.android.com/apk/res-auto"
        android:id="@+id/xml_container"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:background="@android:color/transparent"
        android:gravity="center"
        vmax:adspotKey="<ADSPOT_KEY_GOES_HERE>"
        vmax:refreshRate="<REFRESH_RATE_IN_SECONDS>"
        vmax:uxType="@integer/vmax_ux_native"
```

```
        />
```

view rawNative Refresh.xml

As the minimum refresh rate is 30 seconds; setting a lower value will default it to 30 seconds, unlike the value 0 which indicates that refresh is disabled.

Q: Can I get access to image assets URLs which I can download on my own?
A: No, JioAds Sdk will automatically manage all downloads.

Q: I keep getting OutOfMemory error while loading native ads. How do I fix this?
A:If this error is faced during run-time it might be due to unwanted extra object creation. If this error is faced at compile time You can increase the heap size for compilation to a larger value in **gradle.properties** file.

```
org.gradle.jvmargs=-Xmx1024m
```

Note:

On STB while loading native ads inside application project Activity component, back button key handling needs to be done in publisher app. Below is example usage.

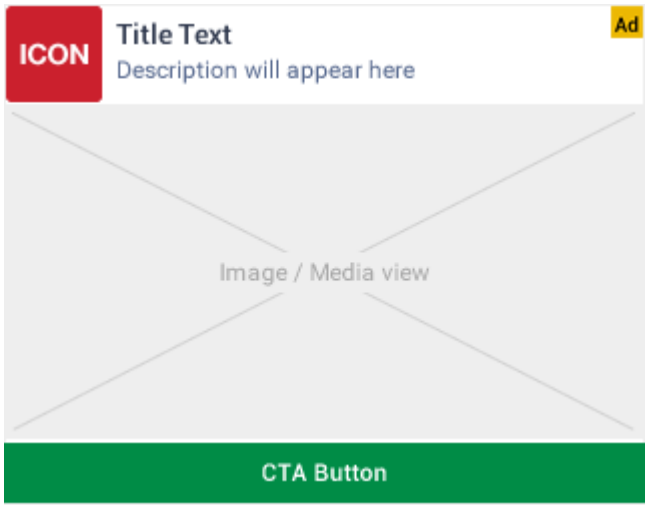
```
  @Override
    public boolean dispatchKeyEvent(KeyEvent event) {
        try{
            if(Utility.getCurrentModeType(this) == Configuration.UI_MODE_TYPE_TELEVISION)
{
                int keyCode = event.getKeyCode();
                if (keyCode == KeyEvent.KEYCODE_BACK) {
                    finish();
                }
            }
        }
        catch (Exception ex)
```

```
        {}

        return super.dispatchKeyEvent(event);

    }
```

Integrating Native Ads(Custom Layout) Programmatic approach – JioAds SDK for Android

Enabling Native Ads

Follow this process if you wish to request for Native Custom ads programmatically.



Native ad elements

Primary Elements

1 Icon layout: Generally contains logo of the brand being advertised.
2. Title: Contains the title of the advertisement.
3. Sponser badge: Contains a highlighted badge which is an indicative that this item is an Ad.
4. Description: Contains the description of the advertisement
5: MediaView: Contains either video or main-image of the ad (

Note: The **Image-Main** element that was used in earlier versions to display the poster image of ad is now deprecated and the poster image of the ad would be populated in the **MediaView** element itself.

6: CTA: A clickable view which redirects to the ad's website on click action.

Secondary Elements

1. Tagline : Additional descriptive text associated with the product or service being advertised.
2. Rating : Rating of the product being offered to the user. For example an app's rating in an app store from 0-5.

3. Downloads: Number of social ratings or “likes” of the product being offered to the user.
4. Sale Price: Sale price that can be used together with price to indicate a discounted price compared to a regular price.
5. Price: Price for product / app / in-app purchase.
6: Phone: Contains contact information.
7: Address: Contains address.
8: Display Url: Display URL for the text ad. To be used when sponsoring entity doesn’t own the content.
9.Likes: Number of social ratings or “likes” of the product being offered to the user.

10.**Custom Image Layout**:Contains Image of custom aspect ratio , which can be added in the ‘Add custom image’ option on the JioAds dashboard.

Note: Taggging of Icon layout **/** MediaView / Custom Image Layout **must be** on **ViewGroup** and **not View**. In case it’s not tagged on **ViewGroup** then it will be discarded with runtime logs.

Create a Custom Container For Native Ads

Create an xml file named vmax_prog_custom_native_layout.xml and place the below example snippet in it. You can adjust the positions, height/width of the sub-views according to your needs

```
<RelativeLayout

    xmlns:android="http://schemas.android.com/apk/res/android"

    xmlns:vmax="http://schemas.android.com/apk/res-auto"

    android:layout_width="match_parent"

    android:layout_height="wrap_content"

    android:background="@drawable/vmax_nativeborder"

    android:layout_gravity="center"

    android:id="@+id/adView"

    android:paddingBottom="10dp"

    android:visibility="gone">




    <ImageView

        android:src="@mipmap/ic_launcher"

        android:id="@+id/vmax_custom_icon"
```

```
        android:layout_width="60dp"
        android:layout_height="60dp"
        android:tag="@string/vmax_native_icon"
        />


         <TextView
            android:layout_marginTop="5dp"
            android:layout_toRightOf="@id/vmax_custom_icon"
            android:id="@+id/vmax_custom_title"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:ellipsize="end"
            android:maxLines="1"
            android:paddingLeft="8dp"
            android:tag="@string/vmax_native_title"
            android:paddingRight="8dp"
            android:text="Title"
            android:textColor="#000000"
            android:textSize="12sp"
            android:textStyle="bold" />



        <LinearLayout
            android:id="@+id/ll_sponser"
            android:layout_alignParentRight="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal">
```

```
        <TextView
            android:id="@+id/vmax_sponsored"
            android:layout_width="15dp"
            android:layout_height="15dp"
            android:layout_gravity="center"
            android:layout_marginRight="2dp"
            android:background="@drawable/vmax_nativebg_ad"
            android:gravity="center"
            android:paddingLeft="2dp"
            android:paddingRight="2dp"
            android:text="Ad"
            android:textColor="#000000"
            android:textSize="8sp"
            android:visibility="visible" />



        <FrameLayout
            android:id="@+id/vmax_custom_adChoiceView"
            android:layout_width="15dp"
            android:layout_height="15dp"
            android:layout_gravity="center"
            android:tag="@string/vmax_native_adchoice_layout"
            android:orientation="vertical"
            android:gravity="top|right"
            android:paddingLeft="1dp"
            android:paddingRight="1dp"
            android:visibility="visible">
```

```
            </FrameLayout>
        </LinearLayout>



        <TextView
            android:layout_toRightOf="@id/vmax_custom_icon"
            android:layout_below="@id/vmax_custom_title"
            android:id="@+id/vmax_custom_desc"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:tag="@string/vmax_native_description"
            android:layout_marginTop="1sp"
            android:ellipsize="end"
            android:lines="2"
            android:paddingLeft="8dp"
            android:paddingRight="8dp"
            android:text="Description"
            android:textColor="#000000"
            android:textSize="8sp" />



 <RelativeLayout
   android:gravity="center"
   android:id="@+id/rl_media"
   android:layout_below="@id/vmax_custom_icon"
   android:layout_width="match_parent"
   android:layout_height="wrap_content">
```

```
    <RelativeLayout
        android:id="@+id/vmax_custom_media_view"
        android:layout_width="match_parent"
        android:layout_height="200dp"
        android:tag="@string/vmax_native_media_layout"
        android:layout_gravity="bottom|center"
        android:gravity="bottom|center"
        />

</RelativeLayout>
   <Button
        android:id="@+id/vmax_custom_cta"
        android:layout_below="@id/rl_media"
        android:layout_width="match_parent"
        android:tag="@string/vmax_native_cta"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:lines="1"
        android:paddingLeft="1dp"
        android:paddingRight="1dp"
        android:text="Install Now"
        android:textAlignment="center" />


       <!---------Optional Secondary Elements----------->
   <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:tag="@string/vmax_native_tagline"
        android:textColor="#000000"
        android:textSize="8sp" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:tag="@string/vmax_native_rating"
        android:textColor="#000000"
        android:textSize="8sp" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:tag="@string/vmax_native_downloads"
        android:textColor="#000000"
        android:textSize="8sp" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:tag="@string/vmax_native_sale_price"
        android:textColor="#000000"
        android:textSize="8sp" />
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:tag="@string/vmax_native_likes"
        android:textColor="#000000"
        android:textSize="8sp" />
    <TextView
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"

        android:tag="@string/vmax_native_price"

        android:textColor="#000000"

        android:textSize="8sp" />

    <TextView

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:tag="@string/vmax_native_phone"

        android:textColor="#000000"

        android:textSize="8sp" />

    <TextView

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:tag="@string/vmax_native_address"

        android:textColor="#000000"

        android:textSize="8sp" />

    <TextView

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:tag="@string/vmax_native_display_url"

        android:textColor="#000000"

        android:textSize="8sp" />

  </RelativeLayout>
```

view rawNative Custom adcontainer.xml

Note: a) The visibility of the parent layout should be kept **android:visibility=”gone”**by default, SDK will automatically make it visible when the ad is rendered.

Attribute usage

1. android:tag : This is mandatory attribute for JioAds SDK to understand what native ad elements to map to tagged subview.

Primary Elements

a. Icon layout: Include attribute **android:tag="@string/ vmax_native_icon_layout"**
b. Title: Include attribute **android:tag="@string/vmax_native_title"**
c. Sponser badge: Include attribute **android:tag="@string/vmax_native_adchoice_layout"**
d. Description: Include attribute **android:tag="@string/vmax_native_description"**
e. Advertiser Name: Include attribute **android:tag="@string/vmax_advertiser_name"(Required only for Flurry native ads)**
f: MediaView: Include attribute **android:tag="@string/vmax_native_media_layout"**
g: CTA: Include attribute **android:tag="@string/vmax_native_cta"**

Secondary Elements

a. Description 2: Include attribute **android:tag="@string/vmax_native_tagline"**.
b. Rating : Include attribute **android:tag="@string/vmax_native_rating"**.
c. Downloads: Include attribute **android:tag="@string/vmax_native_downloads"**.
d. Sale Price: Include attribute **android:tag="@string/vmax_native_sale_price"**.
e. Price: Include attribute **android:tag="@string/vmax_native_price"**.
f: Phone: Include attribute **android:tag="@string/vmax_native_phone"**.
g: Address: Include attribute **android:tag="@string/vmax_native_address"**.
h: Display Url: Include attribute **android:tag="@string/vmax_native_display_url"**.
i. Likes: Include attribute **android:tag="@string/vmax_native_likes"**.

j. Custom Image Layout: Include attribute **android:tag="@string/ vmax_native_custom_image_layout"**.

Note: Taggging of Icon layout **/** MediaView / Custom Image Layout **must be** on **ViewGroup** and **not View**. In case it's not tagged on **ViewGroup** then it will be discarded with runtime logs.

2. android:clickable : Optional attribute to specify whether subview should be clickable. In case none of the subviews are marked with this attribute then it will make the subview for call to action(CTA) clickable by default.

3. android:contentDescription : This optional attribute will allow to set fallback element for image assets. For eg. you may choose to fill your ImageView with other native ad image element(imageMedium) in case primary image element(imageMain) fails.

Create a Container For Ads

Place this code snippet in the **layout** in which you want to show your Ad.

```
    <FrameLayout
      android:id="@+id/adview"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:gravity="center"
      android:background="@android:color/transparent"
      android:layout_gravity="center"
      android:visibility="gone" />
```

view rawProgrammatic Container.xml

Show Native ads

Place this code snippet in an **Activity** class to show native ads.

```
final VmaxAdView vmaxAdViewNative = new VmaxAdView(this, <ADSPOT_KEY_GOES_HERE>, VmaxAdView.UX_NATIVE);
final FrameLayout nativeContainer=(FrameLayout)findViewById(R.id.adview);
RelativeLayout nativeCustomContainer = (RelativeLayout) LayoutInflater.from(this).inflate(R.layout.vmax_p



 vmaxAdViewNative.setAdListener(new VmaxAdListener() {
             @Override
            public void onAdReady(VmaxAdView adView) {


            }



            @Override
            public void onAdError(VmaxAdError Error, VmaxAdView adView) {
            }
```

```
            @Override
            public void onAdClose(VmaxAdView adView) {
            }


            @Override
            public void onAdMediaEnd(boolean isMediaCompleted, long reward, VmaxAdView adView) {
            }
        });
    vmaxAdViewNative.setCustomNativeAdContainer(nativeCustomContainer);
    vmaxAdViewNative.setCompositeAdSize(300,250);









    nativeContainer.removeAllViews();
    nativeContainer.addView(vmaxAdViewNative);
    vmaxAdViewNative.showAd();
```

view rawNative Custom.java

Customization

You can customize the behavior of the video element in native ad by using the below

listed APIs. These are optional APIs and should be used only if your want to override the default values.

```
//To enable or disable mute state in non fullscreen mode- default value false

vmaxAdView.setNativeMuteStateForNonFullscreen(boolean muteState)




//To enable or disable autoplay functionality in non fullscreen mode - default value true

vmaxAdView.setNativeMediaViewAutoPlayMode(boolean autoPlay)




//to play video if setNativeMediaViewAutoPlayMode(boolean flag) is set to false

vmaxAdView.playVmaxNativeMediaView()
```

view rawNative MediaView.java

Handling Native video playback (pause/resume)

Pausing native video ad: Native video ad can be paused using API on VmaxAdView object as shown below:

```
If (vmaxAdViewNative!=null){
   vmaxAdViewNative.pauseNativeVideoAd();
}
```

Resume native video ad: Paused native video ad can be resumed using API on VmaxAdView object as shown below:

```
If (vmaxAdViewNative!=null){
   vmaxAdViewNative.resumeNativeVideoAd();
}
```

Handling Activity life-cycle changes

Lastly, add the following code to your **Activity** to enable **VMAXAdView** object to adapt to **Activity-lifecycle** changes.

```
@Override
   protected void onDestroy() {
       if (vmaxAdViewNative != null) {


           vmaxAdViewNative.onDestroy();
       }
       super.onDestroy();
   }
```

Video controls customization

You can customize video controls by following this link Video layout Customization

Note:

On STB while loading native ads inside application project Activity component, back button key handling needs to be done in publisher app. Below is example usage.

```
 @Override
   public boolean dispatchKeyEvent(KeyEvent event) {
       try{
           if(Utility.getCurrentModeType(this) == Configuration.UI_MODE_TYPE_TELEVISION)
{
               int keyCode = event.getKeyCode();
               if (keyCode == KeyEvent.KEYCODE_BACK) {
                   finish();
               }
           }
       }
       catch (Exception ex)
       {}
       return super.dispatchKeyEvent(event);
```

```
}
```

FAQs

Q: Do I need to manage refresh?
A:No, JioAds Sdk will automatically handle refreshing of Ads.

Q: Can I customize the refresh rate for native ads?
A: The default refresh rate for native ads is set at 30 seconds to ensure a good ad yield without compromising on the user experience. You can manage the the frequency at which the native Ads are refreshed using the **setRefreshRate()**method. Use the following code to set a custom refresh rate:

```
vmaxAdViewNative.setRefreshRate(<REFRESH_RATE_IN_SECONDS>);
```

As the minimum refresh rate is 30 seconds; setting a lower value will default it to 30 seconds, unlike the value 0 which indicates that refresh is disabled.

Ad Events

Ads have events which are exposed to the developer using callbacks, for eg. a rewarded video ad gives a **onAdMediaStart** callback when the ad media starts, hence this is an indicative for the developer to pause all the media being played in the background etc. To handle in app behavior according to Ad-lifecycle events like optionally you need to set a **VmaxAdListener** to the **VmaxAdView** object

```
  vmaxAdView.setAdListener(new VmaxAdListener() {

          @Override

          public void onAdReady(VmaxAdView adView) {

          }

          @Override

          public void onAdError(VmaxAdError error, VmaxAdView adView) {

          }

          @Override

          public void onAdClose(VmaxAdView adView) {

          }

          @Override

          public void onAdMediaEnd(boolean isMediaCompleted, long reward, VmaxAdView
adView) {

          }

          @Override

          public void onAdMediaStart(VmaxAdView adView) {

          }

          @Override

          public void onAdMediaExpand(VmaxAdView adView) {

          }

          @Override

          public void onAdMediaCollapse(VmaxAdView adView) {

          }

          @Override

          public void onAdView(int adStatus, VmaxAdView adView) {

          }
```

```
          @Override
          public void onAdClick(VmaxAdView adView) {
          }
          @Override
          public void onAdRefresh(VmaxAdView adView) {
          }
          @Override
          public void onAdSkippable(VmaxAdView adView) {
          @Override
          public void onAdReceived(VmaxAdView adView) {
          @Override
          public void onAdMediaProgress(int positionInMillis, int
totalDurationInMills, VmaxAdView adView) {
          }
```

view rawVmaxAdListener Callbacks.java

onAdReady(VmaxAdView adView)

This callback is received when Ad is ready to render state. For ad formats like banner/billboard/native where SDK is managing refresh cycle this callback will be fired only till ad-space is filled with ad for the first time. For subsequent refresh **onAdRefresh()** callback will be called.

You are expected to call **showAd()** API after this event. In-case **showAd()** API is being called outside this event then you are expected to check **AdState** as below.

```
 if(vmaxAdView.getAdState()== VmaxAdView.AdState.STATE_AD_READY)
       {
           vmaxAdView.showAd();
       }
```

view rawShow Ad.java

onAdError(VmaxAdError error, VmaxAdView adView)

This callback is received when the SDK fails to fetch/render an ad. Use the below mentioned APIs on the **VmaxAdError** object to obtain more information about the error.

```
error.getErrorTitle();

error.getErrorDescription();

error.getErrorCode();
```

view rawError.java

Below are the possible values for the error code.

1001 – ERROR_NOFILL
1002 – ERROR_ADREQUEST_NOT_ALLOWED
1003 – ERROR_MANIFEST_ENTRY_MISSING
1004 – ERROR_TIMEOUT
1005 – ERROR_INTERNAL_SERVER
1006 – ERROR_SDK_INITIALIZATION
1007 – ERROR_MISMATCHUX_OR_MARKUP
1008 – ERROR_NETWORK_ERROR
1009 – ERROR_MANDATORY_PARAM_MISSING
1010 – ERROR_RENDITION_ERROR
1011 – ERROR_PARSING
1012 – ERROR_UNKNOWN
1013 – ERROR_INVALID_REQUEST_ARGUMENTS
1014 – ERROR_AD_EXPIRED
1015 – ERROR_AUTO_CLOSED

onAdClose(VmaxAdView vmaxAdView)

This callback is received when ad is closed for full-screen ads. You are expected to resume your application animation loops/in-game sounds etc. It is recommended to cache next ad after this event.

onAdView(int visibility, VmaxAdView adView)

This callback is received when ad enters or exits the viewport i.e ad is visible to the user or not. You can check whether ad is in viewport with the below mentioned code.

```
if(adVisiblity==Constants.AdVisiblity.VISIBLE)

 {

  // You are expected to pause your animation loops/in-game sounds etc.

 }



  if(adVisiblity==Constants.AdVisiblity.INVISIBLE)
```

```
    {

      //  Ad has exited the viewport

    }
```

view rawAdVisiblity.java

onAdClick(VmaxAdView adView)

This callback is received when the ad is clicked. This will cause your app to go in pause state.

onAdRefresh(VmaxAdView adView)

This callback is applicable only for ad formats banner/billboard/native where SDK is managing refresh cycle. It will be triggered for every subsequent ad rotated inside ad-space.

onAdSkippable(VmaxAdView adView)

It will be triggered after a particular interval after which the next ad request can be made.

Note: Below callback is applicable only for the ads that has associated media(audio/video)

onAdMediaStart(VmaxAdView adView)

It will be triggered when media starts playing.

onAdMediaEnd(boolean isCompleted, long reward, VmaxAdView adView)

It will be triggered when media ends playing.
boolean isCompleted: Indicator if the media is completed or not.
long reward: Applicable only for Rewarded videos. It will expose reward amount to be gratified to the user on completing the media(can be checked using isCompleted flag).

onAdRender(VmaxAdView adView)

This callback will be triggered when the ad is filled in the ad-space for the first time.

onAdMediaExpand(VmaxAdView adView)

It will be triggered when media on a native ad or a billboard ad is expanded to full-screen view.

onAdMediaCollapse(VmaxAdView adView)

It will be triggered when media on a native ad or a billboard ad is brought back to normal view from full-screen view.

onAdReceived(VmaxAdView adView)

This callback is received as soon as SDK fetched ad data and ad is ready to be shown without caching.

onAdMediaProgress(int positionInMillis, int totalDurationInMillis,VmaxAdView adView)

This callback is received for Audio ads. It gives current position and total duration at every tick of audio playback

AdView States

A VmaxAdView object maintains ad view state information in an enum called AdViewState which can be queried as shown in below example

```
  if(adView.getState() == VmaxAdView.AdViewState.STATE_INVIEW) {
//Do something
}
//Possible values for AdViewState
//STATE_INSTANTIATED (Set by SDK when VmaxAdView object is created)
//STATE_INVIEW  (Set by SDK when Ad is rendered)
```

Ad States

A VmaxAdView object maintains ad state information in an enum called AdState which can be queried as shown in below example

```
 if(adView.getAdState() == VmaxAdView.AdState.STATE_READY) {
//Do something
}
//Possible values for AdState
STATE_AD_NOT_REQUESTED
STATE_AD_REQUESTED
STATE_AD_READY
STATE_AD_ERROR
STATE_AD_STARTED
```

```
STATE_AD_END
STATE_AD_DISMISSED
STATE_AD_EXPAND
STATE_AD_COLLAPSED
STATE_AD_INTERACTED
```

Testing the Integration –JioAds SDK for Android

The SDK Integration can be tested by either setting your integration to test mode for certain devices before requesting ads.

This section shows how to test your integration with JioAds test ads.

Configure Logs

JioAds SDK allows you to enable logs for debugging. Logs with ERROR, WARNING and INFO log levels would be always displayed by default. To enable or disable DEBUG level logs you can use the below mentioned API.

```
 VmaxSdk.getInstance().setLogLevel(LogLevel logLevel);

//Possible values for logLevel

// VmaxSdk.LogLevel.NONE (Default)-To disable debug logs

// VmaxSdk.LogLevel.DEBUG -To enable debug logs
```

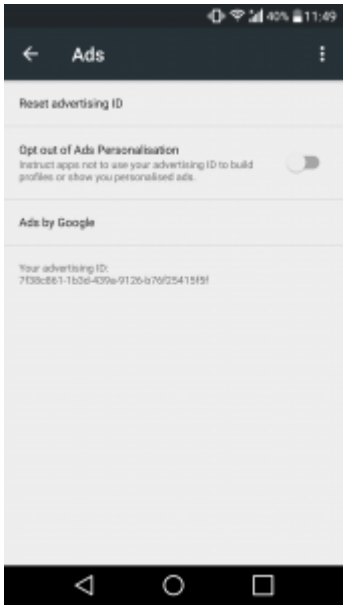
view rawTestmode API Options.java

Note: If you add test devices DEBUG logs will be enabled by default for the added device

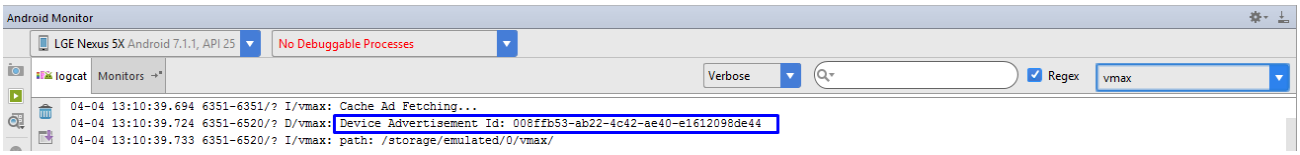
How to fetch your device's Advertiser Id

There are two ways through which you can find device's advertiser id

1.To fetch device's AdvId you can check device's Google settings under ads subMenu



2.You can run your app and navigate to the page which contains ad-request code and check the logs for 'Device Advertiser Id' under the tag vmax.



Global Ad Settings – JioAds SDK for Android

The **VmaxSdk** class provides global settings through which you can enable various features.

GDPR Compliance

JioAds is committed to comply with the European Union's new General Data Protection Regulation (GDPR).This guide will provide information for publishers to be compliant with the regulation. GDPR takes effect starting on May 25, 2018. It's a set of rules designed to give EU citizens more control over their personal data. Any businesses established in the EU or with users based in Europe are required to comply with GDPR or risk facing heavy fines. Publishers requires take the consent

from its user in order to show Personalized ads to user which results in Higher Revenue.

The app Publisher will have to pass this consent value to our SDK. We have provided API in JioAds SDK which check whether user is from EU economic area or not. If User belongs to EEA then it is must for publisher to pass consent to JioAds SDK for that user. Follow the below mentioned steps to pass user consent to JioAds SDK.

1. Check whether GDPR is applicable or not

```
VmaxSdk.getInstance().isUserInEEA(this, new RegionCheckListener() {

                        @Override

                        public void onSuccess(boolean GdprApplicable) {

                            if(GdprApplicable)

                            {

                                //Show Consent Dailog to user

                                //pass the user consent to sdk

                            }

                        }



                        @Override

                        public void onFailure(int errorCode) {



                        }

                    });
```

2. Show Consent Dialog

A consent dialog needs to be shown to the user in order to acquire consent and pass it to the SDK , you can refer our sample project which contains a similar dialog.

3.Set user consent

```
VmaxSdk.getInstance().setUserConsent(boolean isUserInEEA, boolean userConsent);
```

```
// possible values for isUserInEEA true: if user falls under EEA.

//                                  false: if user doesn't fall under EEA.

// possible values for userConsent true: if user accepts to get personalized ads.
```

Targeting

1. Passing Custom Data to all ad requests

```
HashMap<String,String> customData=new HashMap<>();

customData.put("key1","value1");

customData.put("key2","value2");

VmaxSdk.getInstance().setCustomData(customData);
```

view rawGlobal CustomData.java

Note:

1. The above api would apply on all **VmaxAdView** objects used in the project

2. To pass custom data at adView level(on VmaxAdView object) you can refer Ad view Settings

3.In the case where key's of the **Hashmap** passed to **vmaxAdView.setCustomData()**and **VmaxSdk.getInstance().setCustomData()** conflict, then the **Hashmap** passed to **vmaxAdView.setCustomData()** would be given priority.

2. Targeting for specifics

The JioAds SDK allows you to pass extra data to identify users for serving targeted campaign ads. For example, if your app users sign in using an email address, you can send this information to the server and create a campaign to target only specific user email addresses with ads.

```
   VmaxSdk.getInstance().setUserCity("<USER_CITY>");

   VmaxSdk.getInstance().setUserEmail("<USER_EMAIL>");

   VmaxSdk.getInstance().setLoginId("<LOGIN_ID>");

   VmaxSdk.getInstance().setUserAge(<AGE>);

   //possible values

   // GENDER_MALE,GENDER_FEMALE

   VmaxSdk.getInstance().setUserGender(VmaxSdk.Gender.GENDER_MALE);

   VmaxSdk.getInstance().setUserPincode(<PINCODE>);

   VmaxSdk.getInstance().setUserState("<USER_STATE>");
```

```

```

view rawVmaxSdk.java

Note: If invalid data is passed in any of the above parameters the SDK would ignore the data.

Clear Cached Media

Use the below api to clear the cached media of particular type.

```
VmaxSdk.getInstance().clearCachedMedia(context, VmaxSdk.MediaType);
```

Various options for VmaxSdk.MediaType:

VmaxSdk.MediaType.IMAGE	SDK will clear all Cached images from local memory
VmaxSdk.MediaType.VIDEO	SDK will clear all Cached Videos from local memory
VmaxSdk.MediaType.AUDIO	SDK will clear all Cached Audio from local memory
VmaxSdk.MediaType.ALL	SDK will clear all Cached Images, Videos & Audio from local memory

Disabling Google Play Service Lookup

Use the below api to disable google play service lookups. By default it's not disabled i.e. false. **Note that when google play service is disabled Ad SDK will generate custom advertising identifier**.

```
VmaxSdk.getInstance().disableGooglePlayService (true);
```

Disabling BPID Service

Use the below api to disable BPID service. By default it's not disabled(false).

```
VmaxSdk.getInstance().disableUidService (true);
```

Set Chromium Input

Use the below api to notify SDK if app is using Chromium Android. SDK will disable usage of system WebView if input is set to true.

```
VmaxSdk.getInstance().setChromium (true);
```

Releasing global app resources

In order for SDK to understand the app exit(session end) and release app level resources, publisher app can call below API on app exit.

```
VmaxSdk.getInstance().release();
```

Ad view Settings – JioAds SDK for Android

VmaxAdview class provides API'S to make various adspot level settings.

Set Ad Service ConnectionTimeout

This API is used to set connection timeout for ad service end point. Default timeout is **20 seconds**. **It's recommended not to use this API** to override the defaults **unless needed**. API takes inputs in seconds.

```
vmaxAdView.setAdTimeout (25);
```

Set Rendering Timeout

This API is only exposed for video ads and it allows setting rendition timeout when ad is received from ad service and fed to video player. Default value for the same is **20 seconds**. If android media player fails to prepare itself within this time, SDK considers this as an error case. **It's recommended not to use this API** to override the defaults **unless needed**. API takes inputs in seconds.

```
vmaxAdView.setTimeout (25);
```

Setting Child Application package name

Below API can be used to set Child application package name. This API will only work for trusted parent apps. If null string is passed it will be considered as reset.

```
vmaxAdView.setPackageName("PACKAE_NAME_HERE");
```

Enable Ad Storage

SDK provides a below API to cache ad in persistent storage and show it later.

```
vmaxAdView.enableAdStorage(true); //pass true if you want to enable storage.By
default it to set to false
```

Enable Media Caching

SDK provides a below API to cache Image/video/All Ad elements.

```
vmaxAdView.enableMediaCaching(VmaxSdk.CacheMode); //Default is cache disabled
```

If you are using xml Approach you can add the below attribute to the VmaxAdView layout

```
vmax:enableMediaCaching="<CACHE_MODE>"

CACHE_MODE can take any of the below values

@integer/ vmax_cache_mode_video

@integer/ vmax_cache_mode_image

@integer/ vmax_cache_mode_all
```

Various options for VmaxSdk.CacheMode:

VmaxSdk.CacheMode.IMAGE	SDK will download all Image ad elements during cacheAd() call. Only applicable for Native Ads
VmaxSdk.CacheMode.VIDEO	SDK will download Video Ad file during cacheAd() call
VmaxSdk.CacheMode.AUDIO	SDK will download Audio Ad file during cacheAd() call
VmaxSdk.CacheMode.ALL	SDK will download both Image as well as Video element of Ad during cacheAd() call

Closing Ads Programmatically

Below API can be used to close Interstitial/In-stream/Rewarded Video ad programmatically.

```
boolean closeAction = adView.closeAd(); //It will return false if the action is not completed(Ad is not closed)
```

In addition to this, closeAd() API can be called basis on AdView states

```
if(adView.getState() == VmaxAdView.AdViewState.STATE_INVIEW) {

      boolean closeAction = adView.closeAd();

  }
```

Autoclosing Ads

SDK provides a below API to auto close Interstitial/In-stream/Rewarded Video ads after ‘x’ seconds of Ad shown .

```
vmaxAdView.setCloseAfter(10);//SDK Will Attempt to auto close Ad after 10 seconds of ad shown. For Video ads duration will be calculated after media end.
```

Getting VAST Ad Identifier(Metadata)

SDK provides below API to fetch metadata for the VAST ad. This API is applicable only for VAST(Video) ads. Callbacks will be fired just prior to media starts too play.

```
vmaxAdView.getMetaData(new VmaxDataListener() {

@Override

public void onSuccess(String data) {

//metadata will be fetched in the onSuccess callback in JSON format

//eg. { "ad": {"title": "Test Title", "id": "Test id"} }

}



@Override

public void onFailure(VmaxError error) {
```

```
Log.e("vmax","Error : "+error.getErrorTitle()+"Error Desc : "
  "+error.getErrorDescription() + " Code :"+error.getErrorCode());

//Possible value for Error code fetched by error.getErrorCode()

//-> 3003-Error in fetching VAST meta data

 }

 }

});
```

Set Skip Event Key

API is only applicable for STB.Pass the Keycode of the button you want to act as skip button.

```
 vmaxAdView.setSkipEventKey(int KeyCode);
```

Targeting

1. Passing Custom Data to ad request

```
 HashMap<String,String> customData=new HashMap<>();

 customData.put("key1","value1");

 customData.put("key2","value2");

 vmaxAdView.setCustomData(customData);
```

view rawPassing custom data.java

2. Targeting for specifics

There are times when you want to target ads in particular sections of your app, ads to render as per language of your app content, for specific users of your app. Below API'S allows you to do so.

```
 //for complete list of categories refer link https://www.iab.com/guidelines/iab-
quality-assurance-guidelines-qag-taxonomy/

 vmaxAdView.setSectionCategory(Section.SectionCategory.CAREERS);// eg. for CAREERS
section

 vmaxAdView.setPageCategory(Section.BUSINESS.CONSTRUCTION);

 vmaxAdView.setLanguageOfArticle("<LANGUAGE>");
```

view rawLocal AdSettings.java

3. Keywords
You can pass on keyword information during ad request to have keyword targeted ads.

```
vmaxAdView.setKeyword("<KEYWORD>");
```

view rawKeyword.java

Customize ad appearance

Use the below code to customize adview background color. Default color for interstitial ads is black and transparent for banner ads.

```
vmaxAdView.setAdviewBackgroudColor("<COLOR_CODE>");
```

Enable keepScreenOn

Use the below code to override the device sleep time and ensure that the screen remains on. Default behavior is set to false i.e. it will respect the device sleep time.

```
vmaxAdView.keepScreenOn(true); // true to override the default behavior
```

Setting Media Quality

Use the below code to request the media of HD (High Definition) or SD (Standard Definition) quality

```
vmaxAdView.requestMediaQuality(VmaxAdView.MediaQuality.HD);

// for SD use VmaxAdView.MediaQuality.SD
```

Setting Ad Bitrate

Use the below API to request video ad of specific birate. This API is **only available for videos ads**. SDK will select Media file with best suitable bitrate. In case if api is not set, SDK will select media file basis on combination of device type & connection type.

```
vmaxAdView.setRequestedBitRate(int bitrate);
```

Requesting Ad duration

Use the below API to request the video ads of specific duration in seconds. This API is **only available for In-stream videos**.

```
  vmaxAdView.setRequestedAdDuration(30);

 //Here requested video ad duration is 30 seconds.
```

Note: Any input which is less than 5 seconds will be ignored.

Setting AdPod Variant (showing ad in Loop)

Use the below API whenever there is a requirement of showing instream ads in Loop. This will play instream video ad for infinite duration of time. Developer can stop this loop by calling **closeAd()** api of VmaxAdView class.

This API is **only available for In-stream videos**.

```
vmaxAdView.setAdPodVariant(VmaxAdView.AdPodVariant.INFINITE_AD_DURATION_WITH_LOOP);

//For now there is only one variation INFINITE_AD_DURATION_WITH_LOOP
```

Get Refresh Rate

Use below API to get refresh set for an ad.

Note : This is applicable for refreshable ad formats only

```
vmaxAdView.getRefreshRate();
```

Setting Pod Timeout

Use below API to set timeout in case of Instream Pod ads. In case If Pod doesn't start before set timeout value, SDK will give onAdError() callback. This API works for both finite and infinite duration Pods.

```
vmaxAdView.setPodTimeout(int timeout); //pass timeout in seconds
```

Disabling transition loader

This API is applicable **only for Instream ads**. By default SDK shows a Progress bar loader if ad gets time to start during showAd. User below API in order to disable this loader.

```
vmaxAdView.disableTransitionLoader();
```

Dampening ad requests

In order to reduce traffic on ad server, SDK manages record for subsequent no-fill on ad requests. And it allow/attempt next ad request in a Fibonacci manner. In this, Fibonacci number indicates time in minutes after which SDK will allow/attempt ad request. By default SDK progresses fibbonacci number upto 1440 mins and after that SDK will allow/attempt request after every 1440 mins. Developer can override this max limit using below API

```
vmaxAdView.setDampeningLimit(long maxLimit); // maxLimit in minutes
```

Requesting Ad orientation

Use below API to request for specific ad orientation. This API is applicable for interstitial ad formats

```
vmaxAdView.setRequestedOrientation(VmaxAdView.AdOrientation.PORTRAIT); //For portrait

vmaxAdView.setRequestedOrientation(VmaxAdView.AdOrientation.LANDSCAPE); //For landscape
```

Exposing Native Ad Elements

Use below API to request NativeAd object (only for trusted apps) after ad is cached.

```
NativeAd nativeAd = vmaxAdView.getNativeAd();
```

Following are the apis to fetch the individual elements from the nativeAd object

```
if (nativeAd != null){

  //------------ Primary elements ------------

    VmaxImage iconImage = nativeAd.getIcon();  // Icon Image

    String iconImageUrl = iconImage.getUrl();  // getIConUrl



    String title = nativeAd.getTitle();  // Title

    String sponsored = nativeAd.getSponsored();  // Sponsored

    String desc = nativeAd.getDesc();  // Desc

    String nativeAdPartner = nativeAd.getNativeAdPartner();  // NativeAdPartner

    VmaxImage mainImage = nativeAd.getImageMain();  // Main Image

    String mainImageUrl = mainImage.getUrl();  // getMainImgUrl

    String ctaText = nativeAd.getCtaText();  // CTA

    String vastVideoTag = nativeAd.getVastVideoTag();  // Vast XML

     //------------ Secondary elements ------------

    String desc2 = nativeAd.getDesc2();  // Desc2

    String tagLine = nativeAd.getTagLine();  // TagLine

    String rating = nativeAd.getRating();  // Rating

    String downloads = nativeAd.getDownloads();  // Downloads

    String price = nativeAd.getPrice();  // Price

    String salePrice = nativeAd.getSalePrice();  // SalePrice

    String phone = nativeAd.getPhone();  // Phone

    String address = nativeAd.getAddress();  // Address

    String displayUrl = nativeAd.getDisplayurl();  // Display Url

    String likes = nativeAd.getLikes ();  // likes

    VmaxImage customImage = nativeAd.getCustomImage();  // Custom Image

    //------------ Exposing trackers ------------//

    //Use either one of the below approaches to hit tracking events

    //1)Through SDK managed API

     // call when ad is successfully rendered

    nativeAd.handleImpression();
```

```
    // call when ad is clicked/interacted
    nativeAd.handleAdInteraction();
    //2) Manually hitting tracking URLs
    // hit all of the below urls when ad is successfully rendered
    JsonArray impressionUrlArray =nativeAd.getImpressionURL();
    // hit all of the below urls when ad is clicked
    JsonArray clickUrlArray=nativeAd.getClickURL();

}
```

Requesting ADS SPC through JioAds

You can request for Ads SPC using below api

```
VmaxRequest vmaxRequest = new VmaxRequest(this);

vmaxRequest.getAdsSPC(new AdsSPCListener() {
     @Override
     public void onSuccess(JSONObject ads) {
     }

     @Override
     public void onFailure(VmaxAdError error) {
     }
    }, "Adspot_1", "Adspot_2", "Adspot_3", "Adspot_4", "Adspot_5" );

//Adspotkey will be multi valued String argument so we can pass as many we want
```

Additionally you can have following settings (Optional)

Note: Make sure you set this optional data before calling getAdsSPC api

```
VmaxSdk vmaxSdk = VmaxSdk.getInstance();

HashMap<String,String> customData=new HashMap<>();

customData.put("key1","value1");

customData.put("key2","value2");

vmaxSdk.setCustomData(customData);

vmaxSdk.setUserCity("<USER_CITY>");

vmaxSdk.setUserEmail("<USER_EMAIL>");

vmaxSdk.setLoginId("<LOGIN_ID>");

vmaxSdk.setUserAge(<AGE>);

//possible values

// GENDER_MALE,GENDER_FEMALE

vmaxSdk.setUserGender(VmaxSdk.Gender.GENDER_MALE);

vmaxRequest.setSectionCategory(Section.SectionCategory.CAREERS);

vmaxRequest.setPageCategory(Section.BUSINESS.CONSTRUCTION);

vmaxRequest.setLanguageOfArticle("<LANGUAGE>");

vmaxRequest.setKeyword("<KEYWORD>");



HashMap<String,String> customData=new HashMap<>();

customData.put("key1","value1");
```

```
customData.put("key2","value2");

vmaxRequest .setCustomData(customData);
```

Video layout Customization

You can customize the media controls/video elements for the video ads by passing layout files through the API *setLayout(int portraitLayoutId, int landscapeLayoutId, Constants.AdCategory.VIDEO)* on *VmaxAdView* object. Layouts for media controls to be displayed in portrait mode and in landscape mode have to be passed using different layout files if it's different. Follow the below mentioned steps to do the same. The same layout can also be passed in both portrait and landscape if it's same.

Note: Below API will be deprecated in future releases.

setLayout(int portraitLayoutId, int landscapeLayoutId);//Currently calling this API assumes layouts being passed are for Video ads customization.

Publishers are recommended to use below overloaded API.

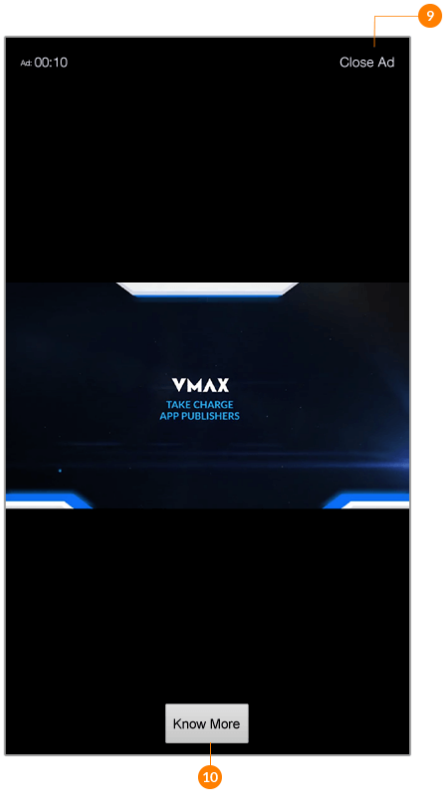
setLayout(int portraitLayoutId, int landscapeLayoutId, Constants.AdCategory.VIDEO))

If you are using XML Approach you can add the below attribute to the VmaxAdView layout

```
vmax:landscapeLayout="@layout/custom_landscape_layout"
  vmax:portraitLayout="@layout/custom_portrait_layout"
```


1
2
You can Skip Ad in 1s
3
WELCOME TO THE WORLD OF
SIMPLIFIED APP MONETIZATION
4
Ad: 00:14
5
6
7


8
WELCOME TO THE WORLD OF
SIMPLIFIED APP MONETIZATION
Ad: 00:04


9
Ad: 00:10
Close Ad
VMAX
TAKE CHARGE
APP PUBLISHERS
Know More
10

Video elements with tags

1 Mute/Un-mute icon

This will mute or un-mute the video

To display this use the tag **@string/vmax_video_volume_icon**

Eg.

```
<ImageView
    android:id="@+id/iv_sound_unmute_button"
    android:layout_width="30dp"
    android:layout_height="30dp"
    android:layout_marginLeft="10dp"
    android:layout_marginTop="10dp"
    android:tag="@string/vmax_video_volume_icon"
    android:src="@drawable/vmax_unmute"
    android:background="@drawable/vmax_mute"
    />
```

Pass drawables for mute and unmute icons in the attributes **android:background** and **android:src** respectively

2,8,9 Skip element,Close ad text ,close icon

This will display skip counter(if ad is having skip duration set), close ad text and close icon whichever is tagged as per your needs.

To display this use the tag **@string/vmax_video_skip_element .** Note that **SKIP_COUNTER** is a client side macro which will get replaced with actual skip down counter. You can position the macro as per you needs.

Eg.

```
<TextView
    android:id="@+id/iv_close_button"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignParentRight="true"
```

```
android:layout_alignParentTop="true"
android:gravity="center_vertical"
android:text="You can Skip Ad in SKIP_COUNTER"
android:drawableRight="@drawable/vmax_close_advertisement"
android:tag="@string/vmax_video_skip_element"
android:textColor="@android:color/white"
android:textAppearance="?android:attr/textAppearanceSmall"
android:contentDescription="Close the Ad"
android:layout_margin="10dp"/>
```

Pass drawable for for close icon in the attribute **android:drawableRight** , pass skip counter text in the attribute **android:text** and pass close ad text in the attribute **android:contentDescription**

Note: For STB an additional element needs to be included so that user can differentiate whether skip button is focused or not

```
<TextView
  android:layout_width="wrap_content"
  android:layout_height="wrap_content"
  android:gravity="center_vertical"
  android:text="Skip Ad"
  android:textAppearance="?android:attr/textAppearanceSmall"
  android:textColor="#B5B5BC"
  android:tag="@string/vmax_video_skip_element_focused"
  android:drawableLeft="@drawable/vmax_back_arrow_focused"
  android:drawablePadding="3dp"
  android:visibility="visible"
  />
```

The drawable paased in **android:drawableLeft** attribute will be shown when skip button is focused

3 Video Container

This is where the video view would be shown

To display this use the tag **@string/vmax_video_player_container**

Eg.

```
<RelativeLayout
    android:id="@+id/fl_video_container"
```

android:layout_width="match_parent"

android:layout_height="match_parent"

android:layout_centerInParent="true"

android:tag="@string/vmax_video_player_container"

android:layout_gravity="center_vertical">

4 Play Pause icon

This will play or pause the video

To display this use the tag **@string/vmax_video_playback_icon**

Eg.

<ImageView

android:layout_width="30dp"

android:layout_height="30dp"

android:layout_marginLeft="10dp"

android:layout_marginBottom="10dp"

android:id="@+id/adPlayback"

android:visibility="visible"

android:tag="@string/vmax_video_playback_icon"

android:layout_alignParentBottom="true"

android:layout_alignParentLeft="true"

android:layout_centerVertical="true"

android:background="@drawable/vmax_vast_play"

android:src="@drawable/vmax_vast_pause"/>

Pass drawables for for play and pause icons in the attributes **android:background** and **android:src** respectively

5 Ad-badge

This is to indicate that the displayed content is an Ad

Eg.

<TextView

```
android:layout_width="wrap_content"
android:layout_height="wrap_content"
android:text="Ad :"
android:layout_gravity="center"
android:textAppearance="?android:attr/textAppearanceSmall"
android:textSize="12sp"
android:textColor="@android:color/white"/>
```

6 Numerical progress

This will indicate progress of the video ad

To display this use the tag **@string/vmax_video_progresscount**

Eg.

```
<TextView
    android:id="@+id/progressCount"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="00"
    android:layout_gravity="center"
    android:padding="5dp"
    android:contentDescription="@string/vmax_video_progresscount_down"
    android:tag="@string/vmax_video_progresscount"
    android:textSize="12sp"
    android:textAppearance="?android:attr/textAppearanceSmall"
    android:textColor="@android:color/white" />
```

Pass values **@string/vmax_video_progresscount_up** or **@string/vmax_video_progresscount_down** in android:contentDescription tag to indicate if the numerical progress should ascend or descend

7 Progress Bar

This will show progress of the video

To display this use the tag **@string/vmax_video_progressbar**

Eg.

```
<ProgressBar
   android:id="@+id/pbProcessing"
   style="?android:attr/progressBarStyleHorizontal"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
  android:tag="@string/vmax_video_progressbar"
   android:layout_below="@+id/tvProcessing"
   android:indeterminateOnly="true"/>
```

10 CTA Button

This will display the Call To Action Button

To display this use the tag **@string/vmax_video_cta**

Eg.

```
<Button
      android:layout_centerHorizontal="true"
      android:layout_alignParentBottom="true"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:visibility="visible"
      android:layout_marginBottom="5dp"
      android:text="Know MOre"
      android:tag="@string/vmax_video_cta"
      />
```

Pass drawables for for mute and unmute icons in the attributes **android:background** and **android:src** respectively

Note: For STB an additional element needs to be included so that user can differentiate whether CTA button is focused or not

```
<TextView
   android:layout_width="wrap_content"
   android:layout_height="wrap_content"
   android:text="Visit Advertiser"
   android:drawableLeft="@drawable/vmax_video_cta_focused"
   android:gravity="center_vertical"
   android:textColor="#B5B5BC"
   android:tag="@string/vmax_video_cta_focused"
   android:visibility="visible"
   android:drawablePadding="3dp"
   />
```

The drawable passed in **android:drawableLeft** attribute will be shown when CTA button is focused

Example portrait and landscape layouts for passing in setLayout() API

Portrait Layout

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"

   android:layout_width="match_parent"

   android:layout_height="match_parent"

   android:id="@+id/rootLayout"

   android:background="@android:color/black"

   >

   <RelativeLayout

       android:id="@+id/fl_video_container"

       android:layout_width="match_parent"

       android:layout_height="match_parent"

       android:tag="@string/vmax_video_player_container"

       android:layout_centerInParent="true"

       android:layout_gravity="center_vertical"/>


   <TextView

       android:id="@+id/iv_close_button"

       android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"

        android:layout_alignParentRight="true"

        android:layout_alignParentTop="true"

        android:padding="10dp"

        android:text="You can close ad in SKIP_COUNTER"

        android:tag="@string/vmax_video_skip_element"

        android:textColor="@android:color/white"

        android:contentDescription="Close Ad"

        android:visibility="gone"

        android:layout_margin="5dp"/>


    <FrameLayout

        android:id="@+id/devProgressLayout"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:layout_alignTop="@+id/fl_video_container"

        android:visibility="visible">


        <LinearLayout

            android:padding="2dp"

            android:layout_width="wrap_content"

            android:layout_height="wrap_content"

            android:orientation="horizontal">


            <TextView

                android:id="@+id/ad_badge"

                android:layout_width="wrap_content"

                android:layout_height="wrap_content"

                android:layout_gravity="center"

                android:text="Ad : "
```

```
                android:textSize="8sp"
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:textColor="@android:color/white" />

            <TextView
                android:id="@+id/progressCount"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_gravity="center"
                android:text="00:00"
                android:tag="@string/vmax_video_progresscount"
                android:contentDescription="@string/vmax_video_progresscount_up"
                android:textSize="8sp"
                android:textAppearance="?android:attr/textAppearanceSmall"
                android:textColor="@android:color/white" />
        </LinearLayout>

    </FrameLayout>
    <Button
        android:layout_centerHorizontal="true"
        android:layout_alignParentBottom="true"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="visible"
        android:layout_marginBottom="5dp"
        android:text="Watch Now"
        android:tag="@string/vmax_video_cta"
        />
</RelativeLayout>
```

Landscape layout

```
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:id="@+id/rootLayout"
    android:background="@android:color/black"
    >
    <RelativeLayout
        android:id="@+id/fl_video_container"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:tag="@string/vmax_video_player_container"
        android:layout_centerInParent="true"
        android:layout_gravity="center_vertical"/>

    <TextView
        android:id="@+id/iv_close_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_alignParentRight="true"
        android:layout_alignParentTop="true"
        android:padding="10dp"
        android:text="You can close ad in SKIP_COUNTER"
        android:tag="@string/vmax_video_skip_element"
        android:textColor="@android:color/white"
        android:contentDescription="Close Ad"
        android:visibility="gone"
        android:layout_margin="5dp"/>
```

```
<FrameLayout
    android:id="@+id/devProgressLayout"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_alignTop="@+id/fl_video_container"
    android:visibility="visible">

    <LinearLayout
        android:padding="2dp"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:orientation="horizontal">

        <TextView
            android:id="@+id/ad_badge"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="Ad : "
            android:textSize="8sp"
            android:textAppearance="?android:attr/textAppearanceSmall"
            android:textColor="@android:color/white" />

        <TextView
            android:id="@+id/progressCount"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_gravity="center"
            android:text="00:00"
```

```
                android:tag="@string/vmax_video_progresscount"

                android:contentDescription="@string/vmax_video_progresscount_up"

                android:textSize="8sp"

                android:textAppearance="?android:attr/textAppearanceSmall"

                android:textColor="@android:color/white" />

        </LinearLayout>


    </FrameLayout>

    <Button

        android:layout_centerHorizontal="true"

        android:layout_alignParentBottom="true"

        android:layout_width="wrap_content"

        android:layout_height="wrap_content"

        android:visibility="visible"

        android:layout_marginBottom="5dp"

        android:text="Watch Now landscape"

        android:tag="@string/vmax_video_cta"

        />

</RelativeLayout>
```

Pass Layouts to JioAds SDK

You can pass the layouts to JioAds SDK using the **setLayout()** API as mentioned below ,this API needs to be called before you call the **showAd()** API

```
vmaxAdView.setLayout(R.layout.custom_portrait_layout,R.layout.custom_landscape_layout,Co
nstants.AdCategory.VIDEO);

//pass separate layouts for portrait and landscape containing media controls
```

```
vmaxAdView.setLayout(R.layout.custom_portrait_layout,-1, Constants.AdCategory.VIDEO);

//if you wish to pass just the portrait layout you can do the same with the landscape
layout
```

Enabling Custom show Ad

SDK allows publisher to have custom implementation to show ad/ads on their own. This feature is applicable only for Instream video ads for below variants.

- Regular instream video ad
- Instream video ad with finite duration
- Instream video ad with infinite duration

Proguard rules:

```
-keep public class com.vmax.android.ads.common.IVmaxAdQueue {
    public <fields>;
    public <methods>;
}
-keep public class com.vmax.android.ads.common.IVmaxAdEvents {
    public <fields>;
    public <methods>;
}
-keep public class com.vmax.android.ads.api.VmaxAd {
    public <fields>;
    public <methods>;
}
-keep public class com.vmax.android.ads.common.VmaxTracker {
     public <fields>;
     public <methods>;
 }
```

Cache Ad:

Use below API to enable custom show ad feature:

```
vmaxAdView.enableCustomShowAd(true);
```

Sample cacheAd code snippet:

```
VmaxAdView instreamAdView = new
VmaxAdView(this,"ADSPOT_ID_HERE",VmaxAdView.UX_INSTREAM_VIDEO);
instreamAdView.enableCustomShowAd(true);

//For finite pod case
//instreamAdView.setRequestedAdDuration(100);
//For infinite pod case
//instreamAdView.setAdPodVariant(VmaxAdView.AdPodVariant.INFINITE_AD_DURATION_WITH_
LOOP);

instreamAdView.cacheAd();
```

Custom show Ad:

Once onAdReady() callback is received, perform below steps.

Step 1: Get Object of IVmaxAdQueue as shown below:

```
iVmaxAdQueue = instreamAdView.getAdQueue();
```

Step 2: Dequeue ad/ads using iVmaxAdQueue object as shown below

```
if(iVmaxAdQueue !=null) {
    iVmaxAdQueue.dequeueVmaxAd(new IVmaxAdEvents() {
        @Override
        public void onReady(VmaxAd vmaxAd) {
            if (vmaxAd != null) {
                JSONObject dataObj = vmaxAd.getMetaData();
            }
         }

        @Override
        public void onError() { }

        @Override
        public void onAllAdsConsumed() {

        }

         @Override
         public void onNextAdReady() {

         }
    });
}
```

Note: For Finite or Infinite Ad pod case, developer is supposed to call dequeueVmaxAd() again and again. For each dequeueVmaxAd() call, SDK will keep next ad ready and give onNextAdReady() callback. Developers can query next ad after this onNextAdReady() callback is received.

Step 3: Fetching Ad Meta data

Developers can opt any of the below approach to get ad meta data

A) Fetching only media URL & duration :

```
iVmaxAdQueue.dequeueVmaxAd(new IVmaxAdEvents() {
    @Override
    public void onReady(VmaxAd vmaxAd) {
       if (vmaxAd != null) {
         String url = vmaxAd.getUrl();
         long duration = vmaxAd.getDuration();

        }}}
```

B) Fetching entire meta data json object:

```
iVmaxAdQueue.dequeueVmaxAd(new IVmaxAdEvents() {
    @Override
    public void onReady(VmaxAd vmaxAd) {
       if (vmaxAd != null) {
                JSONObject dataObj = vmaxAd.getMetaData();



        }}}
```

Step 4: Ad Tracking

SDK provides **VmaxTracker** class to perform ad tracking. Developers can make use of VmaxTracker class to perform tracking as well as Click events

```
iVmaxAdQueue.dequeueVmaxAd(new IVmaxAdEvents() {
    @Override
    public void onReady(VmaxAd vmaxAd) {
       if (vmaxAd != null) {
          VmaxTracker tracker = vmaxAd.getVmaxTracker();
          tracker.onImpression();
          tracker.onStart();
          tracker.onFirstQ();
          tracker.onMidpoint();
          tracker.onThirdQ();
          tracker.onComplete();
          tracker.onSkip();
          tracker.onPause();
          tracker.onResume();
          tracker.onMute();
          tracker.onUnmute();
          tracker.onClick();

          tracker.onStop();

          }}}
```

Note: Once ad is completed, it is mandatory to call onStop() on VmaxTracker object to clean up the resources.

Sample code snippet for Custom show ad:

```
public class AdDequeActivity extends Activity implements IVmaxAdEvents{
    private VmaxAdView instreamAdView;
    private IVmaxAdQueue iVmaxAdQueue;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        VmaxSdk.getInstance().setLogLevel(VmaxSdk.LogLevel.DEBUG);
        setContentView(R.layout.activity_ad_deque);
        instreamAdView = new
VmaxAdView(this,"ADSPOT_ID",VmaxAdView.UX_INSTREAM_VIDEO);
        instreamAdView.setAdListener(new VmaxAdListener() {
            @Override
            public void onAdReady(VmaxAdView adView) {
                iVmaxAdQueue = instreamAdView.getAdQueue();
                dequeueAd();
            }

            @Override
            public void onAdError(VmaxAdError Error, VmaxAdView adView) {}

            @Override
            public void onAdClose(VmaxAdView adView) {}

            @Override
            public void onAdMediaEnd(boolean isVideoCompleted, long reward,
VmaxAdView adView) {}
        });

        instreamAdView.enableCustomShowAd(true);

//instreamAdView.setAdPodVariant(VmaxAdView.AdPodVariant.INFINITE_AD_DURATION_WITH_
LOOP);
```

```
        instreamAdView.setRequestedAdDuration(100);
        instreamAdView.cacheAd();
    }

    private void dequeueAd(){
        iVmaxAdQueue.dequeueVmaxAd(this);
    }

    @Override
    public void onReady(VmaxAd vmaxAd) {
        if (vmaxAd != null) {

            String url = vmaxAd.getUrl();
            long duration = vmaxAd.getDuration();

            JSONObject dataObj = vmaxAd.getMetaData();
            if (dataObj != null) {
                Utility.longInfo(dataObj.toString());
            }

            VmaxTracker tracker = vmaxAd.getVmaxTracker();
            tracker.onImpression();
            tracker.onStart();
            tracker.onFirstQ();
            tracker.onMidpoint();
            tracker.onThirdQ();
            tracker.onSkip();
            tracker.onPause();
            tracker.onResume();
            tracker.onMute();
            tracker.onUnmute();
            tracker.onClick();
            tracker.onComplete();
            tracker.onStop();
        }
    }

    @Override
    public void onError() {}

    @Override
    public void onAllAdsConsumed() {

    }

    @Override
    public void onNextAdReady() {
        //For finite & infinite ad pod case
        dequeueAd(); // Next ad is ready to be dequeued
    }

    @Override
    protected void onDestroy() {
        super.onDestroy();

        if (instreamAdView != null) {
            instreamAdView.onDestroy();
        }
    }
}
```