Welcome to Unity Ads

Unity Ads is a video ad network for Apple AppStore, GooglePlay and Xiaomi Mi GameCenter store that can easily be used to monetize your existing player base, and fuel your player acquisition strategy by advertising your game across the world's largest gaming community.

Existing User Accounts

If you have an existing account with Unity Ads, you will need a UDN account to continue using the old Unity Ads admin.

Already have a UDN Account?

If you registered for a Unity Ads account before September 8th 2015, but also have a UDN account registered with the same email address, you should already be setup to use your UDN account to access the Unity Ads admin.

If you are unable to access the Unity Ads admin using the UDN login, please contact us and provide us with your developer ID, your UDN email address or username, and the email address you normally use to log into the Unity Ads admin.

Don't have a UDN account yet?

If you haven't yet created a UDN account, we've created one for you with the same email and password you normally use to login to the Unity Ads admin.

Instead of entering your email and password into the existing login prompt as you normally would, try logging in through the UDN instead. Select UDN Login or Sign Up and follow the prompt.

If you are unable to login using the UDN login, please contact us and provide us with your email address and the developer ID of your account.

New User Accounts

If you don't already have a Unity Ads account setup, you can set one up now by logging in with your UDN account. If you don't already have a UDN account, you'll need to register for one first before you can continue.

Once you've registered for a UDN account, you can complete your Unity Ads account setup by logging into the new Unity Ads dashboard using your UDN login credentials.

If you are logging in through the old Unity Ads admin, select UDN Login or Sign Up.

When you log in for the first time, you will be prompted to complete your Unity Ads account setup. If you are a member of any UDN organizations and are an owner, you will have the option of selecting one of these organizations for use as your Company.

Note: If you are using the old Unity Ads admin, the organization can only be set during the initial registration process. If your role in the organization is set to Owner, the organization be selectable from the Company dropdown field in the sign-up form. If you do not see your organization in the dropdown, please double check your role in the organization. Please contact us if you are still unable to see your organization as an option in the sign-up form.

If you need to change your organization after registration, please contact us and let us know which organization you would like to change it to. Also, please be sure to provide us with your developer ID, and the developer ID of the new organization.

Monetization

Integration Guide for Unity

Quick Start

Before going into too much detail, let's start off with a simple integration example. Every Unity Ads integration boils down to just 3 basic steps:

1. Setup and initialization.
2. Verify ads are ready.
3. Show an ad.

The following example initializes Unity Ads on Start, and shows the *default* ad placement when ready. If the Ads service is enabled in the Services window, Unity will initialize Unity Ads for you instead. The rest of the example is the same, regardless of whether you are using the Services window or the Unity Ads asset package.

C# Example – ShowAdOnStart.cs

```
using UnityEngine;
    using System.Collections;
    using UnityEngine.Advertisements; // Using the Unity Ads namespace.

    public class ShowAdOnStart : MonoBehaviour
```

```
{
    #if !UNITY_ADS // If the Ads service is not enabled...
    public string gameId; // Set this value from the inspector.
    public bool enableTestMode = true;
    #endif

    IEnumerator Start ()
    {
        #if !UNITY_ADS // If the Ads service is not enabled...
        if (Advertisement.isSupported) { // If runtime platform is supported...
            Advertisement.Initialize(gameId, enableTestMode); // ...initialize.
        }
        #endif

        // Wait until Unity Ads is initialized,
        //  and the default ad placement is ready.
        while (!Advertisement.isInitialized || !Advertisement.IsReady()) {
            yield return new WaitForSeconds(0.5f);
        }

        // Show the default ad placement.
        Advertisement.Show();
    }
}
```

JavaScript Example – ShowAdOnStart.js

```
#pragma strict
    import UnityEngine.Advertisements; // Import the Unity Ads namespace.

    #if !UNITY_ADS // If the Ads service is not enabled...
    public var gameId : String; // Set this value from the inspector.
    public var enableTestMode : boolean = true;
    #endif

    function Start () : IEnumerator
    {
        #if !UNITY_ADS // If the Ads service is not enabled...
        if (Advertisement.isSupported) { // If runtime platform is supported...
            Advertisement.Initialize(gameId, enableTestMode); // ...initialize.
        }
        #endif

        // Wait until Unity Ads is initialized,
        //  and the default ad placement is ready.
        while (!Advertisement.isInitialized || !Advertisement.IsReady()) {
            yield WaitForSeconds(0.5);
```

```
    }

    // Show the default ad placement.
    Advertisement.Show();
}
```

Now let's take a closer look at the Unity Ads integration.

Setup and Initialization

Using the Asset Package

This section covers how to setup and initialize Unity Ads using the Unity Ads asset package in Unity 4.3 and later.

To get started with your Unity Ads integration, follow these steps to setup and initialize Unity Ads using the Unity Ads asset package.

Step 1: Set the Platform to either iOS or Android.

1. Select File > Build Settings... from the Unity menu.
2. Select iOS or Android from the Platform list.
3. Select Switch Platform.

Step 2: Download and import the Unity Ads asset package.

Step 3: Create a new GameObject called *UnityAdsExample*.

1. Select GameObject > Create Empty from the Unity menu.
2. Locate the new GameObject in the Hierarchy window.
3. Right-click on the GameObject and select Rename.
4. Enter *UnityAdsExample* and press the return key.

Step 4: Create a new C# script called *UnityAdsExample*.

1. Select Assets > Create > C# Script from the Unity menu.
2. Enter *UnityAdsExample* as the name and press the return key.
3. Right-click the script and select Open to begin editing it.
4. Copy the following example code into the script:

C# Example – UnityAdsExample.cs

```
using UnityEngine;
    using UnityEngine.Advertisements;

    public class UnityAdsInitializer : MonoBehaviour
```

```
{
    [SerializeField]
    private string
        googleGameId = "12345",
         xiaomiGameId = "12345",
        appleGameId = "12345";

    [SerializeField]
    private bool testMode;

    void Start ()
    {
        string gameId = null;

        #if UNITY_ANDROID
        gameId = googleGameId;//For GooglePlay store
         // Or
         gameId = xiaomiGameId;//For Xiaomi Mi GameCenter
        #elif UNITY_IOS
        gameId = appleGameId;
        #endif

        if (Advertisement.isSupported && !Advertisement.isInitialized) {
            Advertisement.Initialize(gameId, testMode);
        }
    }
}
```

Step 5: Add the script to the *UnityAdsExample* GameObject.

1. Select the *UnityAdsExample* GameObject in the Hierarchy window.
2. Select Component > Scripts > Unity Ads Example from the Unity menu.

Step 6: With the GameObject selected, set the Game ID field in the Inspector window.

If you don't know what your Game ID is, you can find it in the list of platforms for your project in the Unity Ads dashboard.

Improving the UnityAdsExample

Let's expand on the initial example by adding support for the following:

- Game ID fields for both iOS and Android.
- An option to enable Test Mode for Unity Ads.

C# Example – UnityAdsExample.cs

```
using UnityEngine;
    using System.Collections;
```

```
using UnityEngine.Advertisements;

public class UnityAdsExample : MonoBehaviour
{

  [SerializeField] private string googleGameId = "12345"; //ID for testing
  [SerializeField] private string xiaomiGameId = "12345"; //ID for testing
   [SerializeField] private string appleGameId = "12345"; //ID for testing

    [SerializeField] bool enableTestMode;

    void Start ()
    {
        string gameId = null;

        #if UNITY_IOS // If build platform is set to iOS...
        gameId = appleGameId;
        #elif UNITY_ANDROID // Else if build platform is set to Android...
        gameId = googleGameId; //For GooglePlay store

         // Or
         gameId = xiaomiGameId; //For Xiaomi Mi GameCenter
        #endif

        if (string.IsNullOrEmpty(gameId)) { // Make sure the Game ID is set.
            Debug.LogError("Failed to initialize Unity Ads. Game ID is null or
empty.");
        } else if (!Advertisement.isSupported) {
            Debug.LogWarning("Unable to initialize Unity Ads. Platform not
supported.");
        } else if (Advertisement.isInitialized) {
            Debug.Log("Unity Ads is already initialized.");
        } else {
            Debug.Log(string.Format("Initialize Unity Ads using Game ID {0} with Test
Mode {1}.", gameId, enableTestMode ? "enabled" : "disabled"));
            Advertisement.Initialize(gameId, enableTestMode);
        }
    }
}
```

Using Test Mode

We strongly recommend enabling Test Mode for Unity Ads during development. While this option is enabled, only test ads will be shown in game. Test ads are not limited to the standard limit of 25 ads per day, and will not affect stats or accidentally generate revenue when clicked on, which is ideal for testing.

Please be aware that you are prohibited under the Unity Ads Terms of Service agreement from generating impressions or installs by clicking on ads within your own game. Doing so will put you at risk of being banned from the Unity Ads network for attempted fraud. As a precaution, it is best to leave Test Mode enabled during development and while testing Unity Ads.

If you are using the Services window in Unity 5.2 or later, Test Mode for Unity Ads will be disabled by default. You can enable or disable this option under Settings for Ads in the Services window.

If you are initializing Unity Ads using the Advertisement.Initialize method instead, Test Mode can be set by passing a boolean value as the second parameter to the method. Please see the Scripting API for further details.

When you get ready to publish your game, Test Ads should be disabled. You can either disable it in Unity before creating your submission build, or you can disable it from the Unity Ads dashboard under under the Settings tab of each platform in your project.

If you are still using the old Unity Ads admin, the Test Mode settings can be found under the Monetization Settings tab of your game profile.

Showing an Ad Placement

This section covers how to simply show an ad placement.

Create a new C# script called *UnityAdsButton*, and add it to a new GameObject in your scene. Then copy the following code into the script:

C# Example – UnityAdsButton.cs

```
using UnityEngine;
    using System.Collections;
    using UnityEngine.Advertisements;

    public class UnityAdsButton : MonoBehaviour
    {
        void OnGUI ()
        {
            Rect buttonRect = new Rect (10, 10, 150, 50);
            string buttonText = Advertisement.IsReady () ? "Show Ad" : "Waiting...";

            if (GUI.Button (buttonRect, buttonText)) {
                Advertisement.Show ();
            }
        }
    }
```

Showing a Rewarded Ad Placement

This section covers how to show a rewarded ad placement, and handle the show result.

Create a new C# script called *UnityAdsRewardedButton*, and add it to a new GameObject in your scene. Then copy the following code into the script:

C# Example – UnityAdsRewardedButton.cs

```
using UnityEngine;
    using System.Collections;
    using UnityEngine.Advertisements;

    public class UnityAdsRewardedButton : MonoBehaviour
    {
        public string zoneId;
        public int rewardQty = 250;

        void OnGUI ()
        {
            if (string.IsNullOrEmpty (zoneId)) zoneId = null;

            Rect buttonRect = new Rect (10, 10, 150, 50);
            string buttonText = Advertisement.IsReady (zoneId) ? "Show Ad" : "Waiting...";

            ShowOptions options = new ShowOptions();
            options.resultCallback = HandleShowResult;

            if (GUI.Button (buttonRect, buttonText)) {
                Advertisement.Show (zoneId, options);
            }
        }

        private void HandleShowResult (ShowResult result)
        {
            switch (result)
            {
            case ShowResult.Finished:
                Debug.Log ("Video completed. User rewarded " + rewardQty + " credits.");
                break;
            case ShowResult.Skipped:
                Debug.LogWarning ("Video was skipped.");
                break;
            case ShowResult.Failed:
                Debug.LogError ("Video failed to show.");
                break;
            }
        }
```

```
}
```

Scripting API Reference

See the Unity Scripting API for Unity Ads:

- Classes
  - UnityEngine.Advertisements.Advertisement
  - UnityEngine.Advertisements.ShowOptions
- Enumerations
  - UnityEngine.Advertisements.Advertisement.DebugLevel
  - UnityEngine.Advertisements.ShowResult

Integration Guide for Xiaomi

This guide will show you how to integrate Unity Ads into an Android Studio project.

Download the latest Unity Ads here.

Quickstart guide

Create a new game project in the Unity Ads dashboard

Log into the Unity Ads Dashboard using your UDN account.

- If you don't have a UDN account yet, sign up [here][52].

From the Dashboard, create a new game project.

Locate the (7-digit) Game ID by selecting your game project. You will use this number to activate Unity Ads.

Locate your *placement IDs* by selecting a Game ID (AppStore, GooglePlay or Mi GameCenter), then clicking the Placements tab. Initially, each game ID has two placements:

- `video` (default / skip video after 5 seconds)
- `rewardedVideo` (no skip option)

Enable test mode *for each game ID* under the settings tab.

Appstore/GooglePlay/MiGameCenter Game ID > Settings > Test mode: Force ON

Integrate Unity Ads in Android Studio:

1. Download the latest release binaries from GitHub, specifically unity-ads.aar.
2. Create or open your existing Android project in Android Studio.
3. Add new module and import unity-ads.aar. Name the module "unity-ads" for example.
4. Open Module Settings for the app and add "unity-ads" module as a dependency.
5. Add the following imports to your java Activity file:
    - import com.unity3d.ads.IUnityAdsListener;
    - import com.unity3d.ads.UnityAds;
6. Add implements IUnityAdsListener to your class, and have Android Studio generate the required callback methods
7. Within the activity that implements IUnityAdsListener, initialize Unity Ads by calling UnityAds.initialize(this, gameId, this) where gameId is a String value set to the game ID of the Android platform, found under your project in the Unity Ads dashboard.
8. *Note: Unity Ads is only initialized once. SDK 2.0 has a more robust network retry logic. So you can safely initialize even without network connectivity. The SDK will request ads when the network becomes available.*
9. The following code will then show an ad for your *default* placement:
10. if (UnityAds.isReady()) { UnityAds.show(this); }

Unity Ads SDK 2.0 takes activity argument in each show method invocation (this in the above example).

See Unity Ads Android API reference for more information on the API, e.g. showing ads for different placements and refer to the example application for an example on how to implement the Unity Ads SDK into your project.

Integrating without Android Studio

If you can't use the AAR packages with your build system, we also provide the same resources in a ZIP file, unity-ads.zip in GitHub releases. You need to do three things to use Unity Ads successfully.

- Include classes.jar in your build.
- Manually merge manifest from AndroidManifest.xml. Make sure you include both AdUnitActivity and AdUnitSoftwareActivity activities. You also need to add INTERNET and ACCESS_NETWORK_STATEpermissions.
- If you are using ProGuard, add all lines from proguard.txt to your ProGuard configuration.

Unity Ads Android Advanced Guides

- API Reference
    - Errors
    - Finish States

For Additional questions, please contact us at unityads-support@unity3d.com

Designing for Video Ads

Unity Ads offers video ads that can be shown as either rewarded or non-rewarded placements. Rewarded ad placements typically yield higher eCPMs (effective Cost Per 1000 Impressions), since they offer more engagement from users by allowing them to opt-in before watching an ad in exchange for some in-game reward. There are a number of ways you can do this, though.

The following article by Unity Ads evangelist Oscar Clark explores several integration examples from successful games, and discusses what not to do as well.

Unity Blog: A Designer's Guide to Using Video Ads

To get an idea of just how much rewarded video ads can impact your game, check out the following blog post. The info graphics within are the distillation of results from the more than 2,000 mobile game developer surveyed.

Unity Blog: Best Practices for Rewarded Video Ads

Please contact us with any questions or comments you might have about designing your game with Unity Ads in mind.

Intro to Ad Placements

What are ad placements

Ad placements allow you to have better control on where and how your ads are shown in the game.

The default placements for your game are:

- default placement: Skipping enabled after 5 seconds
- Rewarded placement: No skipping allowed

When should I use the different placements?

You can show (typically skippable) advertisement videos using the default placement. This is a common practice for interstitials.

If you wish to show rewarded ads - where a user gets a coin/gem/extra life for watching a video, you can use a placement with skipping disabled.

If you need more than the two default placements (for e.g. tracking), you can create new placements in the Ads dashboard.

How to select which placement to show?

You can use the placements by calling them from your game's code. You can also edit or create new placements.

It's up to you when to actually show the ads, but to get an idea about using different placements, take a look at our example game (Space Ads):

https://github.com/Applifier/unity-ads-demo

Code examples for Android

With native SDK integration you must either A) always use the default placement or B) remember to call setZone(""); before each show call

```
if(UnityAds.canShow()){
    UnityAds.setZone("PLACEMENT ID");
    UnityAds.show(options);
}
```

Advertising

Getting Started

Self-Service User Acquisition

This document serves to help marketers quickly get started with user acquisition campaigns through

Sign Up and Verify Account

The first step in getting started with Unity Ads is to create an account. You can do this by following the steps below:

Step 1: Create a Unity Developer Network account

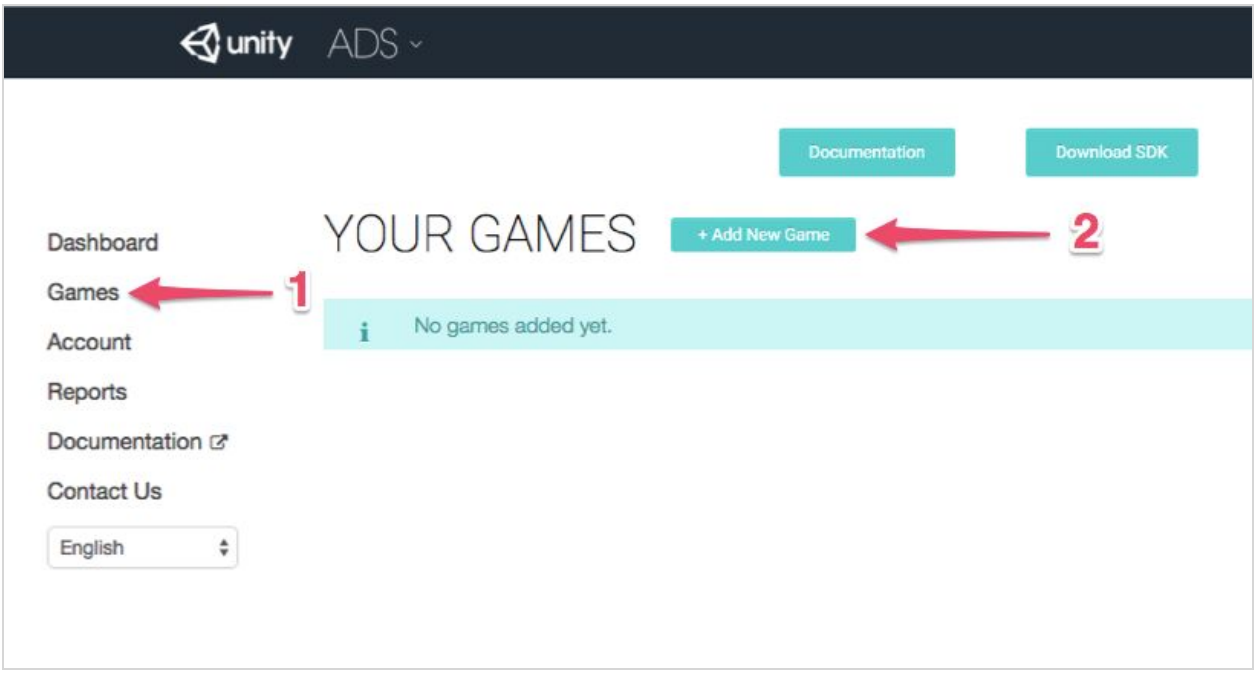
Step 2: Verify your UDN account via email

Step 3: Sign in to the UDN

Step 4: Head over to the Unity Ads signup page and register for the Unity Ads service

Add a Game

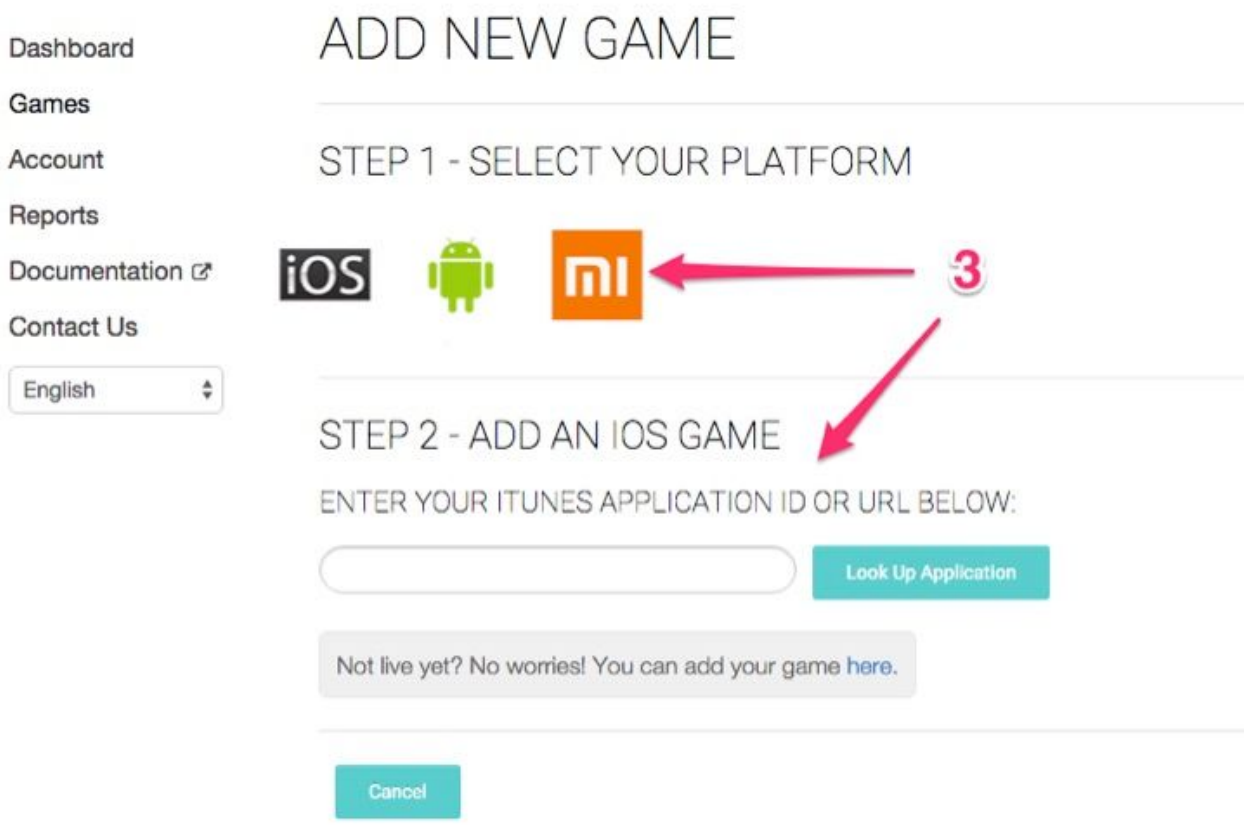
The next step after creating and verifying your account is to add a game. All campaigns and creatives will be stored at the game level in the Unity Ads dashboard.



Step 1: Select "Games" from the left-hand side of the dashboard

Step 2: Hit "Add New Game" to begin the process of adding your game

Step 3: Select which platform your game is live on and input your game's store URL. Then, scrape the app store using the "Look Up Application" button

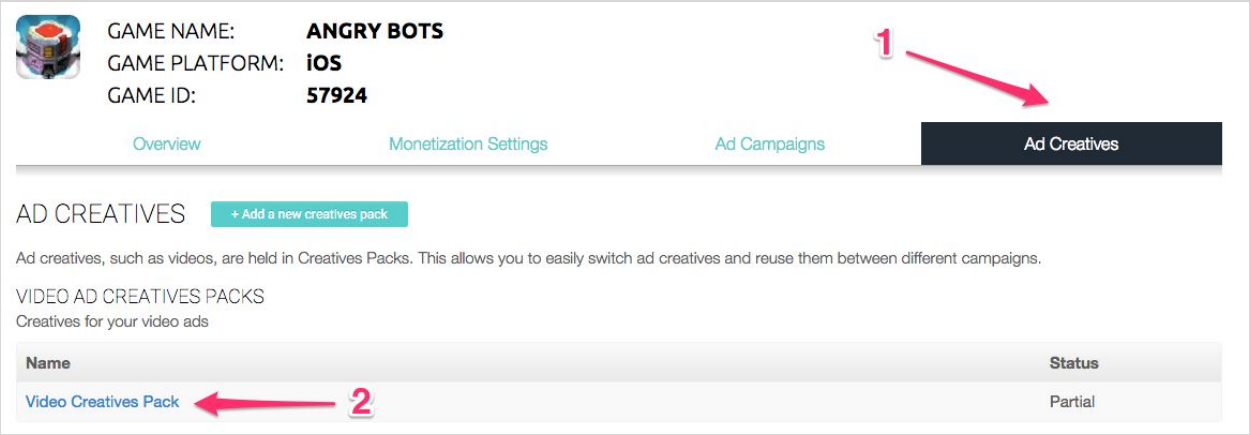


Step 4: Review your game's information, and hit "Continue"

Add Video Creatives

After you've added your game, the next step in starting a UA campaign will be to add the creative assets. These are the videos and end cards that will be shown when the campaigns have begun.

Step 1: Head into the game you just added. To do this, select "Games" from the left-hand side of the dashboard, select the game by clicking on the game name, and then head over to the "Ad Creatives"

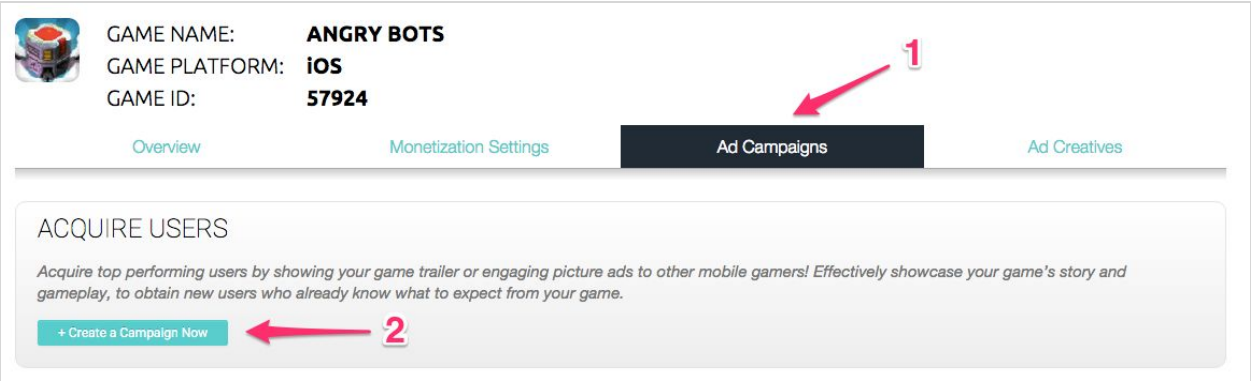


Step 2: By default, we create an empty creative pack for you. Select this creative pack by clicking on "Video Creative Packs"

Step 3: Once you're inside of the default creative pack, you can drag and drop your video and end cards into their respective field. For more information on our creative specs, check out the Video Creative Documentation. Make sure to save your creative pack after everything has uploaded properly (you'll see "Asset uploaded successfully")

Set Up a Campaign

Now that you've added your game to the dashboard and creatives to your game, its time to set up a new campaign.



Step 1: Head back into your game, but this time select "Ad Campaigns", then select "+ Create a Campaign Now" and go through the campaign set up process:

Campaign Name: The name of your campaign (required)

Ad Type: The type of ad you'd like to run on our network (required - only video ads available at this time)

Campaign Type: The payment method you'd like to use for this campaign

- CPI - Cost per Install. You will only be charged when a user installs your app (recommended)
- CPV - Cost per Completed View. You will be charged every time a user completely watches one of your video ads

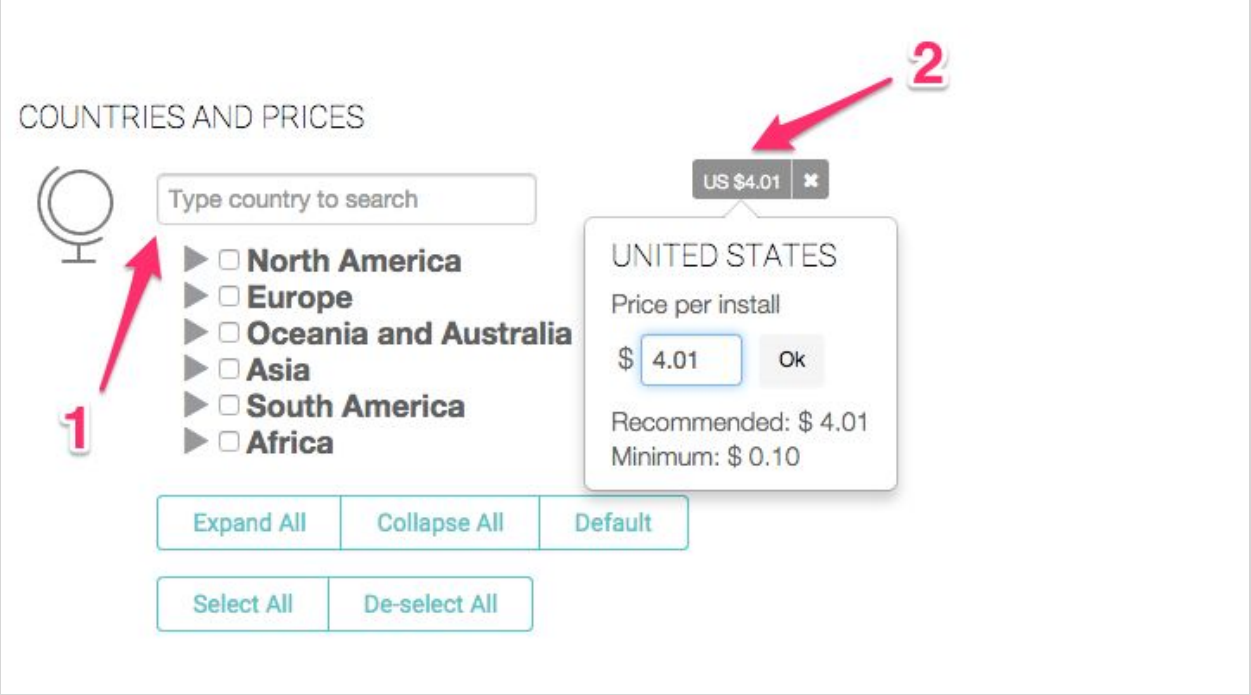
Budget Type: The type of budget you'd like to use for this campaign

- Single budget - this budget input will only be used for this campaign
- Shared budget - you can use this option if you wanted more than one campaign to share a single budget

Scheduling: The start and end date that you want to run your campaign for. By default, there won't be an end date

Creatives Pack: This is the video that will run on this specific campaign. You can select "Change" to change to a different creative pack. Keep in mind that you'll only see one creative pack by default, unless you've added more than one creative pack.

Countries and Prices: In this section, you can change the countries that you want this campaign to target, as well as set the prices for each of those countries. First, click on "Countries and Prices", which will open up the expanded targeting window.



Search the country you want to target by using the "Type country to search" box and hit enter to add that country to the targeting list. You can then change the bid by clicking on the country code box. This will bring up a pop-up window where you can change the bid. Hit "OK" to save the bid for that country.

Targeting:

- Operating System Targeting: The minimum and maximum Operating System targeting parameter allows you to target a specific range of operating systems
- Device Targeting: The device targeting section allows you to target a series of devices (iPad, iPhone, iPad) or a specific device (iPhone 6, iPad Mini 3). This granular targeting is only available on iOS, but will be coming to Android in the near future. Android targeting is broken down by screen size.
- Connection Targeting: This allows you to target users that are connected to WiFi or a cellular connection. If your game is above the download limits (iOS - 100mb/ Google Play - 50mb), then we suggest targeting WiFi only, as users on cellular wouldn't be able to download your game (which lowers conversion rates and makes you less competitive in our network).

Designing Campaigns

This design guide outlines the requirements for creating a Unity Ads campaign.

The following assets must be included with each campaign:

- A video advertising your game (length: min. 15s, max. 30s).
- One landscape (800×600 pixels) promotional graphic with 100-pixel crop zones on left and right sides.
- One portrait (600×800 pixels) promotional graphic with 100-pixel crop zones at the top and bottom.

Video Asset

The video used to promote your game should be between 15 and 30 seconds in length, with a widescreen resolution of 640×360 pixels (16:9 aspect ratio).

We do support a wide range of video formats. However, to help ensure that your video upload is accepted by us, please use one of the recommended formats listed below.

Video Specifications

- *Resolution:* 640×360 pixels (16:9).
- *Length:* 15 to 30 seconds.
- *File Format:* H.264-encoded MOV, MP4, or AVI file formats.
- *File Size:* Recommended size is 10MB. Maximum size is 30MB. Videos will be re-encode to be suitable for various bitrates. The final video shown will be optimized for the user's available network speed and cache settings.
- *For iOS Only:* If the video contains more than one store logo, *App Store logo should always appear in first position.*

End Card Asset

When the video ends, the user is presented with an end screen that shows a large graphic of your game, with an call-to-action button prompting the user to download the game.

End Card Specifications

- *Resolution:* There are two required end cards. A portrait and landscape version:
    - *800×600px:* The landscape version of the end card. Please include a 100px crop zone on the left and right margins (no important information on left or right side, 100px deep).
    - *600×800px:* The portrait version of the end card. Please include a 100px crop zone on the top and bottom margins (no important information on top or bottom, 100px deep).
- *File Format:* JPG or PNG.
- *For iOS Only:* If the video end card contains more than one store logo (App Store, Google Play, etc...), *App Store logo should always appear in first position.*

Optimizing Campaigns

When launching a user acquisition campaign, you want to be sure that you are utilizing all possible levers at your disposal in order to get the most performance out of your campaigns. Of course, raising your bids to competitive levels is the most obvious and one of the most effective ways of increasing volume. However, there are several other ways to optimize without touching your campaign spend. Here we will go over some of the best practices you can implement in order to achieve such results.

Create a Strong Trailer

Having a trailer that resonates with your target audience is the key factor in boosting your conversion rate. This is especially important because conversion rate is one of the most important measures utilized by our algorithm to determine your ad ranking, and thus your visibility and performance on our network.

General characteristics of a strong video trailer include:

- Meaningful, action-packed gameplay footage
- Strong calls-to-action
- References to game dynamics that make your game unique
- Localized in geo-specific languages (where applicable)
- High production value
- Voiceovers/speaking characters/sound effects

Rotate Your Trailers

Users eventually experience ad fatigue when exposed to the same video trailer over and over again. This contributes to a decline in conversion rate and thus a decline in overall volume. Combat this effect by rotating in a new trailer with fresh content approximately once a quarter, if resources permit.

Fine-Tune Your Campaign Targeting

Segmenting your campaigns is a great way to add precision to your user acquisition efforts. Some factors to consider include:

- Choose your geo targets carefully. You might already know what countries perform best for you, and you can assign each a different bid accordingly. If you are just starting out with user acquisition and have not gathered enough data to ascertain what performs best, we recommend first launching in the US, United Kingdom, Canada, and Australia. As you gather data and want to explore more opportunities, you might consider expanding to other English-speaking countries, countries in Western Europe, and the Nordic countries. Conduct periodic reviews of per-country performance and adjust bids from time to time.
- Break out your iOS campaigns to target different bids for iPad and iPhone. iPad users generally tend to be higher quality users than iPhone, so we recommend a higher bid there.
- If you are targeting non-English speaking countries, it is best to run localized versions of your trailer and static thumbnails/endcards.

Remove Underperforming Publishers

You have the ability to track which publishers are delivering you the best and worst quality installs, and target accordingly. By adding a parameter to your tracking URL, we will pass you a 5-digit publisher ID. The macro for this parameter is “{source_game_id}”. Please consult your tracking service for the corresponding parameter name.

Once this parameter is in place, wait until you have collected a statistically significant amount of internal data regarding publisher performance. Then, analyze the user quality data per publisher according to your

key metrics (LTV, ARPU, etc). Send Client Services a list of the low quality publishers, and we can block these for you.

Advertising Stats API

Unity Ads provides an API for Publishers and Advertisers to retrieve monetization and acquisition statistics data directly in CSV format. The Unity Ads Statistics API can used to retrieve the game and campaign data for loading them up to partners own reporting system periodically. In practice, the statistics API is a machine-to-machine interface for fetching the same statistics files that have been available in the Unity Ads Admin Panel automatically.

Overview

The statistics API works in two stages: First, the user performs an GET request to an authentication server which, after a successful authentication, responds with a 302 HTTP redirect message. This response contains a the final URL to the statistics server.

Next, when the user performs an GET request to the signed URL, the located server will respond with the requested data in CSV format in the body of the message:

```
Date,Target campaign id,Target name,clicks
2013-02-28 00:00:00,"5065e1f1fdeb285e4d0430ce","Campaign 1",129
2013-02-28 00:00:00,"50ed569d57fe1e324415fbf7","Campaign 2",428
2013-02-28 00:00:00,"50eeb7c39610c9d21c0225cb","Campaign 3",812
2013-02-28 00:00:00,"511e5f7a73452a3363062d5d","Campaign 4",130
...
```

Authentication

In order to use the Unity Ads Statistics API, you need to get the API key from the Unity Ads Admin Panel. The API key is located in the Account Settings page.

The API key needs to be placed in the authentication request to the `apikey` HTTP GET parameter.

After a successfull authentication the server will respond with a 302 HTTP-redirect message with the data URL to the statistics server located in the `Location` HTTP-header. The real data is fetched from this given redirect URL. This is a standard HTTP behavior and is supported by all HTTP clients. For example `curl -L "http://gameads-admin.applifier.com/stats/acquisition-api?apikey=APIKEY"` will directly output the file to the console.

The statistics server always requires signed URLs and will not work if accessed without a valid signature. If the authentication fails, the authentication server will respond with an HTTP/1.1 200 OK header with an error message in the body:

```
{"error":"Authentication error","responseCode":500,"status":"error"}
```

Request Format

Monetization statistics (Use this API to get statistics from your monetising games)

See separate document about monetisation stats API here: Unity Ads statistics API for monetisation

Acquisition statistics (Use this API to get statistics from your advertising campaigns)

The acquisition statistics API is similar to the monetization one and supports the following request format:

```
http://gameads-admin.applifier.com/stats/acquisition-api?apikey=<apikey>&fields=<fields>[&splitBy=<splitbyfields>][&scale=<scale>][&start=<startDate>][&end=<endDate>]([&targetIds=<targetIds>]|[&campaigns=<campaignIds>]
```

where:

- `<apikey>` is the api authentication key retrieved from the Unity Ads Admin Panel
- `<fields>` contains a comma separated list of available fields:
    - started - number of impressions recorded
    - views - number of completed views recorded
    - clicks – number of clicks recorded
    - installs – number of installs recorded
    - spend – how much money was spent

The default set of fields is all of the above: "`clicks`,`installs`,`spend`".

- `<splitbyfields>` contains a comma-separated list of dimension in which to split the data:
    - `target` – the data is split by target games
    - `campaign` – the data is split by campaigns
    - `country` – the data is split by users' country

The default split by is `country`. If you don't want to split the data at all you can write `splitBy=none`. Either target or campaign split is allowed at the same time. Both can be used with or without country split.

Note: Splitting the statistics data across multiple dimensions at the same time grows the data size exponentially. The processing time might end up being too long and cause the request to fail. All the requests taking more than a minute to generate the data will be terminated at 60 seconds.

- `<scale>` – contains the time resolution of the data. Each day is split at `00:00 UTC`. The available values for scale are:
    - `all` – removes time resolution completely and provides the total sum of values within the defined period
    - `hour`
    - `day`
    - `week`
    - `month`
    - `quarter`
    - `year`

The default is `day`.

- `<startDate>` & `<endDate>` – contains the start (inclusive) and end (exclusive) times for the data. The dates are accepted in following formats:
    - negative numbers are treated as days relative to the current date. For example: `-7` is a week ago.
    - Date string in ISO format `YYYY-MM-DDTHH:mm:SS:hhhZ`, for example: `2013-02-01T14:00:00.000Z`

The default start date is `-7` and end date is `0` to get the past week's data.

Note: The start and end dates will be rounded upwards to next full hour. For example, the `14:00:05.000Z` will be rounded to `15:00:00.000Z`. Also note that if using scale `day`, using non-midnight `<startDate>` & `<endDate>` will still select the range per hour and not the whole day.

- `<targetIds>` – comma separated list of target game ids to filter the results. By default, all the games of the advertiser will be included.
- `<campaigns>` – comma separated list of campaign ids to filter the results. By default, all the campaigns of the advertiser will be included.
- Note: The results can be filtered by either target games or campaigns, but not both at the same time.

Example:

```
curl -L
"http://gameads-admin.applifier.com/stats/acquisition-api?apikey=c4ca4238a0b923820dcc509a6f758
49bc81e728d9d4c2f636f067f89cc14862c&splitBy=campaign,country&fields=views,clicks&start=-31&sca
```

```
le=all&targetIds=8234,7432"
```

Response Format

CSV format in general

The statistics server will output the requested statistics data in the HTTP response body. The data is in CSV format. Comma ‘,’ is used as the field separator and dot ‘.’ as the decimal separator. Text fields will have double quotes (‘”‘) around them. Pure integer fields don’t have double quotes, but decimal numbers (such as revenue and spend) are doublequoted. The unix newline character (0x0D) is used as the line separator. The first line of the output will contain the field names.

Fields

The leftmost field is always the Date. The date is in ISO format YYYY-MM-DD HH:mm:SS.

The next fields are the split dimensions (if selected) in the following order:

- For source splitted data, two fields "Source game id" and "Source game name" will be shown. Game id is an integer.
- For target splitted data, two fields "Target game id" and "Target name" will be shown.
- For campaign splitted data, two fields "Target campaign id" and "Target name" will be shown.
- For country splitted data, two fields "Country code" and "Country tier" will be shown. Country code is the ISO 3166-1 alpha-2 country codes in upper case based the users’ GeoIP data. "--" is shown for those user IPs whose GeoIP location was unknown. - - The country tier is an integer denoting country’s classification in Applifier’s tier structure.

The rightmost fields are the data fields requested in the same order as in the request.

For example:

```
$ curl -L
"http://gameads-admin.applifier.com/stats/acquisition-api?apikey=c4ca4238a0b923820dcc509a6f758
49bc81e728d9d4c2f636f067f89cc14862c&splitBy=country,campaign"
Date,Target campaign id,Target name,Country code,Country tier,clicks,installs,spend
2013-03-01 00:00:00,"50ed569d57fe1f324a15fbf7","Campaign #5","AU",2,71,30,"45.00"
2013-03-01 00:00:00,"50ed569d57fe1f324a15fbf7","Campaign #5","CA",2,129,88,"132.00"
2013-03-01 00:00:00,"50ed569d57fe1f324a15fbf7","Campaign #5","US",1,1745,855,1282.50
2013-03-01 00:00:00,"50eeb7339e10c9d21c0225cb","Campaign #6","AT",3,39,19,"28.50"
2013-03-01 00:00:00,"50eeb7339e10c9d21c0225cb","Campaign #6","AU",2,16,10,"15.00"
2013-03-01 00:00:00,"50eeb7339e10c9d21c0225cb","Campaign #6","BE",3,209,120,"180.00"
2013-03-01 00:00:00,"50eeb7339e10c9d21c0225cb","Campaign #6","CA",2,284,179,"268.50"
2013-03-01 00:00:00,"50eeb7339e10c9d21c0225cb","Campaign #6","CH",3,15,7,"10.50"
...
```

Unity Ads Install Tracking SDK

Get the UnityAdsInstallTracking package from your account manager.

Obtain your game id from the advertiser dashboard or your account manager. Ensure that you are using the correct ids for each app being published in different store (Apple AppStore, GooglePlay, Xiaomi Mi GameCenter). For Xiaomi integrations please follow the instructions for Android or Unity package integration.

iOS

Extract the `UnityAdsInstallTracking.framework.zip` file and add the resulting `UnityAdsInstallTracking.framework` to your Xcode project app target.

In the `applicationDidFinishLaunching` method in your app delegate, add a call to `UnityAdsInstallTracking.track`

Objective-C:
```
[UnityAdsInstallTracking track:@"YOUR_APPSTORE_GAME_ID"];
```
Swift:
```
UnityAdsInstallTracking.track("YOUR_APPSTORE_GAME_ID")
```

Android

Add the `UnityAdsInstallTracking.aar` file to your Android project.

In the `onCreate` function of your main activity add a call to `UnityAdsInstallTracking.track`:
```
UnityAdsInstallTracking.track(self, "YOUR_GOOGLEPLAY_GAME_ID"); // For GooglePlay store
```
Or
```
UnityAdsInstallTracking.track(self, "YOUR_MIGAMECENTER_GAME_ID"); //For Xiaomi store
```

Unity Package

Import generated ```UnityAdsInstallTracking.unitypackage``` to your project.

For simple testing, use the following script & attach it to a game object in your scene:

`InstallTrackingBehaviourScript.cs`

```
```
using UnityEngine;
using UnityEngine.InstallTracking;

public class InstallTrackingBehaviourScript : MonoBehaviour {
	void Awake() {
		#if UNITY_IOS
		Tracker.Track("YOUR_APPSTORE_GAME_ID");
		#elif UNITY_ANDROID
		Tracker.Track("YOUR_GOOGLEPLAY_GAME_ID"); //For GooglePlay store
		//Or
		Tracker.Track("YOUR_MIGAMECENTER_GAME_ID"); //For Xiaomi store
		#endif
	}
}
```
```