AdColony Quick Start Guide

Unity Plug-in for Android
AdColony Version 1.9.11
Updated January 18, 2013

Table of Contents

---

1. Introduction

AdColony 1.9.11 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

---

2. Changes to the Library and Updating Applications

Changes in AdColony 1.9.11 Android
1. Removed GET_TASKS permission requirement.
2. Added READ_PHONE_STATE permission requirement.
3. App must now call AdColony.pause() from the Activity's onPause() and AdColony.resume(this) from the Activity's onResume(). **Note:** This is handled by the AdColony Unity Plugin.

Updating from AdColony 1.9.7 through 1.9.9
Edit your ApplicationManifest.xml file to add the READ_PHONE_STATE permission. If you previously had added GET_TASKS just for AdColony you can remove this permission from the manifest file.

AdColony Unity Plugin Changes

January 10, 2013 -- Android 1.9.12
- Fixed crash on task-switching after long press HOME.
- Added calls to get/set "custom id" to allow third-party V4VC servers to associate installs with account info.
- Fixed V4VC callbacks to ALWAYS be called with a true/false value even in the case of 3rd party server error.
- Fixed an bug that in rare cases could prevent offline impressions from being reported.

October 10, 2012
-Updated with AdColony 1.9.11 Android SDK

August 23, 2012
-Fixed an issue with V4VC post-popups.
-Added support for multiple scene projects.
-Removed dependency on 'Main Camera' object.
-Updated with AdColony 1.9.9 Android SDK

April 6, 2012
Added AdColony.getDeviceID() that returns an "Installation ID" – a globally unique string that only changes its value if an app is removed and reinstalled or the device is wiped.

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

3. AdColony SDK Integration

— Step 1:   Copy plug-in files to Unity project
1.   Copy the contents of the Platform-Android/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/AdColony.cs
> Assets/Plugins/Android/unityadc.jar
> Assets/Plugins/Android/adcolony.jar
> Assets/Plugins/Android/AndroidManifest.xml

— Step 2:   Edit AndroidManifest.xml
Replace the package name "com.jirbo.unitytest" with your own app's package name.

Login to clients.adcolony.com. If you have not already done so, create an app and needed zones on the website. To create new apps and video zones, locate the green buttons on the right-hand side of the Publisher section. Then retrieve your **app ID** and your corresponding **zone IDs** from the AdColony website and make note of them for use in Step 4. Please reference the screenshots below on locations of the **app ID** and **zone IDs**.

— 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
);
```

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.

4. 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;
// Delegates must be assigned BEFORE Configure() is called
AdColony.Configure( … );
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.
IMPORTANT: You must assign the delegates **before** calling `AdColonly.Configure().`

5. Adding Videos-For-Virtual-CurrencyTM

Videos-For-Virtual-CurrencyTM (V4VCTM) 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. AdColony V4VC does not keep track of your users' currency balances; it provides notifications to you when a user needs to be credited with a reward.

AdColony's V4VC system can be implemented in two different ways: client-side or server-side. We recommend that all developers use a server-side integration because it offers the most security for your virtual currency system; however, it requires that you operate your own internet-accessible server. In client-side mode, our V4VC system does not require you to operate a server.

Our help center documentation details recommended usage and settings for V4VC. Please reference the following informational and best practices documents online for more details.
V4VCTM Security and Usage Tips
Videos-For-Virtual-CurrencyTM (V4VCTM)

If you are upgrading from AdColony 1.9.9 or earlier versions and use V4VC in server-side mode, be sure to read section "Server-side Setup" for required changes.

Configuring a Video Zone for V4VC on clients.adcolony.com

— **Step 1**

Sign into your clients.adcolony.com account and navigate to the configuration page for your application's video zone. (You may have to create a new video zone.)

— **Step 2**
Select **Yes** under the **Virtual Currency Rewards** section to enable virtual currency for your video zone. Depending on whether you operate a server to track users' virtual currency balances, select either **Yes** or **No** for **Client Side Only?** Please read our the "V4VC Security and Usage Tips" online document linked above for details the benefits of a server-side

setup.
Select the appropriate settings and enter values for all of the fields except the **Callback URL** field. The **Virtual Currency Name** field should reflect the name of the currency rewarded to the user. The **Daily max per user** should reflect the number of times you want a user to be able to receive rewards per day. The **Reward per completed view** should reflect the amount you wish to reward the user.

AdColony 2.0 introduces a new setting that allows you to fine tune V4VC for your game economy. If the virtual currency that you use with V4VC is more valuable than the typical revenue generated from an AdColony video, then you should supply a value for the **Videos Needed per Reward** field that is greater than 1.

— **Step 3**
If you have selected a server-side integration, fill in the **Callback URL** field with a URL on your server that will be contacted by the AdColony SDK to notify it whenever a user is completes a V4VC video for a reward.

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
Call *AdColony.ShowV4VC(false);* to show a video for virtual currency (if available) and AdColony will then call your delegate method with the result.

Alternatively AdColony provides two default popups to provide the user information about V4VC. These popups include information you entered on clients.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.
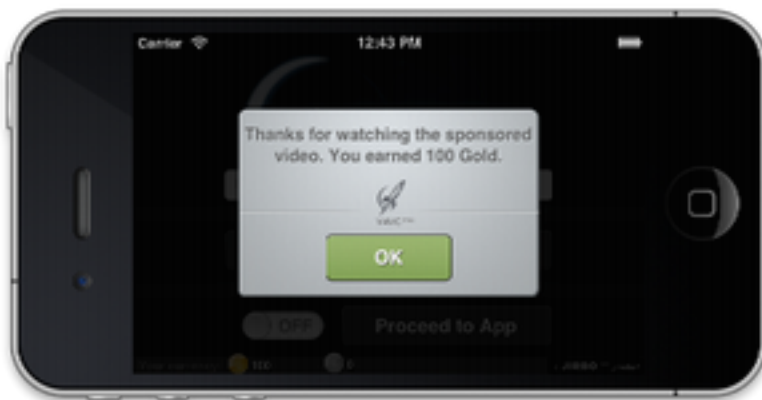
One popup can be triggered which allows users to begin a V4VC video and is referred to in this document as the pre-popup. The other popup can be triggered after the V4VC video finishes and is referred to in this document as the post-popup.

The pre-popup currently has the following
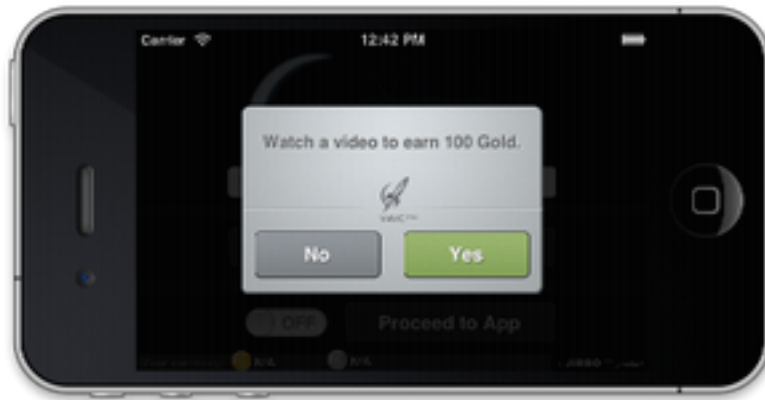
appearance:

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

In AdColony 1.9.7, we 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.

The following steps are only necessary if you are implementing a server-side V4VC setup. If you are upgrading from AdColony 1.9.9 or previous versions, be sure to update your V4VC callback code to account for the new URL parameters, as described in Step 2.

To provide security for your virtual currency economy, AdColony relies upon your game server to mediate virtual currency rewards for users. Without a server-backed system, it is impossible to create a totally secure virtual currency reward system. AdColony issues web calls directly to your servers that handle your virtual currency. These web calls use message hashing for security so that users cannot be rewarded with currency they did not earn.

— Step 1: Create a URL
In order to reward your users with the virtual currency they have earned via AdColony, you must create a callback URL on your game's server system. AdColony will pass data to your game's server via this URL, which are then used to update a user's virtual currency balance in your system.

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, such as HTTPS. AdColony will pass data to your URL using the HTTP verb "GET". You will want to create this URL in a directory that can execute server-side code such as PHP. This URL should match your input in the video zone configuration page on clients.adcolony.com for your virtual currency zone. See Step 3 of the section titled "Configuring a Video Zone for V4VC on clients.adcolony.com".

— Step 2: Add Security and Reward Logic
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=[AdColony device id]&amount=[currency amount to award]&currency=[name of currency to award] &open_udid=[OpenUDID]&udid=[UDID]&odin1=[ODIN1]&mac_sha1=[SHA-1 of MAC address] &verifier=[security value]

URL Parameter Name	Type	Purpose
id	Positive long integers	Unique transaction ID
uid	Alphanumeric string	AdColony device ID (not Apple UDID)
amount	Positive integer	Amount of currency to award
currency	Alphanumeric string	Name of currency to award
open_udid	Alphanumeric string	OpenUDID (not Apple UDID)
udid	Alphanumeric string	Apple UDID
odin1	Alphanumeric string	Open Device Identification Number

mac_sha1	Alphanumeric string	Same as uid (AdColony device ID)
verifier	Alphanumeric string	MD5 hash for transaction security

You need some type of server-side language to process and act upon AdColony's calls to your callback URL. 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. 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; you will simply need to adapt the following code sample to your language of choice. Please note that you must concatenate the URL parameters in the order shown or the hash check will not pass.

```php
<?php

    $MY_SECRET_KEY = "This is provided by adcolony.com and differs for each zone";

    $trans_id = mysql_real_escape_string($_GET['id']);
    $dev_id = mysql_real_escape_string($_GET['uid']);
    $amt = mysql_real_escape_string($_GET['amount']);
    $currency = mysql_real_escape_string($_GET['currency']);
    $open_udid = mysql_real_escape_string($_GET['open_udid']);
    $udid = mysql_real_escape_string($_GET['udid']);
    $odin1 = mysql_real_escape_string($_GET['odin1']);
    $mac_sha1 = mysql_real_escape_string($_GET['mac_sha1']);
    $verifier = mysql_real_escape_string($_GET['verifier']);

    //verify hash
    $test_string = "" . $trans_id . $dev_id . $amt . $currency . $MY_SECRET_KEY .
$open_udid . $udid . $odin1 . $mac_sha1;
    $test_result = md5($test_string);
    if($test_result != $verifier) {
          echo "vc_noreward";
          die;
    }

    $user_id = //get your internal user id using one of the supplied device identifiers

    // the device identifiers (OpenUDID, AdColony ID, ODIN1) can be accessed via a
method call in the AdColony client SDK

//check for a valid user
    if(!$user_id) {
          echo "vc_noreward";
          die;
```

```php
    }

    //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";

    ?>
```

Please note that this code sample is incomplete; it requires application-specific code to be inserted by you at appropriate points to function correctly with your app server. Be sure to use your secret key for your application from clients.adcolony.com during the verification process.

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

```sql
    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=utf8;
```

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 SDK based on the result of the transaction.

Response	Reasons for use	AdColony reaction
vc_success	Callback received and user credited Transaction ID was already rewarded	AdColony finishes transaction
vc_noreward	Unknown user Security check did not pass	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 user cannot be identified, the security check did not pass, or the transaction was a duplicate which was already rewarded.

---

6. 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.

GetODIN1() : string
     Not Supported in Android. This will always return the string 'undefined'.

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 Status

In the AdColony Publisher tab of the AdColony Portal, you will notice the application status when you create new applications:

The following is a detailed explanation of each status:

STATUS	DESCRIPTION
Testing	Indicates that one or more video zones are showing "Test Ads". To remove the "Testing" status from a zone, click on the video zone and select "**No**" in the Development (show test ads only) section.
Active	As soon as the developer integrates the SDK, sets up their zone and select "**No**" in the Development section, the system will auto switch the application status to "Active".

Inactive	Indicates that no video zones are created or all video zones have been inactivated. To inactivate video zones, go inside the video zone and select "**No**" to Integration (zone is active) section.

Note: Even if your application status is "Testing", you can still receive live video ads to your application provided that you have at least one video zone in "Active" status. We recommend that you deactivate video zones you are not using so that the proper application status is displayed.

Test Ads, Live Ads, and Switching

AdColony provides test ads that perform identically to live ads, with few exceptions. Test ads will not affect your account balance. In V4VC zones, test ads will not obey the 'Daily Max Per User' setting. We have a policy to avoid directing live ads to applications that are still in development or testing.

New video zones automatically default to receive test ads. Please note that if you set your zone to receive live ads before your application is live on a user-facing marketplace with AdColony included, it may not receive any ads.

You can toggle test ads in the Publisher section of the clients.adcolony.com control panel.

1. Ensure your App's 'SDK integration' indicates communication with the AdColony server (please note this status updates roughly hourly).
2. Publishers->Click on your app's link-> **Edit.**
3. Click on the link for the **Video Zone** in which you'd like to change the type of video ads (Test or Live Video Ads).
4. Select the corresponding radio button in the **Show test ads only (for dev or debug)?** section and click **Save**. Do this for each video zone in your app

---

7. Integration With 3rd Party Networks and Aggregators

AdColony can be used with multiple external ad *networks* and *aggregators*. In most cases, you may simply integrate the external network or aggregator using its included instructions, then integrate AdColony using these instructions. As of October 26th, 2010, AdColony video ads have been tested

and work side-by-side with the following SDKs:

- AdMob [version dated 2010/09/08] (http://www.admob.com)

- AdWhirl [version 2.6.1] (http://www.adwhirl.com)
- Google AdSense [version 3.1] (https://www.google.com/adsense)
- Medialets [version 2.3.2] (http://www.medialets.com)
- Millennial Media [version 4.0.5] (http://www.millenialmedia.com)
- Mobclix [version 4.1.6] (http://www.mobclix.com)

---

8. Troubleshooting, F.A.Q., and Sample Applications

Please have a look at our sample application. It include helpful comments and are designed to show typical usage scenarios of AdColony in applications. Seeing AdColony in the context of a full application might address issues with API usage.

NOTE: The sample applications already has the AdColony plugin setup. Refer to this if you are unsure of how the AdColony plugin fits into your existing project.

If you are unable to find an answer to your question or this troubleshooting section does not solve your problem, please contact our support team by sending an email to support@adcolony.com

Issue	Resolution
No Video Ads	For testing purposes, set your zone to receive test ads on clients.adcolony.com. Live zones may not receive ads every time.
No Video Ads	You may be requesting that ads play before video ads are ready. You can check if this is occurring by consulting our section titled Checking Video Readiness.
Poor performance during Video Ad Plays.	Ensure that you are pausing any activity going on in Unity. Any sound, animation, or game logic going on at the same time as a Video Ad Play could cause performance issues. In some extreme cases you may need to unload some extra assets that are loaded in Unity.
Application Audio Stops Working After Video Ad Plays	Ensure that you disable your app's sound or music before a video plays and re-enable it afterwards. See the section titled "Advanced AdColony: AdColony Event Delegates". AdColony

	does its best to avoid interference with your application's audio session, but playing audio during a video ad can cause problems.