Welcome to Vungle Placements

early access to placements and advanced reporting

2017-06-21

Contents

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.

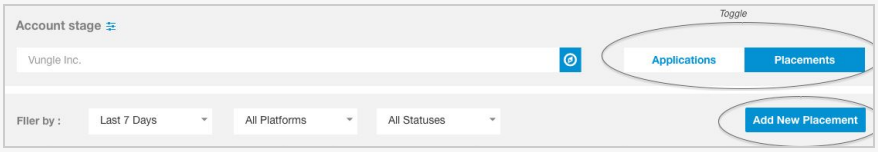
Setting up Placements in your Vungle Dashboard

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.

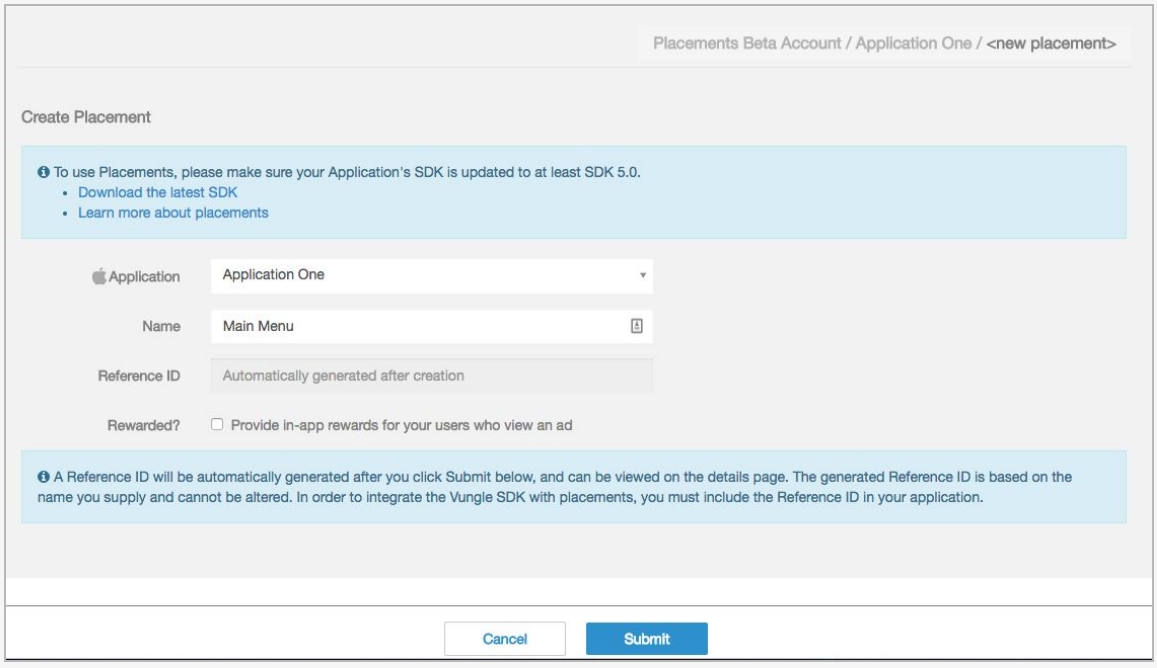
2. There are several ways to get to the Create Placements screen, and we will use the most common one.

    - 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**.



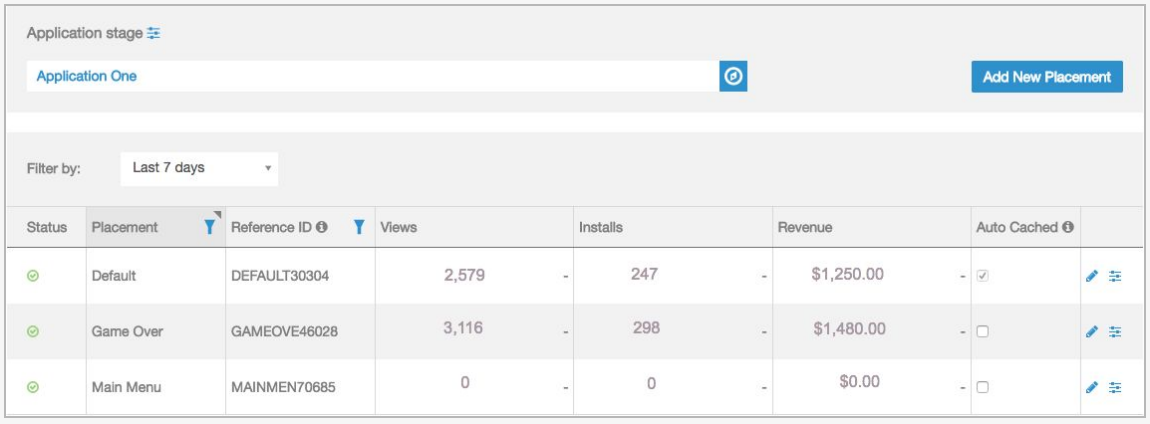
- Another way to create a placement is to choose and app and click on its name. This takes you to the placements page for that app, where you can click **Add New Placement**.
- Finally, you can select an app and click the (edit settings) button to its right, which takes you to the Application Details 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.



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.



Note that:

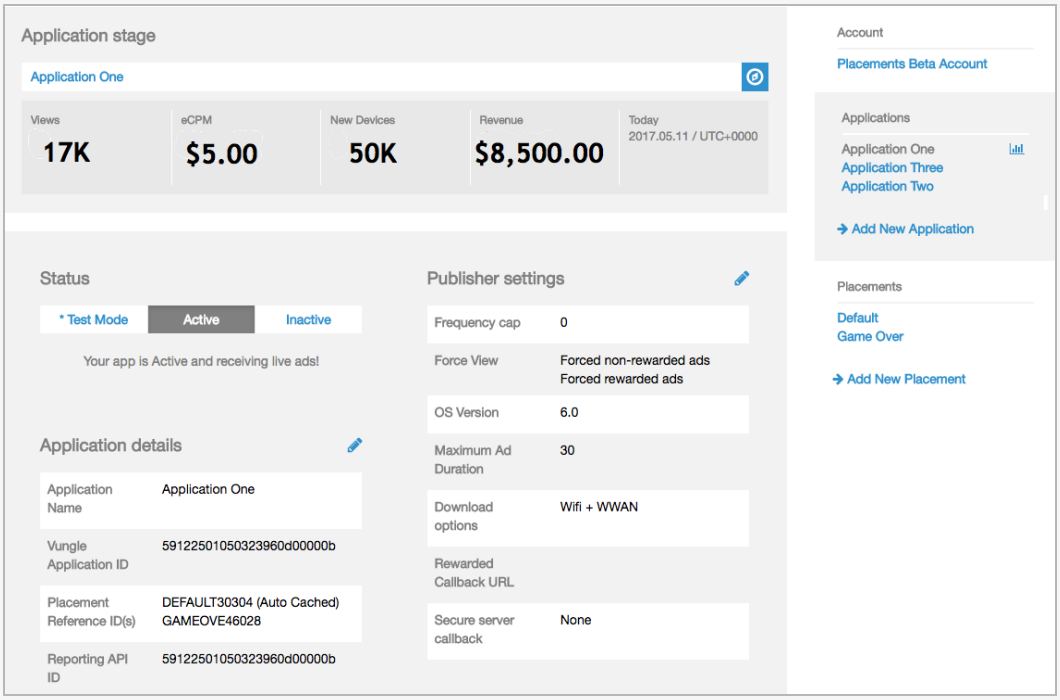
- Vungle will automatically generate a **Reference ID** for each placement you enter; this is the placement's unique identifier and is required information for identifying a placement (for example, if you query the Reporting API and want to filter on a placement, you will need to provide this Reference 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.

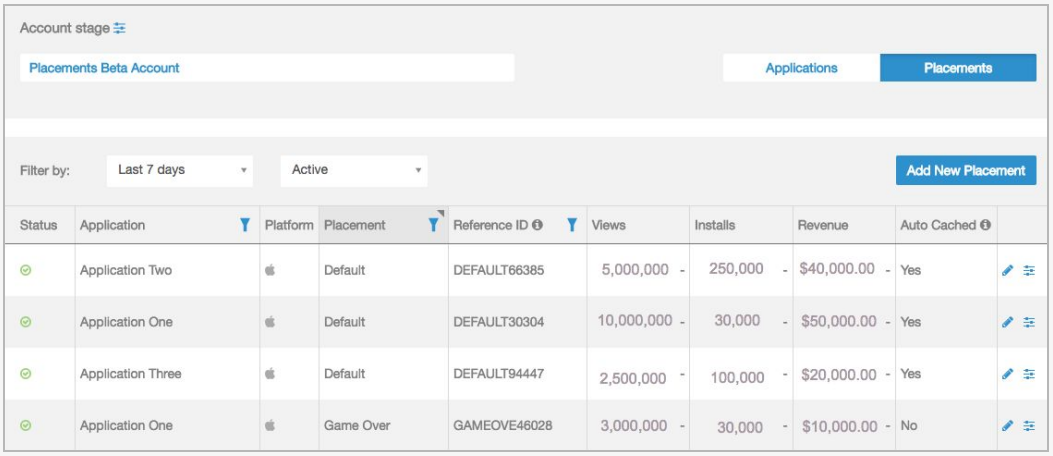
  The first, or default, placement for an app is automatically designated as the auto-cached placement for that app. Once you have multiple placements defined, you can designate the auto-cached one in the placements screens.

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.



Now when you look at your account in the dashboard and toggle to the Placements page, the apps are broken down by placements.

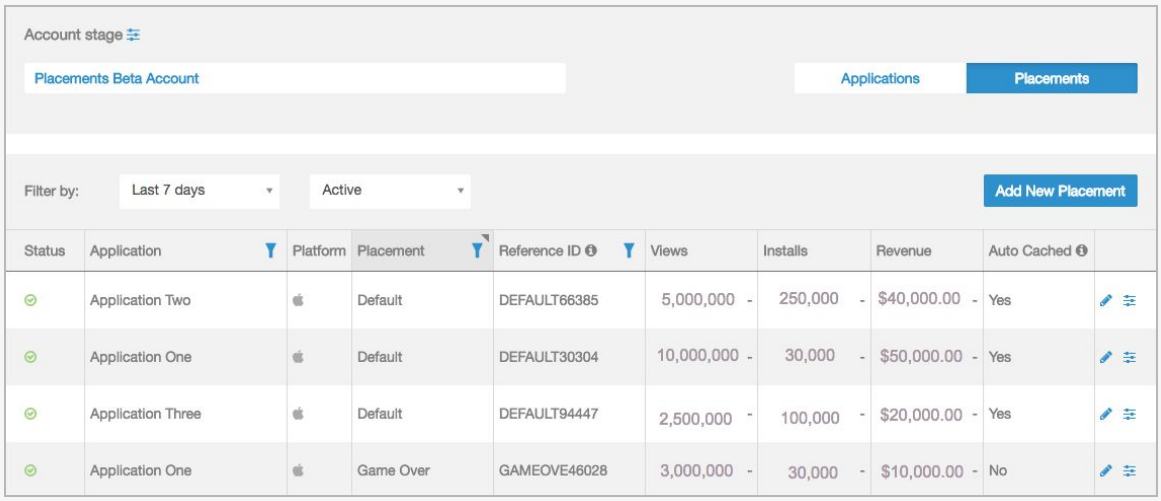


And toggling to the Placements page enables you to review all your placements. You can edit a placement from this page. This is also a good place to find the placement Reference ID, if you need it.

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 Accounts page shows all the placements you have defined for each app, along with the views, new devices, and revenue for each placement.



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.

Integrating the SDK v5.0 for iOS

Step 1. Add the Vungle Framework to your Xcode Project

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).

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**.

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:

```
defaultConfig {
    multiDexEnabled true
}

all projects {
       repositories {
              maven {
                     url “https://jitpack.io”
              }
       …
}
```

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

```
dependencies {
       ….....

       compile 'com.android.support:multidex:1.0.1'
       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:v5.0.0-beta.1-pr7'
    …..
}
```

Step 2. Update AndroidManifest.xml

Add the following lines to your `AndroidManifest.xml`, assigning the application item name to your application class name for multidex:

```
<application
    android:name=”.(YourApplicationName)”
    ...
>


<!-- 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" />
```

Step 3. Enable MultiDex

Create a class file if you don’t have one extended from MultiDexApplication, and add following method:

```
    public class (YourApplicationName) extends MultiDexApplication {

        @override
        protected void attachBaseContext(Context base) {
            MultiDex.install(base);
            super.attachBaseContext(base);
        }
```

```
...
```

Step 4. 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 5. Set the Listeners

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

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

Step 6. 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 `VungleAdEventListener`, 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
clearAndSetEventListeners(VungleEventListener..)	Clears registered EventListeners and then adds the input eventListeners.
clearEventListeners( )	Clears all EventListeners
removeEventListeners(VungleEventListener..)	Removes the input EventListeners.
addEventListeners(VungleEventListener..)	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 placemetReferenceId) {
        // 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 `VunlgeInitListner` 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, placement_list, new VungleInitListener() {

           @Override
           public void onSuccess() {
                // 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 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 immersive mode on KitKat+ devices
setIncentivizedUserId	none	Sets the unique user id to be passed to your application to verify that this user should 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

Integrating the SDK 5.0 for Unity

Before You Begin

Note: The Vungle SDK v5.0 does not currently support the Windows platform.

- The Vungle Unity Plugin for iOS:
    - supports iOS 7
    - supports **Unity 4 and Unity 5.4.1 or higher**.
- The Vungle Unity Plugin for Android:
    - requires Java 1.7 for Android.
    - supports both Unity 4 and Unity 5.

Step 1: Set Up Your Unity Project with the Vungle Unity Plugin

Add the Vungle Unity Plugin to your Unity Project

With your project open in Unity, double-click the downloaded **VunglePlugin.unitypackage** file to add the Vungle Unity Plugin to your application. When the **Import Unity Package** window opens, click **All** to select everything before importing.

Ensure You're Targeting the Correct Platform in Your Build Settings

To avoid compilation errors during the next step, make sure that your project **Build Settings (cmd + Shift + B)** are targeting the iOS or Android platform.

Step 2: Add Code

In this walkthrough we initialize all of our Vungle-related code in a script attached to the main **Game Object**. You can call the **Vungle Unity Plugin** from any scripts you think are appropriate.

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. To initialize the SDK, you will need:

- **all the App IDs for all platforms** you need to support
- **all the Placement IDs you want to use in your app for all platforms** (both active and inactive)

You can find these IDs in the Vungle Dashboard (refer to the "Setting up Placements in your Vungle Dashboard" section of this document).

Sample code:

```
string iOSAppID = "5912326f0e96c1a540000014";
string androidAppID = "591236625b2480ac40000028";
string windowsAppID = "";

#if UNITY_IPHONE
    string placementID1 = "DEFAULT63997";
    string placementID2 = "PLMT02I58266";
    string placementID3 = "PLMT03R65406";
    string appID = "5912326f0e96c1a540000014";
#eli UNITY_ANDROID
    string placementID1 = "DEFAULT18080";
    string placementID2 = "PLMT02I58745";
    string placementID3 = "PLMT03R02739";
    string appID = "591236625b2480ac40000028";
#elif UNITY_WSA_10_0 || UNITY_WINRT_8_1 || UNITY_METRO
#endif
```

```
string[] array = new string[] {placementID1, placementID2, placementID3};
Vungle.init (androidAppID, iOSAppID, windowsAppID, array);
```

Once the SDK is initialized successfully, it calls the following event:

```
public static event Action onInitializeEvent;
```

Refer to the "Event Handling" section of this document.

After the Vungle SDK is initialized, it automatically requests 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 adPlayableEvent event 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 `loadAd` method to load an ad.

```
public static void loadAd(string placementID)
```

Make sure that you are using the `placementID` that is linked to the correct platform.

Sample code:

```
string placementID;
#if UNITY_IPHONE
placementID = <placementID_for_iOS>;
#elif UNITY_ANDROID
placementID = <placementID_for_Android>;
#elif
Vungle.loadAd(string placementID);
```

Check Ad Availability for a Placement

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

```
public static event Action<string, bool> adPlayableEvent;
```

Sample code:

```
Vungle.adPlayableEvent += (placementID, adPlayable) => {
if(placementID == <your_placementID_1>) {
playButtonPlacement1.enabled = adPlayable;
 }
};
```

Note: For the **Auto Cached** placement, this event is called only when an ad becomes available. The SDK will keep requesting an ad for the auto-cached placement. For all other placements, this event is also called in case of “Load Failed” (`adPlayable` returns ‘NO’ in this case).

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

```
public static bool isAdvertAvailable(string placementID);
```

Play an Ad

When there is an ad available for a placement, you can play the ad with the following method:

```
public static void playAd(string placementID);
```

Sample code:

```
Vungle.playAd (<placementID_1>);
```

Event Handling

You can set up EventHandlers for all 5 Vungle SDK events surrounding ad presentation.

- The following event is fired when the SDK starts to play a video ad. This is a great place to pause gameplay, sound effects, animations, etc.

  ```
  public static event Action<string> onAdStartedEvent;
  ```

- The following event is fired when the SDK closes an ad. This is a great place to reward your users and resume gameplay, sound effects, animations, etc.

  ```
  public static event Action<string, AdFinishedEventArgs> onAdFinishedEvent;
  ```

  The `AdFinishedEventArgs` class consists of the following properties for you to check the result of an ad play:

  ```
  public class AdFinishedEventArgs : EventArgs
  {
    //Represents a BOOL whether or not the user clicked the download button.
      public bool WasCallToActionClicked{ get; set;}

    //Represents a bool whether or not the video can be considered a completed
  view.
      public bool IsCompletedView{ get; set;}

      //The time in seconds that the user watched the video.
      public double TimeWatched{ get; set;}
  }
  ```

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

  ```
  public static event Action<string, bool> adPlayableEvent;
  ```

Refer to the "Check Ad Availability for a Placement" section of this document for more detail.

- The following event is fired when the SDK is initialized successfully.

```
public static event Action onInitializeEvent;
```

- The following event is fired when the SDK outputs logs.

```
public static event Action<string> onLogEvent;
```

Sample code:

```
void initializeEventHandlers() {

        Vungle.onAdStartedEvent += (placementID) => {
            DebugLog ("Ad " + placementID + " is starting!  Pause your game
animation or sound here.");
        };

        Vungle.onAdFinishedEvent += (placementID, args) => {
            DebugLog ("Ad finished - placementID " + placementID + " watched
time:" + args.TimeWatched + ", was call to action clicked:" +
args.WasCallToActionClicked +  ", is completed view:"
                        + args.IsCompletedView);
        };

        Vungle.adPlayableEvent += (placementID, adPlayable) => {
            DebugLog ("Ad's playable state has been changed! placementID " +
placementID + ". Now: " + adPlayable);
        };

        Vungle.onLogEvent += (log) => {
            DebugLog ("Log: " + log);
        };

        Vungle.onInitializeEvent += () => {
            adInited = true;
```

```
            DebugLog ("SDK initialized");
        };
    }
```

OnPause and OnResume Functionality

Add code for the `onPause` and `onResume` functionality that enables ads that were paused when an app was backgrounded to resume playing.

```
void OnApplicationPause(bool pauseStatus) {
	if (pauseStatus) {
		Vungle.onPause();
	}
	else {
		Vungle.onResume();
	}
}
```

Customization Options

The `playAd` method can also accept an options dictionary to customize the ad playing experience.

```
public static void playAd(Dictionary<string,object> options, string
placementID);
```

The options dictionary accepts the following keys:

Key	Description

orientation	Sets the orientation of the ad.
	• For iOS, use `VungleAdOrientation`:
	`public enum VungleAdOrientation`
	`{`
	`Portrait = 1,`

	<pre>LandscapeLeft = 2,
	LandscapeRight = 3,
	PortraitUpsideDown = 4,
	Landscape = 5,
	All = 6,
	AllButUpsideDown = 7
	}</pre>
	• For Android, set to `true` for `matchVideo` and `false` for `autoRotate`.

userTag	The user key that is passed to identify users in the S2S call (if there are any).
alertTitle	String that is used as the title of the alert dialog presented when a user closes an incentivized ad experience prematurely.
alertText	String that is used as the body text of the alert dialog presented when a user closes an incentivized ad experience prematurely.
closeText	String title for the close button text of the alert dialog presented when a user closes an incentivized ad experience prematurely.
continueText	String title for the close button text of the alert dialog presented when a user closes an incentivized ad experience prematurely.
immersive	Turn on Immersive mode for Android.

Integrating the SDK 5.0 for Corona

Before You Begin

- **Ads will not work in the Corona simulator. You must build to a device to test our ads.**
- We recommend that you use the latest Corona build for your integration. This guide was written and tested with daily build **2017.3068**. Please contact tech-support@vungle.com if the hosted plugin does not work with the version of Corona SDK you are using.
- The Vungle Corona Plugin for iOS supports iOS 8 and above, limited by the Corona SDK.
- The Vungle Corona Plugin for Android supports Android 4.0.3 (Ice Cream Sandwich - API version 15) and above, limited by the Corona SDK.
- To use our self-hosted plugin, you must have subscription to the Custom Plugin URL feature. Please contact Corona to activate this subscription.
- Download our sample app: https://github.com/Vungle/Corona-Plugin/tree/sdk5.

Step 1: Update build.settings

The Corona Plugin for Vungle SDK v5.0 is not yet available on the official Corona server, but you can use self-hosted plugins from Vungle. Add the following entry into the plugins table of build.settings. When added, the SDK will connect to the Vungle server to integrate the plugin during the build phase:

```
settings = {
    ["CoronaProvider.ads.vungle"] = {
        publisherId = "com.vungle",

        supportedPlatforms = {
            iphone = {
url="https://s3.amazonaws.com/vvv-releases/corona/VungleCoronaiOS-5.0.0.tgz"
},
            android = {
url="https://s3.amazonaws.com/vvv-releases/corona/VungleCoronaAndroid-5.0.0.tg
z" },
            macos = false,
            win32 = false
        },
```

```
    },
},
```

For Android Only

For Android, the following permissions are automatically added when using this plugin:

```
androidPermissions = {
    "android.permission.INTERNET",
    "android.permission.WRITE_EXTERNAL_STORAGE",
    "android.permission.ACCESS_NETWORK_STATE"
},
```

Step 2: Add the Code

Initialize the SDK

We recommend that you initialize the SDK as soon as your app launches to allow the SDK enough time to download ad assets for the auto-cached placement. Initializing the SDK requires:

- importing Vungle ads
- app ID
- all the placement IDs that you will be using in your app

You can find these IDs on the Vungle Dashboard (refer to the "Setting up Placements in your Vungle Dashboard" section of this document).

Sample code:

```
local ads = require "ads"
platform = system.getInfo( "platformName" )
placements = {}

if (platform == "Android") then
    appData = {
        appID="YOUR_ANDROID_APP_ID",
        placements={"PLMT_DEFAULT","PLMT_1","PLMT_2"}
    }
else
    appData = {
        appID="YOUR_IOS_APP_ID",
        placements={"PLMT_DEFAULT","PLMT_1","PLMT_2"}
}
```

```
-- vungleAdListner is optional
ads.init("vungle", appData.appID .. "," .. appData.placements[1] .. ","
.. appData.placements[2] .. "," .. appData.placements[3] [,
vungleAdListener])
```

Once the SDK is initialized successfully, it calls the following event:

```
event.type == "adAvailable"
```

After the Vungle SDK is initialized, it automatically requests 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 **`adAvailable`** event is called with the Placement ID matching your Auto-Cached placement. For the auto-cached placement, only when ad availability changes from true to false or from false to true is this 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" (`adAvailable` returns `'NO'` in this case).

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

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

Load an Ad for a Placement

For placements other than the auto-cached placement call the **`ads.load()`** method to load an ad. It takes a placement ID string as an argument and attempts to load an ad for that particular placement only when **`ads.load()`** is issued and ad is not available for that placement:

```
ads.load( "PLMT_1" )
```

Note: This method is only used for placements other than the auto-cached one, because auto-cached ads are loaded immediately after playing the pre-cached ad.

Play an Ad for a Placement

After allowing enough time for the download of ad assets to complete, you can play the ad by calling **`ads.show()`**. You should check whether there is an ad ready for this placement by calling **`event.adPlayble`** before attempting to play an ad.

Sample code:

```
if (event.placementID == appData.placements[1]) then
    if ( event.adPlayable == true ) then
        ads.show( "PLMT_DEFAULT")
    end
end
```

Event Handling

You can pass optional event listeners to **`ads.init()`** by setting up the listener before the initialization takes place. Below is the list of event listeners available.

- `adStart`
  `type:` `adStart`
  `placementID:` placement ID
  `isError:` `true` if an ad could not be played; `false` if an ad started playing

- `adLog`
  `type:` `adLog`
  `message:` ad activity message

- `adInitialize`
  `type:` `adInitialize`

- `adAvailable`
  `type:` `adAvailable`
  `placementID:` placement ID
  `isAdPlayable:` `true` if an ad is available to played; `false` otherwise

- `adEnd`
  `type:` `adEnd`
  `placementID:` placement ID
  `didDownload:` `true` if the user clicked the download button; `false` otherwise
  `completedView:` `true` if the user watched 80% or more of the video; `false` otherwise

- `vungleSDKlog`
  `type:` `vungleSDKlog`
  `message:` SDK event message

Sample code:

```
local function vungleAdListener( event )
    if ( event.type == "adStart" and not event.isError ) then
        -- adStart event is called and ad will play
    end
    if ( event.type == "adStart" and event.isError ) then
        -- Ad has not finished caching and will not play
    end
    if ( event.type == "adLog") then
        -- adLog event is called to pass ad activity information
    end
    if ( event.type == "adInitialize") then
        -- adInitilizaed is called when placement has successfully initialized
    end
    if ( event.type == "adAvailable" ) then
        -- adAvailable is called when playablility changes to/from true/false
        -- Usage example: setting a flag to true when download completes
        --   Check event.placementID and event.isAdPlayable to set a flag to
true
        --   Then check this flag later in your app and play an ad when it is
true
    end
    if ( event.type == "adEnd" ) then
        -- adEnd is called when the end card is closed and and control is
return to the hosting app


    end
    if ( event.type == "vungleSDKlog" ) then
        -- vungleSDKlog is called when logging event from SDK takes place
    end
end
```

Customization Options

The **`ads.show()`** method can accept an options dictionary to customize the ad play experience.

Key	Value	Description
`incentivized`	Bool	Set to `true` if you want to reward the user for completing a video ad. You can also customize a message to display to users when they attempt to close the video before completion. The following keys are available to provide customized message:

		• `alertTitle` • `alertText` • `alertClose` • `alertContinue` • `placement`
`isAutoRotation`	Bool	For Android, if true (default), the video ad will rotate automatically with the device's orientation. If false, it will use the ad's preferred orientation. For iOS, refer to the `orientation` key below.
`isSoundEnabled`	Bool	If true (default), sound will be enabled during video ad playback, subject to the device's sound settings. If false, video playback will begin muted. Note that the user can mute or unmute sound during playback.
`immersive`	Bool	For Android only, enables or disables immersive mode on KitKat+ devices.
`large`	Bool	For iOS only, draws larger buttons that control ad functions such as mute or close.
`orientation`	Integer	For iOS only, set to 4 for landscape, 0 for portrait, or 5 to rotate automatically.

Sample code:

```
if (platform == "Android") then
    options = {
        placementId = placements[i],
        incentivized = isIncentivized,
        isAutoRotation = isAutoRotate,
        immersive = isImmersive,
        isSoundEnabled = not isMuted
    }
else
    options = {
        placementId = placements[i],
        incentivized = incentivized.isOn,
        large = large.isOn,
        isSoundEnabled = not muted.isOn
    }
end
```

```
if (not isempty(alertTitle.text)) then
            options.alertTitle = alertTitle.text
        end
        if (not isempty(alertText.text)) then
            options.alertText = alertText.text
        end
        if (not isempty(alertClose.text)) then
            options.alertClose = alertClose.text
        end
        if (not isempty(alertContinue.text)) then
            options.alertContinue = alertContinue.text
        end
        if (not isempty(placement.text)) then
            options.placement = placement.text
        end
end

ads.show(options)
```

Integrating the SDK 5.0 for Adobe Air

Follow these instructions to integrate our Vungle Adobe Air Plugin into a basic sample application. The source code we reference here is available on our public GitHub repository.

Before you begin

- **The Vungle Extension Requires Adobe AIR SDK 4.0 or higher.** For instructions on updating the AIR SDK in Flash Builder or Flash Professional, refer to the "Additional Notes" section.
- If you are working with Android, the Vungle AIR extension requires JDK 6 or JDK 7 (depending on the version of Flash you are using) to be installed on the development system. **Android 3.0 (Honeycomb - API version 11) or later is required for the application to run.**
- You can refer to the ActionScript 3 Class Documentation.
- Review `example/VungleExample.as` for a sample application class. (If you're a Flash Professional user and are not sure how to use a Document Class, see "How do I use the VungleExample Document Class in Flash CS6?" at the end of this guide.)

Step 1. Include the Extension Library

Start by creating a new AIR for mobile project and adding the native extension.

If you're targeting Android, you may also need to add Google Play Services library to your project. Because many other extensions already include this library, it may already be present. To add the extension, repeat the steps below, but use **`com.vungle.extensions.android.GooglePlayServices.ane`** in place of `com.vungle.extensions.Vungle.ane`.

For Animate and Flash Professional CS6 or Higher

1. Create a new AIR for Android or AIR for iOS project.
2. Select **File → Publish Settings...**.

3. Select the wrench icon next to **Script for 'ActionScript Settings'**.
4. In the **Library Path** tab, click **Browse for Native Extension (ANE) File** and select the `com.vungle.extensions.Vungle.ane` file. Click **OK**.
5. Select the wrench icon next to **Target for 'Player Settings'**.
6. If targeting Android: In the **Permissions** tab, enable 'INTERNET', 'WRITE_EXTERNAL_STORAGE', and 'ACCESS_NETWORK_STATE'.
7. Select the **Manually manage permissions and manifest additions for this app** option and click **OK**.

For Flash Builder 4.6 or Higher

1. In **Project Properties**, under **Actionscript Build Path**, select **Native Extensions**.
2. Choose **Add ANE...** and navigate to the `com.vungle.extensions.Vungle.ane` file.
3. Select **Actionscript Build Packaging → Google Android**.
4. In the **Native Extensions** tab, select the **Package** option next to the extension.
5. If targeting iOS, repeat steps 3 and 4 for the 'Apple iOS' target.

Step 2. Update Your Application Descriptor

For Vungle to work, changes are required to the application XML file for your app. Modify the XML file created by your IDE with the following changes.

Note: If you're a Flash Professional user, make sure you've followed the steps above for including the extension library "For Animate and Flash Professional CS6 or Higher"; otherwise, Flash may undo your changes as you make them.

1. Set your AIR SDK to 4.0 (or later) in the app descriptor file:

```
<application xmlns="http://ns.adobe.com/air/application/4.0">
```

2. Include a link to the extension in the descriptor:

```
<extensions>
<extensionID>com.vungle.extensions.Vungle</extensionID>
</extensions>
```

3. If targeting Android: you may need to include the Google Play Services extension. Add its extension ID here as well.

```
<extensions>
<extensionID>com.vungle.extensions.Vungle</extensionID>
<extensionID>com.vungle.extensions.android.GooglePlayServices</extensionID>
</extensions>
```

For AIR Applications Targeting Android

If targeting Android, update your Android Manifest Additions in the android XML element to:

- include the `INTERNET`, `WRITE_EXTERNAL_STORAGE`, and `ACCESS_NETWORK_STATE` permissions

- add the `VideoFullScreenAdActivity` and the `MraidFullScreenAdActivity` activity definitions

- add the `google-play-services` version metadata tag:

```
<android>
<manifestAdditions><![CDATA[
<manifest android:installLocation="auto">

 <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>
   <meta-data android:name="com.google.android.gms.version"
android:value="@integer/google_play_services_version"/>

   <activity android:name="com.vungle.publisher.VideoFullScreenAdActivity"
android:configChanges="keyboardHidden|orientation|screenSize|screenLayout|smal
lestScreenSize"
android:configChanges="keyboardHidden|orientation|screenSize|screenLayout|smal
lestScreenSize"
```

```
     android:theme="@android:style/Theme.NoTitleBar.Fullscreen"/>
   <activity android:name="com.vungle.publisher.MraidFullScreenAdActivity"
android:configChanges="keyboardHidden|orientation|screenSize|screenLayout|smal
lestScreenSize"
     android:theme="@android:style/Theme.Translucent.NoTitleBar.Fullscreen"/>
 </application>
</manifest>
]]></manifestAdditions>
</android>
```

Step 3. Integrate the Vungle API

The Vungle API can be added your application in just a few lines of ActionScript.

Initialize the Vungle Extension

Initialize the API when your application starts.

- If using pure ActionScript, do this in the constructor of your Document class.
- If using Flex, call this in the `initialize()` event of the main class.
- If using timeline code in Flash, do this on Frame 1.

1. Import the API Classes:

   ```
   import com.vungle.extensions.*;
   import com.vungle.extensions.events.*;
   ```

2. Initialize the API by calling `Vungle.create()`, and passing in a string of your application ID and an array that contains the placement ID of the application from the Vungle Dashboard. If you are targeting both iOS and Android from the same project, provide different application ID and its placement array per platform to the `create()` method.

   Wrap your call to `Vungle.create()` in a try/catch because Vungle may throw an error during the creation process (for instance, the extension throws an error if running on the desktop):

```
try
{
// initialize with your app id
Vungle.create("your_vungle_id", [“placement1”, “placement2”, “placement3”]);

} catch (error:Error) {
 // could not create extension. Are you running on something besides
iOS/Android?
}
```

Load a Placement Ad

To play a placement ad, you must load an ad for that placement. Note that your auto-cached placement doesn’t require you to call this method. The SDK will try to load the auto-cached placement internally.

```
Vungle.vungle.loadAd(“non_auto_cached_placement”);
```

Play a Placement Ad

You will know when the ad is ready to play through event listeners. Once a placement is available to play, you can play an ad.

```
If (Vungle.vungle.isAdAvailable(“placement_id”)) {
      Vungle.vungle.playAd(“placement_id”);
}
```

Add Event Listeners

The Vungle Extension dispatches four events: `VungleEvent.AD_PLAYABLE`, `VungleEvent.AD_STARTED`, `VungleEvent.AD_FINISHED`, `VungleEvent.AD_FAILED`, `VungleEvent.AD_INIT` and `VungleEvent.AD_LOG`.

1. The `AD_PLAYABLE` is dispatched when an ad is ready to play.

```
Vungle.vungle.addEventListener(VungleEvent.AD_PLAYABLE, onAdPlayable);
function onAdPlayable(e:VungleEvent):void
```

```
    {
       if (e.isAdPlayable) {
             trace(“ad is playable for placement: “ + e.placement);
             Vungle.vungle.playAd(e.placement);
        } else {
                    trace(“ad not playable for placement: “ + e.placement);
       }

}
```

2. The `AD_STARTED` and `AD_FINISHED` events are dispatched when an ad is displayed and dismissed, respectively:

```
Vungle.vungle.addEventListener(VungleEvent.AD_STARTED, onAdStarted);
Vungle.vungle.addEventListener(VungleEvent.AD_FINISHED, onAdFinished);

function onAdStarted(e:VungleEvent):void
{
       trace("ad displayed for placement: " + e.placement);
}

function onAdFinished(e:VungleEvent):void
{
       trace("ad dismissed for placement: “ + e.placement + “, CTA = " +
       e.wasCallToActionClicked);

       if (e.wasSuccessfulView)
       {
       trace("counts a completed view - present reward.");
       }
}
```

3. The `AD_INIT` is dispatched when Vungle SDK has finished initialization.

```
function onAdInit(e:VungleEvent):void
{
       trace(“Vungle SDK is initialized: “ + e.isInitialized”);
}
```

4. The `AD_LOG` is dispatched when a log message is sent by the the Vungle SDK. You can use it for debugging. Logging is implemented only in Vungle SDK for iOS, so this event is platform-specific.

```
Vungle.vungle.setLoggingEnabled(true);
Vungle.vungle.addEventListener(VungleEvent.AD_LOG, onAdLog);

private function onAdLog(e:VungleEvent):void
{
   log("ad log: " + e.message);
}
```

More Options

As you have already seen, you can pass an object with configuration options when calling the `playAd()` method. These are the available properties in `VungleAdConfig`:

`orientation`
With this property you can specify the orientation of the ad. There are different set of flags for Android and iOS. See the `VungleOrientation` class for details. The flags can be combined with a bitwise `OR` operator:

```
config.orientation = VungleOrientation.ANDROID_AUTOROTATE |
VungleOrientation.IOS_PORTRAIT;
```

`soundEnabled`
You can use this property to toggle whether ads will play sound or be muted.

`backButtonImmediatelyEnabled`
This option is Android-specific. If true, it 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.

`immersiveMode`
This option is Android-specific. It enables or disables immersive mode on KitKat+ devices.

`incentivizedUserId`

The unique user ID to be passed to your application to verify that this user should be rewarded for watching an incentivized ad.

`incentivizedCancelDialogTitle`, **`incentivizedCancelDialogBodyText`**, **`incentivizedCancelDialogCloseButtonText`**, **`incentivizedCancelDialogKeepWatchingButtonText`**
These options enable you to customize the confirmation dialog that appears when skipping an incentivized ad.

`extra1` ... **`extra8`**
COMING SOON - You can use this to keep track of metrics such as age group, gender, etc.

Global Defaults

You can use the global configuration object to set default values for the options:

```
// set any configuration options you like
VungleAdConfig.globalConfig.orientation =
VungleOrientation.ANDROID_MATCH_VIDEO;
VungleAdConfig.globalConfig.soundEnabled = false;
```

Then every new `VungleAdConfig` object is created with these values set by default. `playAd()` without options also uses the global configuration.

Note: While your app is in Test mode, you will not be able to download any of the apps advertised. Additionally, the Dashboard will not report the number of impressions. This is because test ads are used only to verify that you have integrated the SDK correctly. This functionality becomes available once your app has gone live in active mode.

Additional Notes

Using the VungleExample.as Document Class in Animate or Flash Professional CS6

1. First, create the application and add the extension by following Steps 1-3 of this integration section.
2. Copy and paste `VungleExample.as` into the same folder as your `.fla`. Do not copy and paste its contents onto the timeline.
3. Change the app IDs on line 51 to be your own Vungle App IDs.
4. In Flash properties, under **Document Class**, type `VungleExample` and click **OK**.
5. Build and install the application.

Installing a Newer Version of the AIR SDK (4.0 or Higher) in Flash Professional CS6

Follow this link to find the latest AIR SDK. If you have already installed AIR 4.0 or higher, you may skip this step. Otherwise, follow the instructions below:

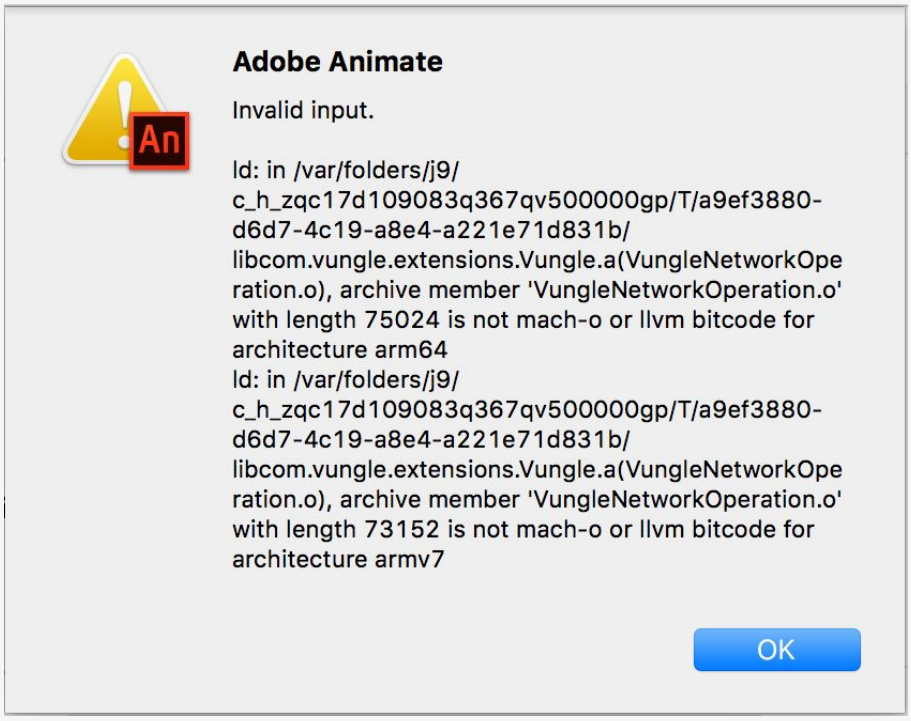
1. Unzip the AIR 4.0 or later SDK package to a location on your hard drive.
2. Launch Flash Professional CS6.
3. Select **Help → Manage AIR SDK...**
4. Click the **+** (plus) button and navigate to the location of the unzipped AIR SDK.
5. Click **OK**.
6. Select **File → Publish Settings**.
7. Select the latest AIR SDK for iOS from the **Target** dropdown menu.

Installing a Newer Version of the AIR SDK (4.0 or Higher) in Flash Builder

Follow this link to find the latest AIR SDK. If you have already installed AIR 4.0 or higher, you may skip this step. You can also make use of Adobe's latest instructions for updating Flash Builder AIR SDKs.

Resolving the 'Adobe Animate Invalid Input' Error

If you are receiving an error as shown in the image below, refer to this article.



Integrating the SDK 5.0 with the MoPub Adapter for iOS

Before You Begin

- **Important:** We recommend that you integrate the Vungle iOS SDK with MoPub 4.14. The Vungle iOS SDK has not been tested with MoPub 4.15.

- The MoPub Dashboard does not yet include Vungle placements; check your placement-level performance using our New Reporting API.

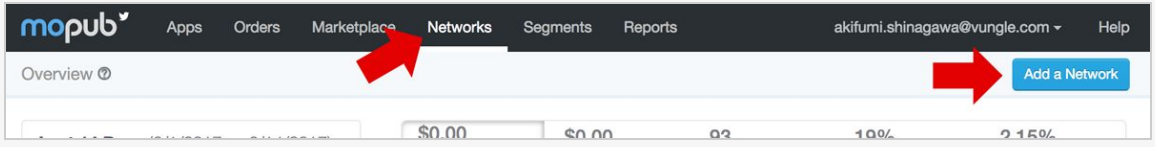
- MoPub must be set up in your app before starting this tutorial. For a step-by-step guide, refer to MoPub's Getting Started Guide for iOS.

- Refer to our MoPub integration documentation on integrating MoPub with your app.
    - Interstitial: https://github.com/mopub/mopub-ios-sdk/wiki/Interstitial-Integration-For-iOS
    - Rewarded Video: https://github.com/mopub/mopub-ios-sdk/wiki/Rewarded-Video-Integration

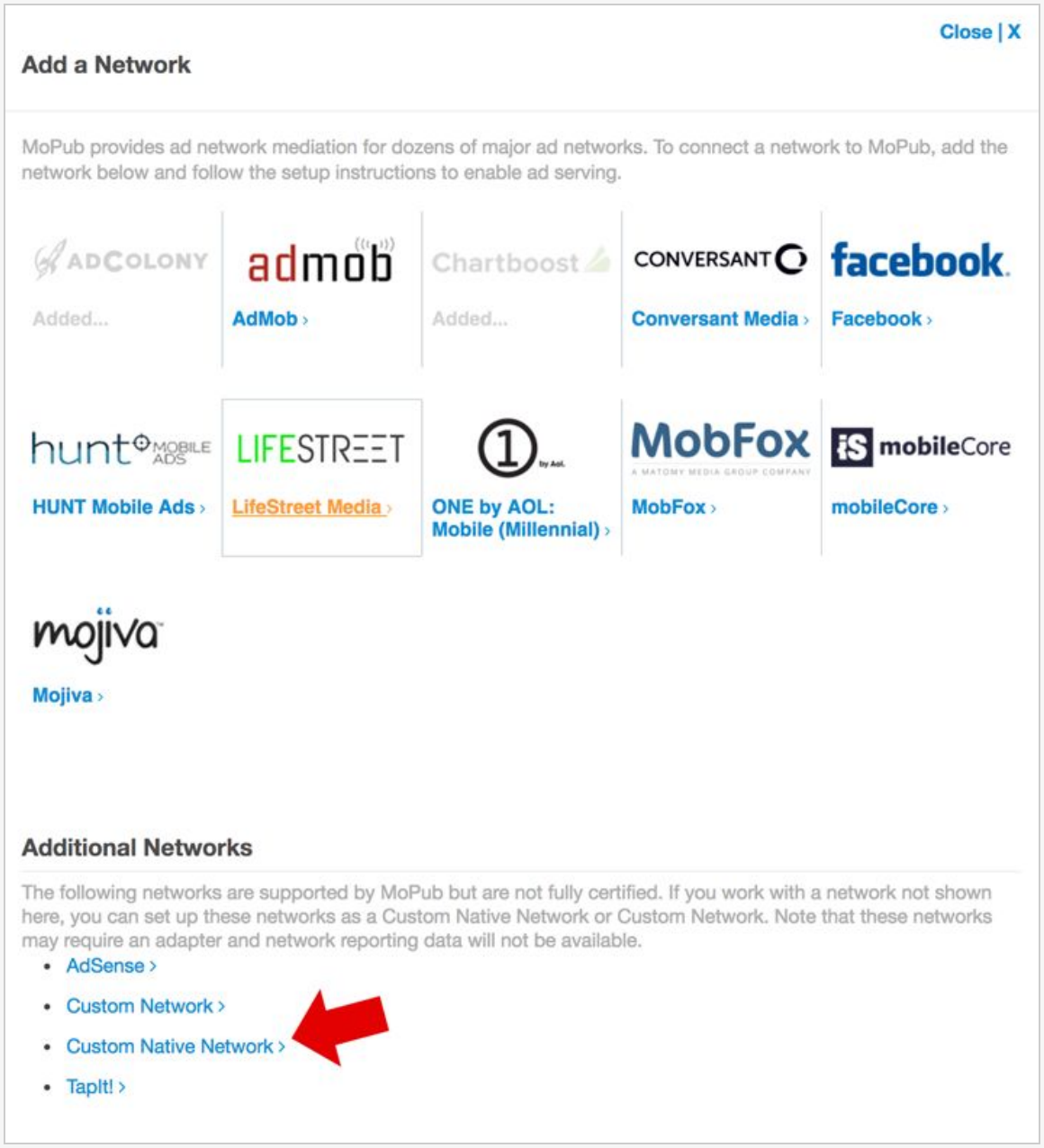
Step 1. Set Up Vungle as a Custom Native Network

To ensure that the MoPub servers recognize the integrated Vungle Adapters, set up Vungle as a Custom Native Network in the MoPub dashboard.

1. Log into the MoPub dashboard, and navigate to **Networks** by the main navigation bar.
2. Click **Add a Network**...



and select **Custom Native Network** at the bottom of the page.



3. In the Set Up Your Inventory section, for each ad unit that will use the Vungle SDK to present video ads, ensure that the MoPub SDK can find the network adapter we used to interact with the Vungle SDK:

    - If you are using **MoPub Rewarded Video** for an Ad Unit, add `VungleRewardedVideoCustomEvent` under the **Custom Event Class** section.

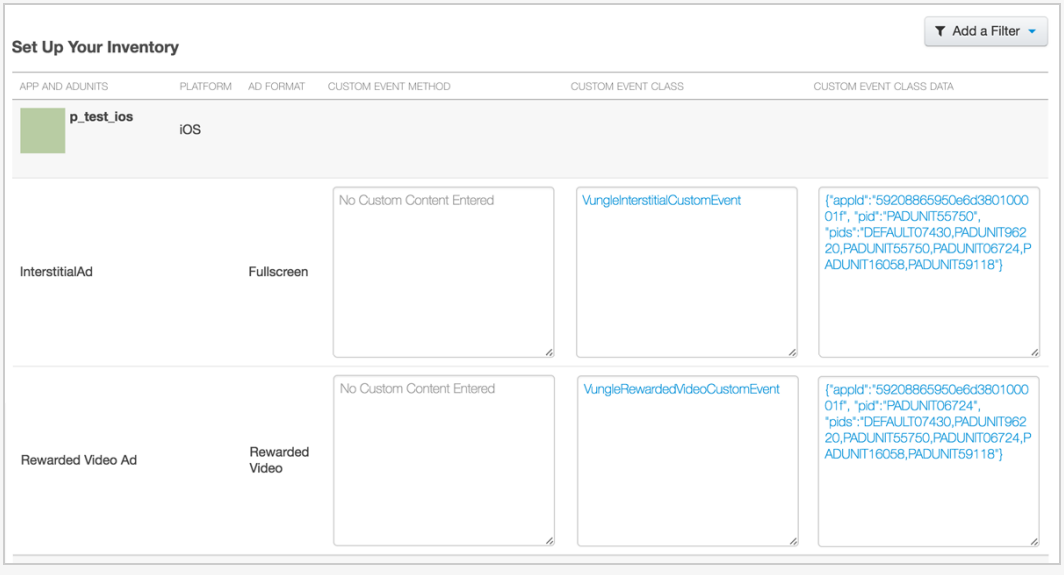
- If you are using **Fullscreen (Interstitial)** for an Ad Unit, add `VungleInterstitialCustomEvent` under the **Custom Event Class** section.

- Add your Vungle **App ID**, **Placement ID**, and all **Placement IDs** in JSON format under the **Custom Event Class Data** section.

- Set the Placement ID you want to link to the MoPub Ad Unit for **pid**.

- Make sure to link the Placement ID set to “Rewarded” to the MoPub Ad Unit that is set to **Rewarded Video**; and link the Placement ID that is NOT set to “Rewarded” to the MoPub Ad Unit is set to **Fullscreen (Interstitial)**.

```
{
"appId":"59208865950e6d380100001f",
"pid":"PADUNIT55750",
"pids":"DEFAULT07430,PADUNIT96220,PADUNIT55750,PADUNIT06724,PADUNIT16058
,PADUNIT59118"
}
```



Note: The Vungle SDK will cache an ad automatically for the placement you selected to be **Auto Cached** in the Vungle Dashboard. We highly recommend selecting the MoPub Ad Unit shown first or the Ad Unit shown most frequently as the **Auto Cached** placement.

Step 2. Add Vungle as a Third-Party Ad Network

To add the Vungle iOS SDK into your app using MoPub mediation:

1. Add all the Vungle-related adapter files (under `/AdNetworkSupport/Vungle/` in Beta MoPub Adapter Repo) to your app's project and make sure they are included in the project's **Build Phases → Compile Sources**.

   ```
   MPInstanceProvider+Vungle.h
   MPInstanceProvider+Vungle.m
   MPVungleRouter.h
   MPVungleRouter.m
   VungleInstanceMediationSettings.h
   VungleInstanceMediationSettings.m
   VungleInterstitialCustomEvent.h
   VungleInterstitialCustomEvent.m
   VungleRewardedVideoCustomEvent.h
   VungleRewardedVideoCustomEvent.m
   ```

2. Copy the `VungleSDK.embeddedframework` folder into `/AdNetworkSupport/Vungle/SDK/`.
   - Make sure the project's **Build Settings → Framework Search Path** includes the `VungleSDK.embeddedframework`. For example: `$(PROJECT_DIR)/AdNetworkSupport/Vungle/SDK/VungleSDK.embeddedframework`
   - Make sure `VungleSDK.embeddedframework` is included in the project's **Build Phases → Link Binary With Libraries**.

3. Follow the instructions in the "Add Other Required Frameworks" section in "Integrating the SDK v5.0 for iOS" of this document to add the necessary frameworks.

4. Follow the instructions in the "Add the "-ObjC" Linker Flag" section in "Integrating the SDK v5.0 for iOS" of this document to add the linker flag.

5. Follow the instructions in the "Remove the iOS Status Bar" section in "Integrating the SDK v5.0 for iOS" of this document o remove the status bar.

6. Call the `MPInterstitialAdController`'s `showFromViewController` method (or, for rewarded ads, call the `MPRewardedVideo`'s `presentRewardedVideoAdForAdUnitID` method) when you want to present a Vungle video ad! (No need to call the Vungle iOS SDK methods directly. The MoPub SDK will do it.)

Integrating the SDK 5.0 with the MoPub Adapter for Android

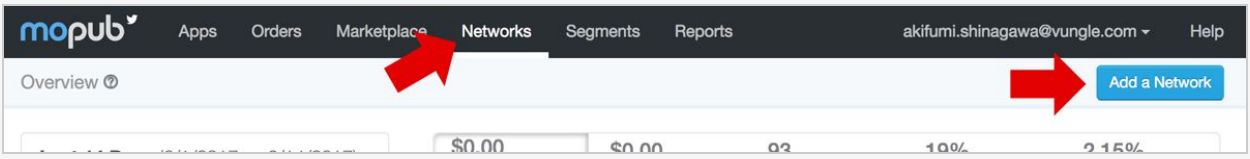
Before You Begin

- The Vungle Android SDK has not been tested with MoPub 4.15 and therefore should only be integrated with MoPub 4.14.
- The MoPub Dashboard does not yet include Vungle placements. You can check placement level performance using our new Reporting API.
- MoPub must be set up in your app before starting this tutorial. For a step-by-step guide, refer to MoPub's Getting Started Guide for Android.
- You will need the Vungle's App ID and all the Placement IDs you want to use in your app. You can find these IDs in the Vungle Dashboard (refer to the "Setting up Placements in your Vungle Dashboard" section of this document).
    - Interstitial: https://github.com/mopub/mopub-android-sdk/wiki/Interstitial-Integration
    - Rewarded Video: https://github.com/mopub/mopub-android-sdk/wiki/Rewarded-Video-Integration
- Vungle SDK requires:
    - 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

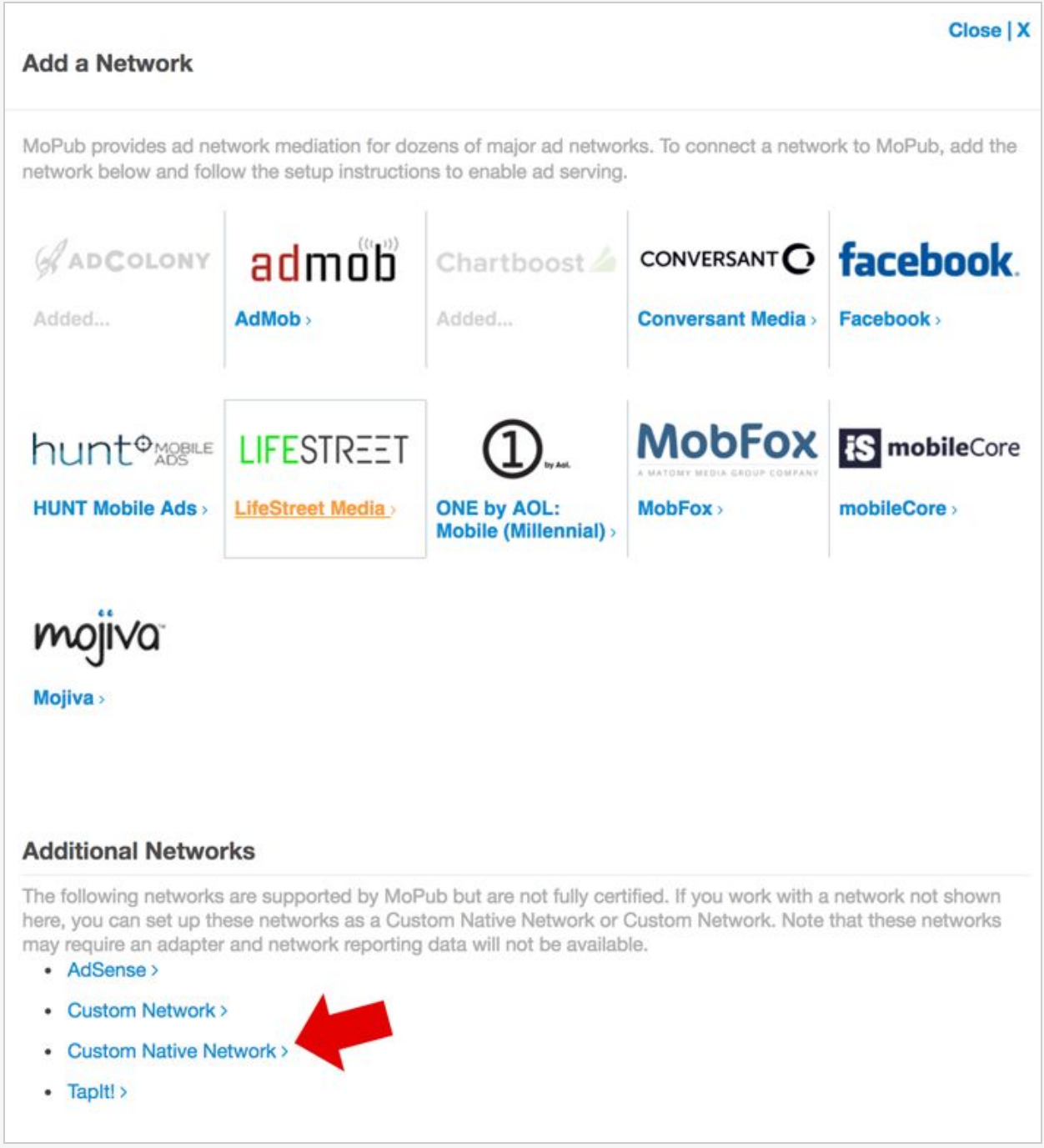
Step 1. Set Up Vungle as a Custom Native Network

To ensure that the MoPub servers recognize the integrated Vungle Adapters, set up Vungle as a Custom Native Network in the MoPub dashboard.

1. Log into the MoPub dashboard, and select **Networks** from the main navigation bar.
2. Click **Add a Network...**.



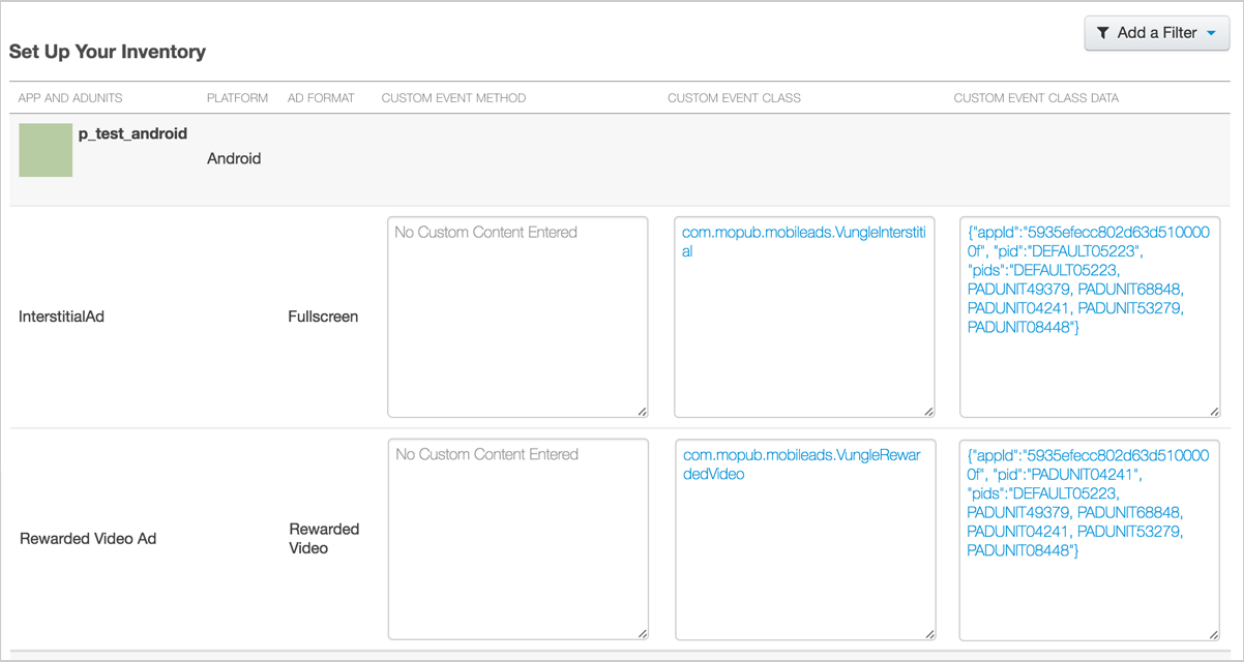
and select **Custom Native Network** at the bottom of the page.



3. In the **Set Up Your Inventory** section, for each ad unit that will use the Vungle SDK to present video ads, ensure that the MoPub SDK can find the network adapter used to interact with the Vungle SDK:

    - If you are using **MoPub Rewarded Video** for an Ad Unit, add `com.mopub.mobileads.VungleRewardedVideo` under the **Custom Event Class** section.
    - If you are using **Fullscreen (Interstitial)** for an Ad Unit, add `com.mopub.mobileads.VungleInterstitial` under the **Custom Event Class** section.
    - Add your **Vungle App ID**, **Placement ID**, and all **Placement IDs** in JSON format under the **Custom Event Class Data** section.
    - Set the Placement ID you want to link to the MoPub Ad Unit for **pid**.
    - Make sure to link the Placement ID set to “Rewarded” to the MoPub Ad Unit that is set to **Rewarded Video**; and link the Placement ID that is NOT set to “Rewarded” to the MoPub Ad Unit that is set to **Fullscreen (Interstitial)**.

    ```
    {
    "appId":"5935efecc802d63d5100000f",
    "pid":"DEFAULT05223",
    "pids":"DEFAULT05223, PADUNIT49379, PADUNIT68848, PADUNIT04241,
    PADUNIT53279, PADUNIT08448"
    }
    ```



Note: The Vungle SDK will cache an ad automatically for the placement you selected to be **Auto Cached** in the Vungle Dashboard. We highly recommend selecting the MoPub Ad Unit shown first or the Ad Unit shown most frequently as the **Auto Cached** placement.

Step 2. Add Vungle as a Third-Party Ad Network

To add the Vungle Android SDK into your app using MoPub mediation:

1. Add all the Vungle-related adapter files (under `/extras/src/com/mopub/mobileads/` in Beta MoPub Adapter Repo) to `/src/main/java/com/mopub/mobileads/` directory in your app's project. This step is necessary because the MoPub SDK references these files as "Custom Event Class properties" in the directory.
    - VungleInterstitial.java
    - VungleRewardedVideo.java

2. Follow the instructions in the "Step 1. Update the Gradle Script" section under "Integrating the SDK v5.0 for Android" in this document to add the necessary frameworks.

3. Follow the instructions in the "Step 2. Update AndroidManifest.xml" section under "Integrating the SDK v5.0 for Android" in this document to update AndroidManifest.xml in your project.

4. Follow the instructions in the "Step 3. Enable MultiDex" section under "Integrating the SDK v5.0 for Android" in this document to enable MultiDex.

5. Call the **MoPubInterstitial's** `show` method (or, for rewarded ads, call the **MoPubRewardedVideos's** `showRewardedVideo` method) when you want to present a Vungle video ad! (No need to call the Vungle Android SDK methods directly. The MoPub SDK will do it.)

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.api.vungle.com	/ext/pub/reports/performance

Authentication

Security is handled in the request header. 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 and version are now passed through a request's header, rather than as a parameter.

Request Headers

Header Key	Header Value	Description
Authorization	Bearer [API KEY]	API Key
Vungle-Version	1	API Version
Accept	• text/csv • application/json	Accepted data format for results. Default is text/csv.

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)
start	ISO8601 date	Limits the result set to performance data no earlier than this date	Reject request	startDate=2017-01-01
end	ISO 8601 date	Limits the result set to performance data no later than this date	Reject request	endDate=2017-01-02
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
placement	Grouped by Placement 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, where each ‘row’ is a JSON object, organised in a JSON array. You can specify which format you prefer in the request header.

Range

The API supports data retrieval from March 1, 2017.

Example

Query

```
curl -i -H "Authorization: Bearer [API KEY]" -H "Vungle-Version:1" -H
"Accept:application/json" -X GET
"https://report.api.vungle.com/ext/pub/reports/performance?dimensions=placemen
t&aggregates=views,revenue&start=2017-03-01&end=2017-03-05"
```

Response

```
[
    {
        "placement id" : "12345678",
```

```
        "placement name": "level 3",
        "views": 1234,
        "revenue": 123.0
    }
]
```