Unity Plug-in for iOS

AdColony Version 1.9.7

Updated April 6, 2012

Table of Contents

Introduction

AdColony 1.9.7 features high-definition (HD), Instant-Play™ video advertisements that can be played anywhere within your application. Video ads may require a brief waiting time on the first attempt to play; after which, the videos will play without any delay. AdColony also contains a secure system for rewarding users with virtual currency upon the completion of video plays. In addition, AdColony provides comprehensive app analytics and campaign metric reporting, visible in your account on adcolony.com.

This document describes how to easily integrate AdColony into your Unity applications and quickly add video advertisements and virtual currency rewards. If you need more information about any of these steps, consult our sample applications or contact us directly for support. We are dedicated to providing quick answers and friendly support.

Support E-mail: support@adcolony.com

Changes

April 6, 2012
Updated plug-in to use AdColony 1.9.9, which removes reliance on the iOS UDID. AdColony iOS now uses a combination of MAC Address and OpenUDID to track ad delivery. The plugin has two new methods if your app needs to access either of those identifiers: AdColony.GetDeviceID() return the MAC Address and AdColony.GetOpenUDID() returns the OpenUDID.

November 3, 2011
This is the first release of the Unity AdColony plug-in.

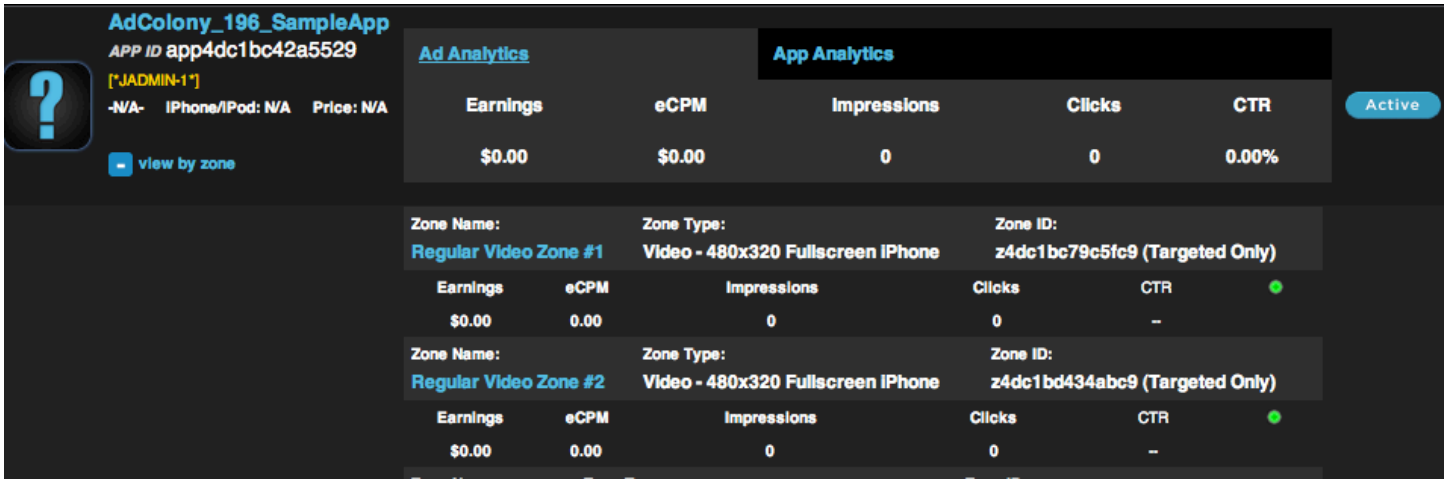
AdColony SDK Integration

— Step 1: Copy plug-in files to Unity project

1. Copy the contents of the Assets/Plugins folder that came with the UnityADC downoad into the Assets/Plugins folder of your Unity project. You should now have the following seven files:

    Assets/Plugins/UnityADC.cs
    Assets/Plugins/iOS/UnityADC.mm
    Assets/Plugins/iOS/AdColonyPublic.h
    Assets/Plugins/iOS/libAdColony.a
    Assets/Plugins/Android/unityadc.jar
    Assets/Plugins/Android/adcolony.jar
    Assets/Plugins/Android/AndroidManifest.xml

— Step 2: Gather Information from Your AdColony Account
Login to adcolony.com. If you have not already done so, create an app and needed zones on the website, following the instructions provided there. Then retrieve your **app ID** and your corresponding **zone IDs** from the AdColony website and make note of them for use in Step 4.



— Step 3: Call AdColony.Configure()

Call *AdColony.Configure()* from the *Start()* method of one of your game objects (usually an overall game manager). The necessary arguments are as follows (by example):

```
AdColony.Configure(
  "1.0",                // Arbitrary app version
  "app4d87a5ca2e592",   // ADC App ID from adcolony.com
  "z4d87a5e1b8967",     // A zone ID from adcolony.com
  "z4daf3029bdd8a"      // Any number of additional Zone IDS
);
```

— Step 4: Add CoreTelephony to your iOS project (iOS only)

Add the CoreTelephony framework to the Xcode project that Unity generates.

Congratulations! You have successfully integrated the AdColony SDK into your application. The following sections explain how to add video advertisements at specific places in your app.

Adding Video Ads

You can call *AdColony.ShowVideoAd()* as desired (often at the beginning of a new game or level) to show an interstitial video. If a video is available it will be shown; otherwise nothing will happen.

```
// Example 1: Play a video ad if available.
AdColony.ShowVideoAd();


// Example 2: Play from a specific zone, if available.
AdColony.ShowVideoAd( "z4d87a5e1b8967" );
```

Your app is now ready to play video ads! Build and run your app on an iOS device. After your app begins running, give AdColony time to prepare your ads after the first launch; 1 minute should be sufficient. Then trigger video ads to be played. You should see an AdColony test ad play. If no video ads play, double check the previous steps. Make sure that you are providing the correct App ID and Zone ID.

The AdColony plug-in will automatically pause and resume your game by setting Time.timeScale to 0 and then restoring it to its prevoius value. If you wish to perform additional actions (such as pausing background music), there two video playback delegates you can hook into. "AdColony.OnVideoStarted" will be called when the video starts to play and "AdColony.OnVideoFinished" will be called after AdColony has finished. For example:

```
// Example 3: Play an ad with start/finish delegate notifications.
AdColony.OnVideoStarted = OnVideoStarted;
AdColony.OnVideoFinished = OnVideoFinished;
AdColony.ShowVideoAd();
...
void OnVideoStarted()
{
  Debug.Log( "An AdColony video is starting to play." );
}

void OnVideoFinished()
{
  Debug.Log( "AdColony finished." );
}
```

Note that the *OnVideoFinished* delegate will be called whether or not a video actually plays.

Adding Videos For Virtual Currency

Videos-For-Virtual-Currency™ (V4VC™) is an extension of AdColony's video ad system. V4VC allows application developers to reward users with an app's virtual currency or virtual good after they have viewed an advertisement. In addition to the standard setup for video ads, one should set up optional communication between AdColony's servers and your servers so that you can maintain the security of your virtual currency system, and properly notify app users about changes to their virtual currency balance within your app.

Configuring a Video Zone for Virtual Currency on adcolony.com

— Step 1
Sign into your adcolony.com account and navigate to the configuration page for your application's video zone.

— Step 2
Tick the checkbox to enable virtual currency for your video zone, and enter values for all of the fields except the URL field. The currency name field is used so that in an app with multiple types of currency, you can configure the type of currency to award from the AdColony control panel, and also to be able to award a different currency for each video zone.

— Step 3
Fill in the URL field with a convenient URL on your server that will house a callback accessed by AdColony when awarding currency. See the *Server-side Changes* section for details about this callback.

Using Videos for Virtual Currency in Your App

— Step 1: Set up an AdColonyV4VCListener to receive results
After a V4VC video plays AdColony will inform your app of the results. Create a method that conforms to the AdColony.V4VCResultDelegate signature and assign the method to AdColony.OnV4VCResult. For example:

```
...
void Start()
{
  AdColony.OnV4VCResult = OnV4VCResult;
}

void OnV4VCResult( bool success, string name, int amount )
{
  if (success) Debug.LogWarning("V4VC SUCCESS: "+amount+" "+name );
  else         Debug.LogWarning("V4VC FAILED!" );
 }
```

If *success* is true than the virtual currency transaction is complete and your server has awarded the currency. This callback should appropriately update your application's internal state to reflect a changed virtual currency balance. For example, contact the server that manages the virtual currency balances for the app and retrieve a current virtual currency balance, then update the user interface to reflect the balance change. Apps may also want to display an alert to the user here to notify them that the virtual currency has been credited.

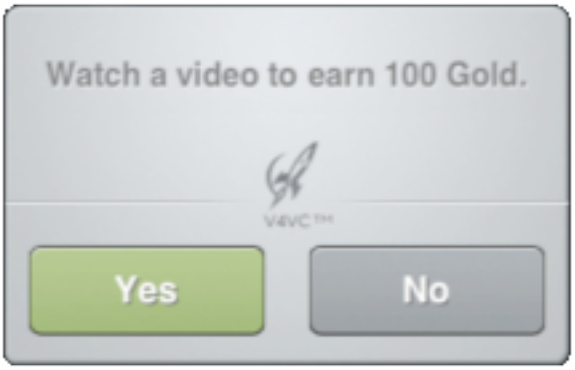
If *success* is false then the video played but for some reason the currency award failed (for example, perhaps the virtual currency server was down). Apps may want to display an alert to user here to notify them that virtual currency rewards are unavailable.

IMPORTANT: In the event of a various network problems, a currency transaction will not be instantaneous, which can result in this callback being executed by AdColony at any point during your application. Delayed results can also be returned from previous runs of the program, so to make sure you don't miss any notifications you should assign your OnV4VCResult delegate **before** calling *AdColony.Configure().*

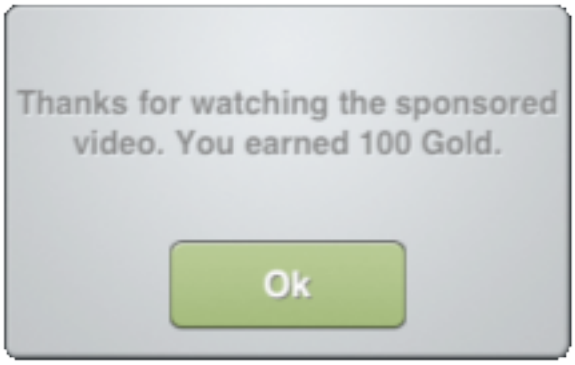
— Step 2: Show a V4VC ad as-is or with pop-ups
Calling *AdColony.ShowV4VC(false);* to show a video for virtual currency (if available) and then call your delegate method with the result.

Alternatively AdColony provides two default popups to provide the user information about V4VC. One popup can be triggered which offers users the choice of watching a V4VC video and is referred to in this document as the pre-popup. The other popup can be triggered after the V4VC video finishes and is referred to in this document as the post-popup. These popups include information you entered on adcolony.com, informing users of the name and amount of currency they will receive. You may choose to use these popups or to ignore them. Many apps implementing V4VC implement their own custom popups to match the app's look.

The pre-popup currently has the following appearance:



The post-popup currently has the following appearance:



To use only the post-popup, send "true" to *ShowV4VC* as follows:

```
AdColony.ShowV4VC( true );
```

To use the pre-popup, call *AdColony.OfferV4VC(bool showPostPopup)* instead. The following call would use both pre and post popups:

```
AdColony.OfferV4VC( true );
```

Server-side Changes to Reward Virtual Currency Users

To provide security for your virtual currency economy, AdColony issues callbacks which use message hashing for security directly to your servers that handle your virtual currency. In order to reward your users with the virtual currency rewarded by AdColony, you should create a callback URL on your game's server system. AdColony will pass URL parameters to your game's server via this URL, which are then used to update a user's virtual currency balance in your system.

In AdColony 1.9.7, we have added an option to enable client-side handling of virtual currency. Please note that use of this option is not advised because there is no way to create a secure client-side virtual currency system. While we do our best to obfuscate our client-side system, it is not possible to ensure its security. If you are unable to use a server to manage your virtual currency system, contact support@adcolony.com for usage guidelines.

— Step 1:
You must create a URL on your servers to receive the AdColony callback. The callback URL must not require any authentication to reach your server. Once you have chosen this URL, you should input it in the video zone configuration page on adcolony.com.

— Step 2:
You must make your URL respond appropriately to the AdColony callback. The format of the URL that AdColony will call is as follows, where brackets indicate strings that will vary based on your application and the details of the transaction:

[http://www.yourserver.com/anypath/callback_url.php]?id=[transaction id]&uid=[user id]&amount=[currency amount to award]&currency=[name of currency to award]&verifier=[security value]

Parameter Name	Type	Purpose
id	Positive long integers	Uniquely identifies transactions
uid	Alphanumeric string	Unique user ID
amount	Positive integer	Amount of currency to award
currency	Alphanumeric string	Name of currency to award
verifier	Alphanumeric string	MD5 hash for transaction security

It is not necessary to use PHP for your callback URL. You can use any server side language that supports an MD5 hash check to respond to URL requests on your server.

For your convenience, the following PHP with MySQL sample code illustrates how to access the URL parameters, perform an MD5 hash check, check for duplicate transactions, and how to respond appropriately from the URL.

```
$MY_SECRET_KEY = "This comes from adcolony.com";

$trans_id = mysql_real_escape_string($_GET['id']);
$dev_id = mysql_real_escape_string($_GET['uid']);
$amt = mysql_real_escape_string($_GET['amount']);
$currency = mysql_real_escape_string($_GET['currency']);
$verifier = mysql_real_escape_string($_GET['verifier']);

//verify hash
$test_string = "" . $trans_id . $dev_id . $amt . $currency . $MY_SECRET_KEY;
$test_result = md5($test_string);
if($test_result != $verifier) {
	echo "vc_decline";
	die;
}

//check for a valid user
$user_id = //get your internal user id from the device id here
if(!$user_id) {
	echo "vc_decline";
	die;
}

//insert the new transaction
$query = "INSERT INTO AdColony_Transactions(id, amount, name, user_id, time) ".
	"VALUES ($trans_id, $amt, '$currency', $user_id, UTC_TIMESTAMP())";
$result = mysql_query($query);
if(!$result) {
	//check for duplicate on insertion
	if(mysql_errno() == 1062) {
		echo "vc_success";
		die;
	}
	//otherwise insert failed and AdColony should retry later
	else {
		echo "mysql error number".mysql_errno();
		die;
	}
}

//award the user the appropriate amount and type of currency here
echo "vc_success";
```

The MySQL database table referenced by the previous PHP sample can be created using the following code:

```
CREATE TABLE `AdColony_Transactions` (
  `id` bigint(20) NOT NULL default '0',
  `amount` int(11) default NULL,
  `name` enum('Currency Name 1') default NULL,
  `user_id` int(11) default NULL,
  `time` timestamp NULL default NULL,
  PRIMARY KEY  (`id`)
) ENGINE=MyISAM DEFAULT CHARSET=latin1;
```

To prevent duplicate transactions, you must make a record of the id of every transaction received, and check each incoming transaction id against that record after verifying the parameters. If a transaction is a duplicate, there is no need to reward the user, and you should return a success condition.

After checking for duplicate transactions, you should reward your user the specified amount of the specified type of currency.

— Step 3:
You must ensure your callback returns the appropriate string to the AdColony server based on the result of the transaction.

Response	**Reasons for use**	**AdColony reaction**
vc_success	Callback received and user credited Duplicate transaction which was already rewarded	AdColony finishes transaction
vc_decline	uid was not valid Security check was not passed	AdColony finishes transaction
everything else	For some reason the server was unable to award the user at this time--this should only be used in the case of some error	AdColony periodically retries to contact your server with this transaction

Note: The only acceptable reasons to not reward a transaction are if the uid was invalid, the security check did not pass, or the transaction was a duplicate which was already rewarded.

Advanced AdColony

AdColony Methods

The following AdColony static methods are available. *AdColony.Configure()* must be called before any of the others. Assign event delegates (listed on the next page) before calling *Configure()*.

Configure(app_version:string, app_id:string, zone_id_1:string, ...)
Configures AdColony with one or more Video Zone IDs.

GetDeviceID() : string
Returns the MAC Address of the device.

GetOpenUDID() : string
Returns a device identifier as reported by the OpenUDID library. Does not access the standard iOS UDID call that has been deprecated.

GetV4VCAmount() : int
Returns the amount that can be obtained from *ShowV4VC()*.

GetV4VCName() : string
Returns the name of the virtual currency as set on the AdColony server.

IsV4VCAvailable() : bool
The player will get a virtual currency reward after the ad.

isVideoAvailable() : bool
Returns "true" if a video will play when you call *ShowVideoAd()*.

OfferV4VC(postPopup:bool)
OfferV4VC(postPopup:bool, zone_id:string)
Shows a built-in popup asking the user if they want to watch a video for virtual currency. If they say "yes" then the video will automatically be shown. If no *zone_id* is given than the default is used.

ShowV4VC(postPopup:bool)
ShowVideoAd(postPopup:bool, zone : string)
Attempts to play a video for virtual currency. If no *zone_id* is given than the default is used.

ShowVideoAd()
ShowVideoAd(zone : string)
Attempts to play a video ad. If no *zone_id* is given than the default is used.

AdColony Event Delegates

OnVideoStarted()

This delegate is called when a video begins playback.

OnVideoFinished()

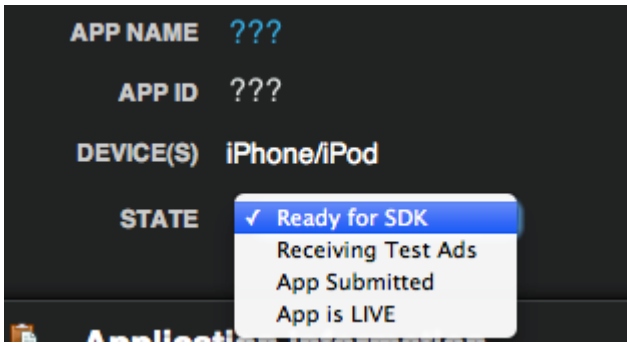
This delegate is called both when a video finishes playback and when no video plays after a call to *ShowVideo()* or *ShowV4VC()*.

OnV4VCResult(bool success, string name, int amount)

This delegate is called when a V4VC video has played and the virtual currency result is available.

Managing Application State

On the Adcolony Publisher Portal, you will notice an application state field when you create or edit an application:



The following is a detailed explanation of each state:

STATE	**DESCRIPTION**
Ready for SDK	The DEFAULT state set on application creation.
Receiving Test Ads	As soon as the developer integrates the SDK, sets up their zone and receives 1 test impression, the system will auto switch the application status to "Receiving Test Ads."
App Submitted	The application will continue to receive test ads until the developer submits his or her app to the App Store and selects "App Submitted" from the drop-down menu.
App is LIVE	When the application goes live, the developer must select "App is LIVE" from the drop-down menu to enable live campaigns.