



# Welcome to Vungle Placements

early access to placements and advanced reporting



# Contents

<b>Welcome! Here's what's new.</b>	<b>4</b>
Placements for Publishers	4
Advanced Reporting for Publishers	5
Advanced Reporting for Advertisers	5
<b>Setting up Placements in your Vungle Dashboard</b>	<b>5</b>
<b>Advanced Reporting from the Vungle Dashboard</b>	<b>10</b>
<b>Integrating the SDK v5.0 for iOS</b>	<b>12</b>
<b>Integrating the SDK v5.0 for Android</b>	<b>21</b>
<b>Advanced Reporting Using the Vungle API</b>	<b>30</b>

*Vungle*



# Welcome! Here's what's new.

Thank you for taking part in Vungle's Placements Beta Program. Among other advancements, Vungle's SDK v5 provides you with placements and advanced reporting to increase your monetization revenue with Vungle. Use this document as a guide to our new features.

## Placements for Publishers

Placements support enables publishers to customize the ad experience that shows in each placement. Use it to test which ad types are working best for each placement with placement-specific reporting.

Consider these applications for the placements feature:

- **Optimize for specific goals:** Defining different placements for ads within your app enables you to optimize some for performance and others for fill. For example, some developers prefer to optimize for performance for interstitial ads; whereas rewarded placements may be resilient enough to optimize for fill.
- **Multiple positions in the mediation waterfall:** With the placements feature, and if your mediation partner supports it, Vungle can now exist more than once in your mediation waterfall. Mediation partners typically keep the order of ad networks static, so that after a number of ad requests, the top ad network is delivering poorly performing ads until it stops filling ads for that device altogether, and another network gets a chance to optimize for performance.

Now, with a compatible mediation partner, you can position Vungle at the top of the waterfall, placing it in a premium performance slot. Ad requests for this placement optimize for performance. Enter Vungle again as a remnant inventory slot at the bottom of the waterfall, and now ads requested for this placement can focus on filling at volume after all the top-performing ads have been served.



## Advanced Reporting for Publishers

- **Placement:** Publishers who are using our new placements product can break down performance by placement.
- **Incentivized:** Publishers who use both our incentivized and our interstitial offerings will be able to break down performance by each format.
- **Hourly Granularity:** Publishers can break down performance by hour-of-the-day.

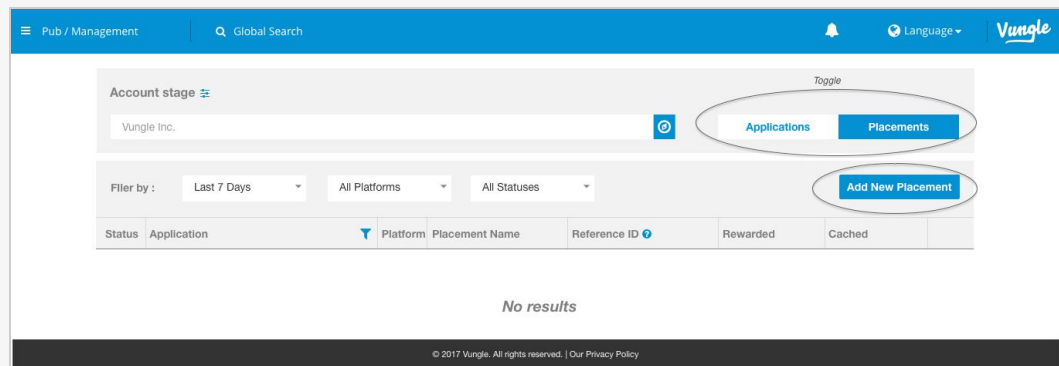
## Advanced Reporting for Advertisers


- **Creative:** Advertisers can now view the performance of their creatives as well as their campaigns. Creatives define the user's experience, so they're a much more intuitive way to analyse an advertiser's assets on the network.
- **Hourly Granularity:** Advertisers can break down performance by hour-of-the-day.
- **Publisher (Site):** Advertisers get insight into the performance of their ads by publisher app.

# Setting up Placements in your Vungle Dashboard

Head over to the familiar Vungle dashboard, which you can now use to define the placements in your app. We assume you are setting up placements for an existing app which is already registered with Vungle.

1. Log in to the [Vungle dashboard](#).
2. There are several ways to get to the Create Placements screen, and we will use the most common one.
  - Notice that in the upper right corner of the Publisher Application Management page, you can now toggle between Applications and the new Placements tab. One way to add a new placement is to toggle to **Placements** and click **Add New Placement**.



- Another way to create a placement is to choose an app and click on its name. This takes you to the placements page for that app, where you can click **Add New Placement**.
  - Finally, you can select an app and click the  (edit settings) button to its right, which takes you to the Application Details page. There, you can click the **Add New Placement** link in the right panel.
  - Basically, anytime you are editing an app, you can work with its placements.
3. In the **Create Placement** page, select the **Application** to which you are adding the placement. The app name may already be filled in for you if you accessed the Create Placement page from a specific app.

### Create Placement

**To use Placements, please make sure your Application's SDK is updated to at least SDK 5.0.**

- [Download the latest SDK](#)
- [Learn more about placements](#)

Application
 

Test App 1

Name
 

EndOfLevel


Reference ID
 


Automatically generated after creation

Rewarded?
 










☐ Provide in-app rewards for your users who view an ad

4. Now enter the placement **Name** of your first placement (the Reference ID will be assigned automatically), and indicate whether it is a **Rewarded** ad experience.
5. Click **Submit**. You see the placements defined for that app.

Application stage 

Vungle Exchange Demo


Add New Placement

Status	Placement 	Reference ID 	Auto Cached 	Bid Floor	
	EndOfGame	ENDOFGA96028	<input type="checkbox"/>	\$0.00	 
	EndOfLevel	ENDOFLE21435	<input checked="" type="checkbox"/>	\$0.00	 

Note that:

- Vungle will automatically generate a **Reference ID** for each placement you enter; this is the placement's unique identifier and is required information for identifying a placement (for example, if you query the Reporting API and want to filter on a placement, you will need to provide this Reference ID).
- Vungle will only auto-cache ads on the user's device for one placement. Ads for other placements are cached when your app calls the loadAd method in the SDK.

If this is the first placement you are creating for an app, it is automatically



designated as the auto-cached placement for that app. Once you have multiple placements defined, you can designate the auto-cached one in the placements screens. Typically, developers select their most frequently occurring placement for auto-caching.

Now your Application Details page includes any placements defined for the app. You can find the placement Reference ID in the Application Details page.

The screenshot shows the Vungle Application Details page for an application named "Justin Diaz Design". The page is divided into several sections:

- Status:** Shows the application is in "Test Mode" and "Active". It notes that the app is in Test Mode and is receiving test ads.
- Publisher settings:** Includes fields for Frequency cap (10), Force View (Skippable non-incentivized ads, Skippable incentivized ads), OS Version (6.0), Maximum Ad Duration (15), Download options (Wifi + WWAN), Incentivized Callback URL, and Secure server callback (None).
- Application details:** Includes fields for Application Name (Justin Diaz Design), Vungle Application ID (58893cb0ba2fbbcf31000607), Reporting API ID (58893cb0ba2fbbcf31000607), App Store URL, Category (Business), Platform (iOS), Type (Free), and Orientation.
- Placements:** This section is circled in red and shows three entries, all with the same Reference ID: 234234124324. The entries are labeled "Placement Name" and "Reference ID".

Now when you look at your apps in the dashboard, they are broken down by placements.





Pub / Management

Global Search

Language

Vungle

Account stage

Vungle Inc.

Applications

Placements

Filter by :

Last 7 Days

All Platforms

All Statuses

Add New Application

Status	Application	Platform	Date modified	Views	Completes	New Devices	Revenue	eCPM	
	Angry Birds Placements: 1		2017.02.09 9:42pm	50,000	100,000	25	\$50,000	\$100.00	
	Application Name Placements: 2		2017.02.09 9:42pm	50,000	100,000	25	\$50,000	\$100.00	
	Application Name Placements: 2		2017.02.09 9:42pm	50,000	100,000	25	\$50,000	\$100.00	
	Application Name Placements: 2		2017.02.09 9:42pm	50,000	100,000	25	\$50,000	\$100.00	

© 2017 Vungle. All rights reserved. | Our Privacy Policy

And toggling to the Placements page enables you to review all your placements. This is also a good place to find the placement Reference ID, if you need it.

Pub / Management

Global Search

1

Vungle

Account stage

Vungle, Inc.

Applications

Placements

Filter by:

All Statuses

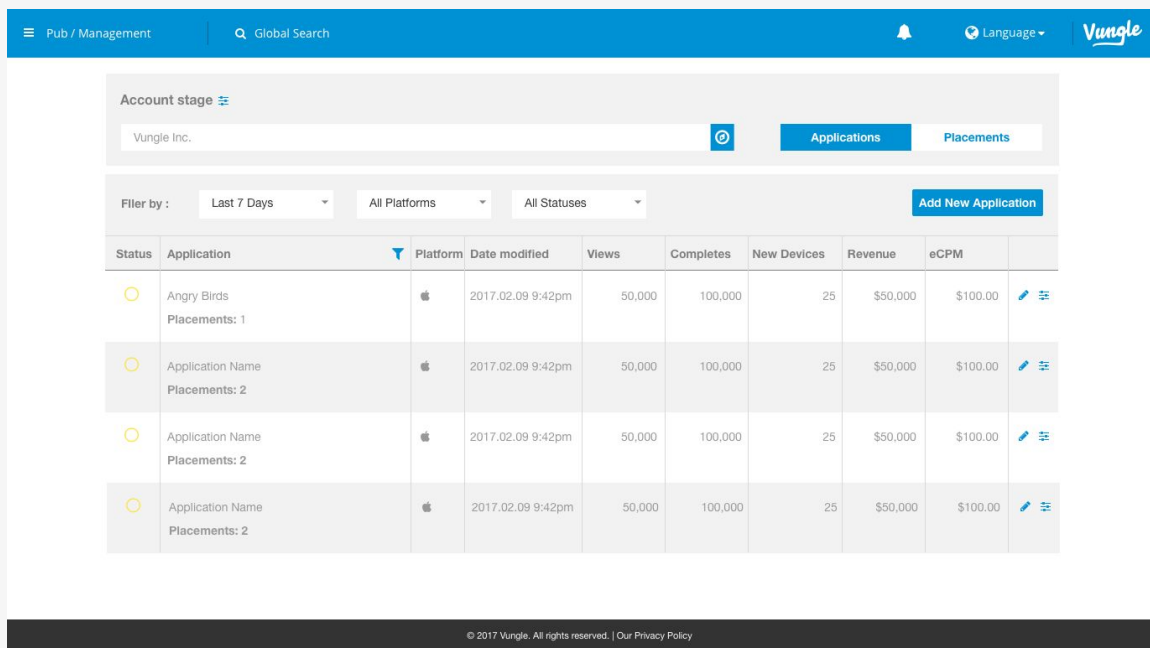
Add New Placement

Status	Application	Platform	Placement	Reference ID	Auto Cached	Bid Floor	
	Test App 1		EndOfGame	ENDOFGA02182	No	\$0.00	
	Test App 1		EndOfLevel	ENDOFLE27088	Yes	\$0.00	

# Advanced Reporting from the Vungle Dashboard

Back in our Vungle dashboard, reporting on one of your apps now includes information on all its placements.

1. Log in to the [Vungle dashboard](#).
2. Recall that you can now toggle between the Applications page and the new Placements page. Toggle to **Applications**.
3. Now when you look at your apps in the dashboard, the Applications page shows all the placements you have defined for each app, along with the views, completes, new devices, revenue, and eCPM for your app, broken down by each placement.



Status	Application	Platform	Date modified	Views	Completes	New Devices	Revenue	eCPM	
○	Angry Birds Placements: 1	Apple	2017.02.09 9:42pm	50,000	100,000	25	\$50,000	\$100.00	<a href="#">edit</a> <a href="#">delete</a>
○	Application Name Placements: 2	Apple	2017.02.09 9:42pm	50,000	100,000	25	\$50,000	\$100.00	<a href="#">edit</a> <a href="#">delete</a>
○	Application Name Placements: 2	Apple	2017.02.09 9:42pm	50,000	100,000	25	\$50,000	\$100.00	<a href="#">edit</a> <a href="#">delete</a>
○	Application Name Placements: 2	Apple	2017.02.09 9:42pm	50,000	100,000	25	\$50,000	\$100.00	<a href="#">edit</a> <a href="#">delete</a>

© 2017 Vungle. All rights reserved. | [Our Privacy Policy](#)

Toggling to the Placements Management page enables you to review the setup for all your placements.

4. Navigate to **Reports** from your Publisher menu. You can now download reports for your apps that are broken down by placement.
5. You can also use the Vungle API for advanced reporting that now includes placement information. Refer to the [“Advanced Reporting Using the Vungle API”](#) section in this document.

*Vungle*



# Integrating the SDK v5.0 for iOS

## Step 1. Add the Vungle Framework to your Xcode Project

### 1. Add the VungleSDK.embeddedFramework to your Project

If you are updating from a previous version of the Vungle SDK, first remove the `VungleSDK.embeddedFramework` directory completely before adding the new SDK.

Find the extracted files and drag the `VungleSDK.embeddedFramework` directory into Xcode under **Frameworks**. Be sure to add the `VungleSDK.embeddedFramework` folder as a group (yellow folder) and not as a reference (blue folder).

### 2. Add Other Required Frameworks

The Vungle SDK requires that you link a few other native frameworks to your project, so click on your project in **Project Navigator** and go to **General → Linked Frameworks and Libraries**.

Many of these frameworks are already included as a default for most Xcode projects, but be sure to add any of the following that are not already included:

- `AdSupport.framework`
- `AudioToolbox.framework`
- `AVFoundation.framework`
- `CFNetwork.framework`
- `CoreGraphics.framework`
- `CoreMedia.framework`
- `Foundation.framework`
- `libz.dylib` or `libz.tbd`
- `libsqlite3.dylib` or `libsqlite3.tbd`
- `MediaPlayer.framework`
- `QuartzCore.framework`



- StoreKit.framework
- SystemConfiguration.framework
- UIKit.framework
- WebKit.framework

Make sure that the VungleSDK framework appears under **Linked Frameworks and Libraries**.

### 3. Add the “-ObjC” Linker Flag

Click on your project in **Project Navigator** and go to **Build Settings** → **Linking** → **Other Linker Flags**. Add **-ObjC** to **Other Linker Flags**.

## Step 2. Remove the iOS Status Bar

Although this step is not required, we recommend that you remove the iOS status bar to ensure that Vungle's ad interaction and presentation perform smoothly. To remove the status bar, open your Info.plist, add the key **View controller-based status bar appearance**, and set it to **No**.

## Step 3. Add Code

### Initialize the SDK

Initialize the SDK as soon as your app starts in order to give the SDK enough time to cache an ad for the auto-cached placement. You will need the App ID and all the Placement IDs you want to use in your app (both active and inactive) to initialize the SDK. You can find these IDs in the Vungle Dashboard (refer to the [“Setting up Placements in your Vungle Dashboard”](#) section of this document).

```
- (BOOL)startWithAppId:(nonnull NSString *)appID placements:(nonnull  
NSArray<NSString *> *)placements error:(NSError **)error;
```

Sample code

```
NSString* appID = @"Your_AppID_Here";
```



```
NSArray* placementIDsArray = @[@"Your_PlacementID_1", @"Your_PlacementID_2",  
@"Your_PlacementID_3"];  
VungleSDK* sdk = [VungleSDK sharedSDK];  
[sdk startWithAppId:appID placements:self.placementIDsArray error:&error];
```

Once the SDK is initialized successfully, the following callback method is called:

```
- (void)vungleSDKDidInitialize;
```

Refer to the [“Delegate Callbacks”](#) section of this document.

You can also check the status of the SDK initialization with the following property:

```
@property (atomic, readonly, getter=isInitialized) BOOL initialized;
```

After the SDK is initialized, it automatically caches an ad for the placement you selected as **Auto Cached** in the Vungle Dashboard. We recommend selecting the most viewed placement for auto-caching.

Once an ad is cached successfully, the `vungleAdPlayabilityUpdate` callback method is called with the Placement ID matching your **Auto Cached** placement. (Refer to the [“Check Ad Availability for a Placement”](#) section of this document.)

## Load an Ad for a Placement

For placements other than the auto-cached placement, call `loadPlacementWithID` method to load an ad.

```
- (BOOL)loadPlacementWithID:(NSString *)placementID error:(NSError **)error;
```

Sample code

```
VungleSDK* sdk = [VungleSDK sharedSDK];  
[sdk loadPlacementWithID:@"Your_PlacementID" error:&error];
```

Refer to the [“Check Ad Availability for a Placement”](#) section of this document.



## Check Ad Availability for a Placement

Once the SDK finishes caching an ad for a placement, the following callback method is called:

```
- (void)vungleAdPlayabilityUpdate:(BOOL)isAdPlayable placementID:(nullable  
NSString *)placementID;
```

Sample code:

```
- (void)vungleAdPlayabilityUpdate:(BOOL)isAdPlayable placementID:(NSString  
*)placementID {  
    if([placementID isEqualToString:@"<Your_PlacementID_1>"]) {  
        self.playButtonPlacement1.enabled = isAdPlayable;  
    }  
}
```

**Note:** For the auto-cached placement, only when an ad becomes available is this callback method called. The SDK will keep requesting an ad for the auto-cached placement. For all other placements, this callback method is called in case of "Load Failed" (isAdPlayable returns 'NO' in this case).

You can also check the ad availability for a placement with the following property:

```
- (BOOL)isAdCachedForPlacementID:(nonnull NSString *)placementID;
```

## Play an Ad

After you make sure that an ad is ready for a placement, you can play the ad with the following method:

```
- (BOOL)playAd:(UIViewController *)controller options:(nullable NSDictionary  
*)options placementID:(nullable NSString *)placementID error:( NSError  
*__autoreleasing _Nullable *_Nullable)error;
```

Sample Code:

```
VungleSDK* sdk = [VungleSDK sharedSDK];
```



```
NSError *error;
[self.sdk playAd:self options:nil placementID:kVungleTestPlacementID01
error:&error];
if (error) {
    NSLog(@"Error encountered playing ad: %@", error);
}
```

## Delegate Callbacks

You can receive callbacks from the SDK with `VungleSDKDelegate`. There are four callback methods in the delegate in which you are notified of the SDK events.

You can attach and detach your delegate with:

```
// Attach
[[VungleSDK sharedSDK] setDelegate:yourDelegateInstance];

// Detach
[[VungleSDK sharedSDK] setDelegate:nil];
```

**Note:** Remember to clear the registered delegate when it's no longer needed to avoid memory leaks.

The following method is called when the SDK is about to play a video ad. This is a great place to pause gameplay, sound effects, animations, etc.

```
- (BOOL)vungleWillShowAdForPlacementID:(nullable NSString *)placementID;
```

The following method is called when the SDK is about to close an ad. This is a great place to reward your user and resume gameplay, sound effects, animations, etc.

```
- (void)vungleWillCloseAdWithViewInfo:(nonnull VungleViewInfo *)info
placementID:(nonnull NSString *)placementID;
```

`VungleViewInfo` includes the following properties for you to check a result of ad play:

```
@interface VungleViewInfo : NSObject <NSCopying>
//Represents a BOOL whether or not the video can be considered a completed
view.
```





```
@property (nonatomic, readonly) NSNumber *completedView;  
//The time in seconds that the user watched the video.  
@property (nonatomic, readonly) NSNumber *playTime;  
//Represents a BOOL whether or not the user clicked the download button.  
@property (nonatomic, readonly) NSNumber *didDownload;  
@end
```

The following method is called when the SDK has changed ad availability status. The `isAdPlayable` boolean denotes the new playability of a specific `placementID`.

```
- (void)vungleAdPlayabilityUpdate:(BOOL)isAdPlayable placementID:(nullable  
NSString *)placementID;
```

Refer to the [“Check Ad Availability for a Placement”](#) section of this document.

The following method is called when the SDK is initialized successfully:

```
- (void)vungleSDKDidInitialize;
```

## Customization Options

Use these options to customize the ad experience for playback.

Option Keys	Default	Description
VunglePlayAdOptionKeyOrientations	UIInterfaceOrientationMaskAll An NSNumber representing a bitmask with orientations (defaults to autorotate).	Sets the orientation of the ad. We recommend allowing ads to autorotate, even if your app is in portrait. This way, the user has the option to watch full-size videos, resulting in a better user experience. You can achieve this by setting the orientation on a view controller level (rather than a project level).
VunglePlayAdOptionKeyUser	nil NSString	Sets your user ID. The value is passed to Vungle server, and then sent to your server through server-to-server callback system if an placement is set to “Rewarded”.

VunglePlayAdOptionKeyIncentivizedAlertTitleText	nil NSString	String that is used as the title of the alert dialog presented when a user closes an incentivized ad experience prematurely.
VunglePlayAdOptionKeyIncentivizedAlertBodyText	Are you sure you want to skip this ad? If you do, you might not get your reward NSString	String that is used as the body text of the alert dialog presented when a user closes an incentivized ad experience prematurely.
VunglePlayAdOptionKeyIncentivizedAlertCloseButtonText	Close NSString	String title for the close button text of the alert dialog presented when a user closes an incentivized ad experience prematurely.
VunglePlayAdOptionKeyIncentivizedAlertContinueButtonText	Continue NSString	String title for the close button text of the alert dialog presented when a user closes an incentivized ad experience prematurely.

## Sample code

```

NSDictionary *options = @{@"VunglePlayAdOptionKeyOrientations:
    @(UIInterfaceOrientationMaskLandscape),
                           VunglePlayAdOptionKeyUser: @"userGameID",
                           VunglePlayAdOptionKeyIncentivizedAlertBodyText :
@"If the video isn't completed you won't get your reward! Are you sure you
want to close early?",

                           VunglePlayAdOptionKeyIncentivizedAlertCloseButtonText : @"Close",

                           VunglePlayAdOptionKeyIncentivizedAlertContinueButtonText : @"Keep Watching",
                           VunglePlayAdOptionKeyIncentivizedAlertTitleText :
@"Careful!"};

// Pass in dict of options, play ad
NSError *error;
[self.sdk playAd:self options:options placementID:<your_placemnt_id_here>
error:&error];

```



```
if (error) {  
    NSLog(@"Error encountered playing ad: %@", error);  
}
```

## Debug

If you need to get SDK info, you can get info with this property:

```
- (NSDictionary *)debugInfo;
```

If you want the SDK to output logs, use the following method:

```
- (void)setLoggingEnabled:(BOOL)enable;
```

## VungleSDKLogger Protocol

```
@protocol VungleSDKLogger  
- (void)vungleSDKLog:(NSString*)message;  
@end
```

The VungleSDK singleton sends logging events to any attached class following the VungleSDKLogger protocol. The log event contains the NSString value that is also printed to console (if logging has been enabled). To attach your logger, use the following:

```
[sdk attachLogger:yourLoggerInstance];
```

As mentioned above, it's important to clear out attached loggers from the VungleSDK. Loggers can be detached using the following approach:

```
[sdk detachLogger:yourLoggerInstance];
```

## assetLoader Protocol

```
@protocol VungleAssetLoader  
/**  
 * should return a valid NSData containing the (raw) data of an image for the  
 specified path or nil. */
```



```
- (NSData*)vungleLoadAsset:(NSString*)path;

/**
 * should return a valid UIImage for the specified path, or nil.
 */
- (UIImage*)vungleLoadImage:(NSString*)path;
@end
```

# Integrating the SDK v5.0 for Android

## Requirements

- Android 3.0 (Honeycomb - API version 11) or later
- Java 1.7 - For Android 5.+ compatibility purposes, JDK 7 is required
- Java 1.8 - For Android 7.+ compatibility purposes, JDK 8 is required

## Step 1. Update the Gradle Script

Add the following compile options in your `build.gradle` file:

In the Project gradle script, add maven URL:

```
all projects {
    repositories {
        maven {
            url "https://jitpack.io"
        }
        ...
    }
}
```

In the Module gradle script, enable multiDex and add the following compile dependencies:

```
dependencies {
    ....
    compile('javax.inject:javax.inject:1',
            'com.google.dagger:dagger:2.7',
            'de.greenrobot:eventbus:2.2.1',
            'io.reactivex:rxjava:1.2.0',
            'io.reactivex:rxandroid:1.2.1')
    compile
'com.github.vungle:vungle-android-sdk:feature-and-718-javadoc-SNAPSHOT'
    ....
}
```



## Step 2. Update AndroidManifest.xml

Add the following lines to your AndroidManifest.xml:

```
<!-- permissions to download and cache video ads for playback -->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"
android:maxSdkVersion="18"/>
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />

<application>
...

</application>
```

## Step 3. Initialize the Vungle SDK

### Application Startup

Initialize the Vungle Publisher SDK in your application's first Activity with the active placement IDs you want to use inside the app. The SDK will be initialized asynchronously and will return a callback to the `VungleInitListener` provided in `init`.

```
public class FirstActivity extends android.app.Activity {

    // get the VunglePub instance
    final VunglePub vunglePub = VunglePub.getInstance();

    // get your App ID from the app's main page on the Vungle Dashboard after
    setting up your app

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // initialize Publisher SDK with app id, placement id list and init
        callback handler
        vunglePub.init(this, app_id, new String[] { placementID1, placementID2,
        placementID3 }, vungleInitListener);
```

```
    ...  
  }  
}
```

## Each Activity

In addition, override the `onPause` and `onResume` methods in each Activity (including the first) to ensure that the Vungle Android SDK is properly updated when your application gains or loses focus.

```
public class EachActivity extends android.app.Activity {  
  
    // get the VunglePub instance  
    final VunglePub vunglePub = VunglePub.getInstance();  
    ...  
    @Override  
    protected void onPause() {  
        super.onPause();  
        vunglePub.onPause();  
    }  
  
    @Override  
    protected void onResume() {  
        super.onResume();  
        vunglePub.onResume();  
    }  
}
```

## Step 4. Set the Listeners

The Vungle SDK raises several events that you can handle programmatically by implementing `VungleEventListener` classes and registering them using `clearAndSetEventListeners`. Remember to remove the event listener when you don't need to use it anymore to prevent memory leaks.

```
vunglePub.clearAndSetEventListeners(vungleDefaultListener,  
vungleSecondListener);
```



## Step 5. Load and Play an Ad

Once the Vungle SDK is successfully initialized, you can load your placement and play the ad when it's ready. If you set the `VungleEventListener`, it will notify through the `onAdAvailabilityUpdate(String placementReferenceId, boolean isAdAvailable)` callback when an ad is available to play.

```
public class GameActivity extends android.app.Activity {

    // get the VunglePub instance
    final VunglePub vunglePub = VunglePub.getInstance();
    final String placementIdForLevel = "your placement id";

    private void onLevelStart() {
        vunglePub.loadAd(placementIdForLevel);
    }

    private void onLevelComplete() {
        if (vunglePub.isAdPlayable(placementIdForLevel)) {
            vunglePub.playAd(placementIdForLevel);
        }
    }
}
```

Note that for the auto-cached placement, you don't need to call `loadAd` because the SDK will automatically load an ad after initialization. We recommend choosing most viewed placement as your auto-cached selection.

To define whether a user has the option to close out of an ad, use the forced view options in the [Vungle Dashboard](#).

**Note:** Test mode is not supported in the placement SDK yet.

## Advanced Settings

### Google Play Services (Optional)

Coming soon!





Proguard

Coming soon!

## The EventListener Interface

Available methods to manipulate the `VungleAdEventListener` are listed below:

Method	Description
<code>clearAndSetEventListeners(VungleEventListener..)</code>	Clears registered EventListeners and then adds the input eventListeners.
<code>clearEventListeners( )</code>	Clears all EventListeners
<code>removeEventListeners(VungleEventListener..)</code>	Removes the input EventListeners.
<code>addEventListeners(VungleEventListener..)</code>	Adds the input eventListeners

`VungleAdEventListener` delegate call API:

```
public class FirstActivity extends android.app.Activity {  
    ...  
  
    private final VungleAdEventListener vungleListener = new  
    VungleAdEventListener(){  
  
        @Override  
        public void onAdEnd(String placementReferenceId, boolean wasSuccessfulView,  
        boolean wasCallToActionClicked) {  
            // Called when user exits the ad and control is returned to your  
            application  
            // if wasSuccessfulView is true, the user watched the ad and should be  
            rewarded  
            // (if this was a rewarded ad).  
            // if wasCallToActionClicked is true, the user clicked the call to  
            action  
            // button in the ad.
```



```
    }

    @Override
    public void onAdStart(String placementReferenceId) {
        // Called before playing an ad
    }

    @Override
    public void onUnableToPlayAd(String placementReferenceId, String reason) {
        // Called after playAd(placementId, adConfig) is unable to play the ad
    }

    @Override
    public void onAdAvailabilityUpdate(String placementReferenceId, boolean
isAdAvailable) {
        // Notifies ad availability for the indicated placement
        // There can be duplicate notifications
    }
};

@Override
public void onCreate(Bundle savedInstanceState) {
    ...

    vunglePub.init(this, app_id, placement_id_list, initCallback);
    vunglePub.clearAndSetEventListeners(vungleListener);

};

@Override
public void onDestroy() {
    ...
    vunglePub.clearEventListeners();

};
}
```

Vungle also provides `VungleInitListener` for SDK initialization event update.



```
public void onSuccess();  
public void onFailure(Throwable error);
```

## UI Thread Note

Callbacks are executed on a background thread, so any UI interaction or updates resulting from an event callback must be passed to the main UI thread before executing. Two common ways to run your code on the UI thread are:

- [Handler](#)
- [Activity.runOnUiThread\(Runnable\)](#)

## Configuration Options

### Global Ad Configuration

After calling `init` you can optionally get access to the global `AdConfig` object. This object allows you to set options that will be automatically applied to every ad you play.

```
public class FirstActivity extends android.app.Activity {  
    ...  
    @Override  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        ...  
        vunglePub.init(this, app_id);  
  
        // get a reference to the global AdConfig object  
        final AdConfig globalAdConfig = vunglePub.getGlobalAdConfig();  
  
        // For a full description of available options, see the 'Config Object'  
        section.  
        globalAdConfig.setSoundEnabled(true);  
    }  
}
```

### Single Ad Configuration

You can optionally customize each individual ad you play by providing an `AdConfig` object to `playAd`. If you set any options in the global ad configuration, those options will be

overridden by the provided options. Pass an override AdConfig with the following approach:

```
public class GameActivity extends android.app.Activity {
    ...
    private void onLevelComplete() {
        // create a new AdConfig object
        final AdConfig overrideConfig = new AdConfig();

        overrideConfig.setSoundEnabled(false);

        // the overrideConfig object will only affect this ad play.
        vunglePub.playAd(yourPlacementId, overrideConfig);
    }
}
```

## The AdConfig Object

The override AdConfig has a collection of options that can be set for an individual ad play. Available options are listed below:

Method	Default	Description
setOrientation	Orientation. matchVideo	Orientation.autoRotate indicates that the ad will autorotate with the device orientation. Orientation.matchVideo indicates that the ad will play in the best orientation for the video (usually landscape).
setSoundEnabled	true	Sets the starting sound state for the ad. If true, the audio respects device volume and sound settings. If false, video begins muted but user may modify.
setBackButtonImmediatelyEnabled	false	If true, allows the user to immediately exit an ad using the back button. If false, the user cannot use the back button to exit the ad until the on-screen close button is shown.
setImmersiveMode	false	Enables or disables <a href="#">immersive mode</a> on KitKat+ devices

setIncentivizedUserId	none	Sets the unique user id to be passed to your application to verify that this user should be rewarded for watching an incentivized ad. N/A if ad is not incentivized.
setIncentivizedCancelDialogTitle	"Close video?"	Sets the title of the confirmation dialog when skipping an incentivized ad. N/A if ad is not incentivized.
setIncentivizedCancelDialogBodyText	"Closing this video early will prevent you from earning your reward. Are you sure?"	Sets the body of the confirmation dialog when skipping an incentivized ad. N/A if ad is not incentivized.
setIncentivizedCancelDialogCloseButtonText	"Close video"	Sets the 'cancel button' text of the confirmation dialog when skipping an incentivized ad. N/A if ad is not incentivized.
setIncentivizedCancelDialogKeepWatchingButtonText	"Keep watching"	Sets the 'keep watching button' text of the confirmation dialog when skipping an incentivized ad. N/A if ad is not incentivized.
setTransitionAnimationEnabled	false	Enables or disables standard fragment transition animation



# Advanced Reporting Using the Vungle API

## Host and Path

Vungle's new reporting API has a new home. All subsequent revisions and improvements to Vungle's reporting will be at the host described below.

Host	Path
https://report.vungle.com	/api/v1/ext/pub/reports/performance

## Authentication

Authentication is done using the same reporting API key that you currently use in our existing API. Each User in an account may have their own API key. You can find and generate API keys in your account page on the Vungle dashboard.

The API key is now passed through a request's header, rather than as a parameter.

Header Key	Header Value
Authorization	Bearer [API KEY]

## Query

Queries to our API are controlled in 3 ways: filters, dimensions, and aggregates.

### Filters

Filters allow you to restrict the result set to the data that you are interested in. You can specify date ranges, specific countries, and specific applications. They are separate parameters that you can add to your query:

Parameter Name	Format	Action	If not in query	Usage Example(s)
startDate	ISO8601 date	Limits the result set to performance data no earlier than this date	Reject request	startDate=2017-01-01
endDate	ISO 8601 date	Limits the result set	Reject	endDate=2017-01-02

		to performance data no later than this date	request	
country	Comma separated list of ISO 3166-1 Alpha-2 country codes	Returns only performance data matching the listed countries	Return all countries	country=US country=US,CA country=US,CA,AU
applicationId	Comma separated list of Vungle Application IDs to return	Returns only performance data for the listed applications	Return all applications	applicationId=586e201e242e3fd30123450220
incentivized	'true'/'false' or 1/0	Returns only performance data for incentivized or non-incentivized traffic	Return both incent and non-incent	incentivized=true incentivized=false

## Dimensions

Dimensions allow you to determine the granularity of your request. For example, you can break results down by platform, date, or application. They are passed in one parameter: dimensions.

Parameter Name	Format	Example(s)
dimensions	Comma separated list of specific strings, listed in the table below	dimensions=platform dimensions=application,date,country

Below is the list of supported dimensions:

Dimension Name	Returns
platform	Grouped by platform ('android', 'ios', 'windows')
application	Grouped by application ID and name
date	Grouped by date
country	Grouped by country

incentivized	Grouped by incentivized/un-incentivized traffic
--------------	---

## Aggregates

Aggregates allow you to specify the performance data that you are interested in, like impression counts, revenue totals, or eCPM. They are requested in one parameter: aggregates.

Parameter Name	Format	Example(s)
aggregates	Comma separated list of specific strings, listed in the table below	aggregates=views aggregates=views,revenue,ecpm

Below is the list of supported aggregates:

Aggregate Name	Returns
views	integer
completes	integer
clicks	integer
revenue	float
ecpm	float

## Results

### Format

By default, we return a plain text result as a CSV. We also support a JSON result, as where each 'row' is a JSON object, organised in a JSON array. You can specify which format you prefer using the 'format' parameter.

Parameter Name	Format	Example(s)
format	Either 'csv' or 'json'	format=csv format=json





## Limit

The reporting API currently supports a maximum return of 1000 rows. If you receive an error that your result set is too large, you should try again with a more narrow query.

## Range

The API supports data retrieval from March 1, 2017.

## Example

### Query

```
https://report.vungle.com/api/v1/ext/pub/reports/performance?dimensions=placement&aggregates=views,revenue&startDate=2017-03-01&endDate=2017-03-05&format=json
```

### Response

```
[
  {
    "placement id" : "12345678",
    "placement name": "level 3",
    "views": 1234,
    "revenue": 123.0
  }
]
```