AdColony Quick Start Guide

Android SDK

AdColony Version 1.9.13
Updated April 19, 2013

Table of Contents

Introduction

AdColony features high-definition (HD), Instant-PlayTM 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 application/s 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

1.9.13 – April 19, 2013

1. The second parameter of AdColony.configure() has new behavior. Instead of an app version string such as “1.0” you should now send the following string: “version:1.0,store:google”, where *version* is an arbitrary app version number and *store* should be either “google” or “amazon” depending on where you will publish your app. AdColony now supports apps published on Amazon Appstore and will not serve ads to Amazon apps that link to the Google Play store (which would otherwise lead to Amazon rejecting the app).
2. The additional parameter *skippable* may be passed along with *version* and *store* as described above (example: “version=1.1,store:google,skippable”). Specifying *skippable* allows users to cancel a video in progress and return control to the app. **No publisher earnings or V4VC rewards will occur if an ad is canceled using this method.** This should only be used by apps that must respond to an incoming event like a VoIP phone call.
3. Low-level refinements open up a wider selection of ad inventory - for example, ads that are intended for tablets only.
4. AdColony now respects interval and V4VC Daily Max per User settings in control panel.

1.9.12 – January 10, 2013

1. Fixed crash on task-switching after long press HOME.
2. Changed the AdColonyV4VCListener’s onV4VCResult() to always be called even if a custom currency server is not replying with a properly formatted V4VC result.
3. Added AdColony.setCustomID(String) and AdColony.getCustomID():String. The custom ID will pass through to server-side V4VC callbacks to use at your discretion.
4. Fixed an bug that in rare cases could prevent offline impressions from being reported.

1.9.11 – October 9, 2012

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().

1.9.10 – September 21, 2012

Fixed video to no longer resume playback behind lock screen.

1.9.9 – April 24, 2012

Various bug fixes.

1.9.7b – September 2, 2011

This is the first release of AdColony for Android.

AdColony SDK Integration

— Step 1: Drop adcolony.jar into your project

Eclipse

1. Place **adcolony.jar** in your project's root directory.
2. Right click your project name and select **Properties.**
3. Select **Java Build Path®Add External JARs...**
4. Select **adcolony.jar** in the dialog. It will now appear in your project's list of Referenced Libraries.

Command Line

1. Place **adcolony.jar** in your project’s **libs** folder.

— Step 2: Edit AndroidManifest.xml

1. Ensure the following four permissions are set:

INTERNET
READ_PHONE_STATE
WRITE_EXTERNAL_STORAGE

ACCESS_NETWORK_STATE

You can do this by pasting the following lines before the <application…> tag.

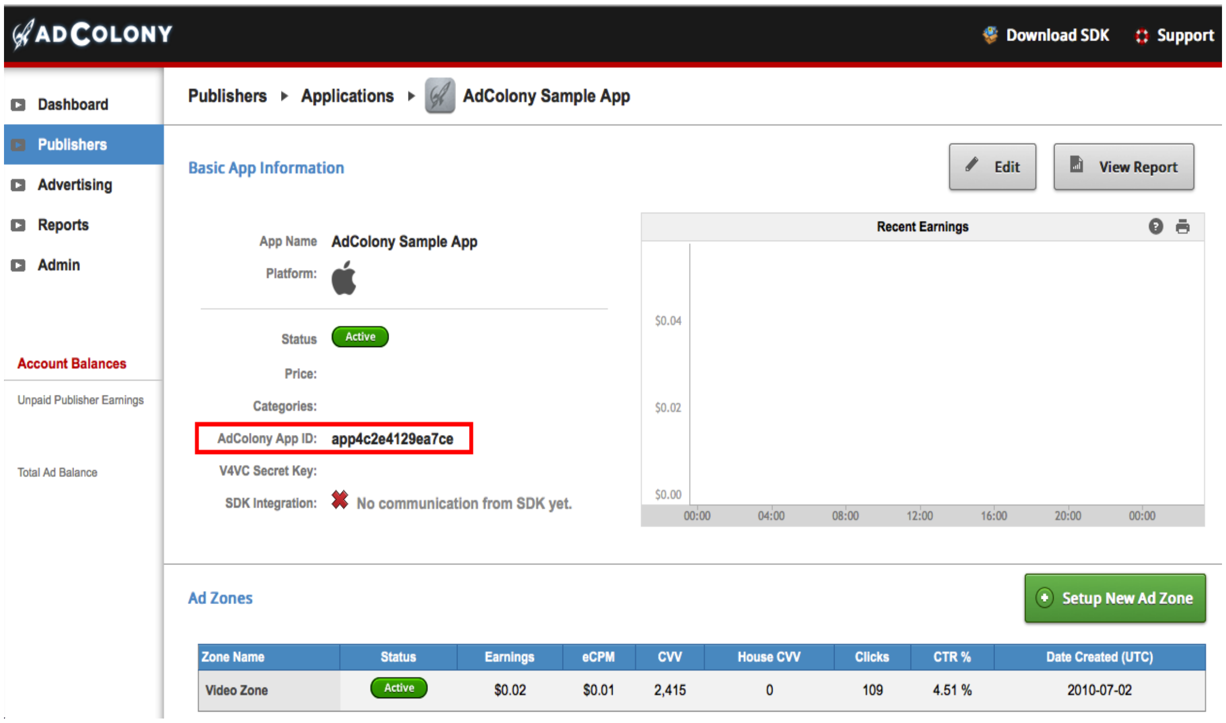
```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission
 android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission
 android:name="android.permission.ACCESS_NETWORK_STATE" />
```

2. Next, copy and paste the following three activity definitions just before the “</application>” tag near
the bottom:

```
<activity android:name="com.jirbo.adcolony.AdColonyOverlay"
   android:configChanges="keyboardHidden|orientation"
          android:theme=
   "@android:style/Theme.Translucent.NoTitleBar.Fullscreen" />
<activity android:name="com.jirbo.adcolony.AdColonyFullscreen"
   android:configChanges="keyboardHidden|orientation"
   android:theme="@android:style/Theme.Black.NoTitleBar.Fullscreen" />
<activity android:name="com.jirbo.adcolony.AdColonyBrowser"
   android:configChanges="keyboardHidden|orientation"
   android:theme="@android:style/Theme.Black.NoTitleBar.Fullscreen" />
```

— Step 3: 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.



App Name	AdColony Sample App
Platform:	
Status	Active
Price:	
Categories:	
AdColony App ID:	app4c2e4129ea7ce
V4VC Secret Key:	
SDK Integration:	✗ No communication from SDK yet.

Zone Name	Status	Earnings	eCPM	CVV	House CVV	Clicks	CTR %	Date Created (UTC)
Video Zone	Active	$0.02	$0.01	2,415	0	109	4.51 %	2010-07-02

— Step 4: Import com.jirbo.adcolony.* and call AdColony.configure()

Add the following *import* to your activity's **.java** file:

```
import com.jirbo.adcolony.*;
```

Add the following call in your activity's *onCreate()* method:

```
AdColony.configure(
  this,                 // Activity reference
  "1.0",                // Arbitrary app version
  "app4d87a5ca2e592",  // ADC App ID from adcolony.com
  "z4d87a5e1b8967",         // Video Zone ID #1 from adcolony.com
        "z4daf3029bdd8a"     // Zone #2 - you can have any number
);
```

— Step 5: Call AdColony.pause() and AdColony.resume() from onPause() and onResume()

Override your Activity's *onPause()* and *onResume()* methods to call corresponding AdColony methods as follows:

```
  public void onPause()
  {
    super.onPause();
    AdColony.pause();
  }

  public void onResume()
  {
    super.onResume();
    AdColony.resume( this );
  }
```

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 create AdColonyVideoAd objects on the fly and tell them to show a video. If a video is available it will be shown. You can pass an optional listener that will be called afterwards in either case (either the video has finished or no video was available):

```
// Example 1: play from any available video zone with no completion
// listener.
new AdColonyVideoAd().show( null );


// Example 2: play from a specific zone and print out a message once
// the video has finished.
new AdColonyVideoAd("z4d87a5e1b8967").show(
          new AdColonyVideoListener()
          {
          public void onAdColonyVideoFinished()
          {
          Toast.makeText( this, "Video Finished",
          Toast.LENGTH_SHORT ).show();
          }
          }
  );
```

Your app is now ready to play video ads! Build and run your app on an Android 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 zone ID or slot number.

Adding Videos For Virtual Currency

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. 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 class that implements the AdColonyV4VCListener interface and add the listener to AdColony immediately after calling *AdColony.configure()*.

```
  AdColony.addV4VCListener( listener );
  // 'listener' is an object implementing interface AdColonyV4VCListener
```

The listener object should define the following method:

```
  public void onV4VCResult( boolean success, String name, int amount )
```

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.

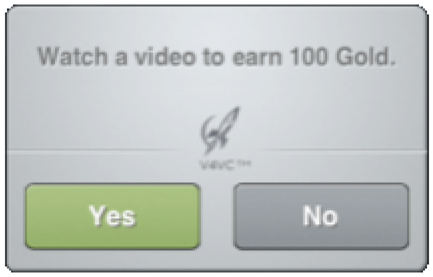
— Step 2:   Create an AdColonyVideoAd object
Create an AdColonyVideoAd object (as in Step 1 of the previous section, *Adding Video Ads*). If you have both V4VC video zones and non-V4VC video zones you must specify a V4VC Zone ID when creating the object.

— Step 3:   Show a V4VC ad as-is or with pop-ups
Calling *showV4VC(AdColonyVideoListener)* will show a video for virtual currency and then call your listener's *onV4VCResult()* method.

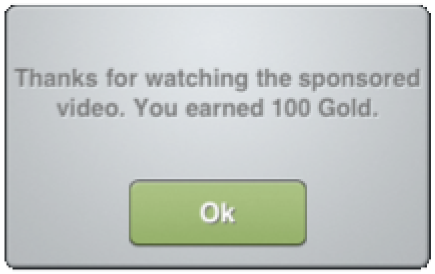
Alternatively AdColony provides two default popups to provide the user information about V4VC.

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:


Thanks for watching the sponsored video. You earned 100 Gold.

To use only the post-popup, add a second parameter to *showV4VC* call as follows:

```
video_ad.showV4VC( null, true );
```

To use the pre-popup, call *offerV4VC(AdColonyVideoListener listener, boolean showPostPopup)* instead. The following call would use both pre and post popups:

```
video_ad.offerV4VC( listener, true );
```

Note that the listener's *onAdColonyVideoFinished()* method is not called if the user declines to watch the video in the pre-popup.

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 we have 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

If your application provides a custom ID to AdColony, you will need to add “&custom_id=[CUSTOM_ID]” to your zone’s callback URL or it will not be provided to your server.

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 except *setDeviceID()*.

configure(Activity, app_version:String, adc_app_id:String, video_zone_1:String, ...)
Configures AdColony with one or more Video Zone IDs.

addV4VCListener(AdColonyV4VCListener)
Registers a listener to be notified about V4VC results.

enable(boolean)
Call to enable or disable all of AdColony. Note that AdColony is automatically disabled on first-generation Android devices but you can call *AdColony.enable(true)* **after** calling

AdColony.configure() to ensure that ads are enabled on those devices as well.

getCustomID() : String
Returns the String previously passed into *setCustomID()* or “” (empty string) if no custom ID is set. See *setCustomID()* for more information.

getDeviceID() : String
Returns a globally unique identifier tied to the current installation of an app. While an appropriate ID is generated automatically when the app is installed, you can also use *setDeviceID()* to set an ID of your choosing. In either case, make sure that the device ID you use to contact the virtual currency server with is the same device ID that AdColony is using.

isTablet() : boolean
Returns "true" when run on Android tablets.

pause()
Call this method from your Activity’s onPause() method.

resume(activity:Activity) : boolean
Call this method from your Activity’s onResume() method. You must pass in your Activity reference (“AdColony.resume(this);”). Returns “true” if AdColony is resuming a paused video ad or modal prompt – this information is used by the AdColony Unity plug-in.

removeV4VCListener(AdColonyV4VCListener)
AdColony stops sending V4VC results to the specified listener.

setCustomID(String id)
Sets a custom ID string that is passed through to server-side V4VC callbacks (as “&custom_id=…”) and can be used at your discretion. See also: *getCustomID()*.

setDeviceID(String id)
Specifies the string identifier to use throughout the app instead of the automatically generated ID. Calls to *getDeviceID()* will return this new ID as well. Must be called **before** *AdColony.configure().* Note: setting your own device ID is completely optional.

AdColonyVideoAd Methods
The following methods can be called on AdColonyVideoAd objects.

AdColonyVideoAd()
Creates a video ad that will play from the first available zone. You should create new

AdColonyVideoAd objects every time you wish to play a video to avoid using outdated data internally.

AdColonyVideoAd(zone : String)
Creates a video ad object to play a video from the specified zone.

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.

getV4VCPlays() : int
Returns the number of times that a V4VC video has been shown today.

getV4VCPlayCap() : int
Returns the maximum number of times that a V4VC may be shown today. This is automatically enforced; you don't need to act on this information.

getV4VCRewardAvailable() : boolean
The player will get a virtual currency reward after the ad.

isReady() : boolean
Returns "true" if a video will play when you call *show()*.

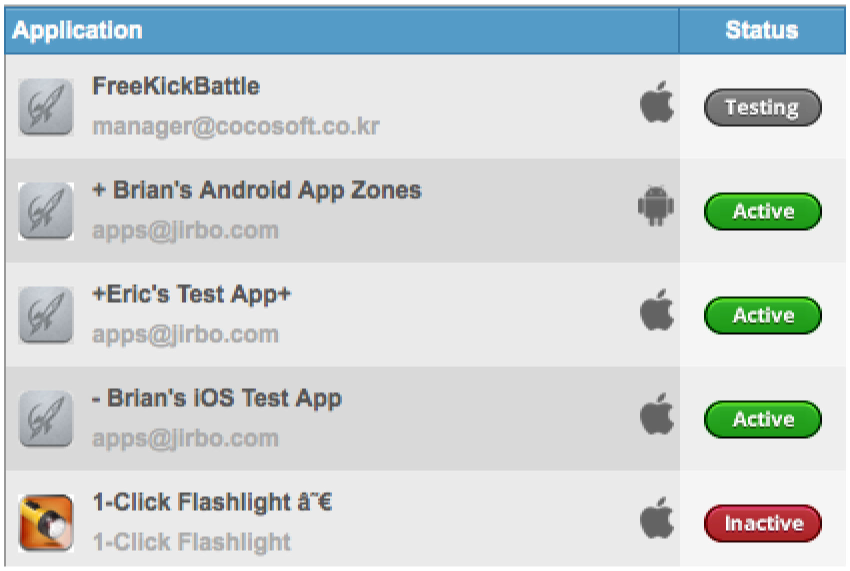
offerV4VC(AdColonyVideoListener, [postPopup:boolean])
Shows a built-in popup asking the user if they want to watch a video for virtual currency. If they say "yes" then the video will automatically be shown. *postPopup* is an optional parameter (default: false).

show(AdColonyVideoListener) : boolean
Attempts to play a video and returns "true" if a video was started successfully. If the listener is non-null, its *onAdColonyVideoFinished()* method will be called regardless.

showV4VC(AdColonyVideoListener, [postPopup:boolean]) : boolean
Attempts to play a video for a virtual currency reward. If no reward is available the video will not play. "true" is returned if a video is started. If non-null, the listener's onAdColonyVideoComplete() method will be called regardless. *postPopup* is an optional parameter (default: false).

Managing Application State

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



Application	Status
FreeKickBattle manager@cocosoft.co.kr	Testing
+ Brian's Android App Zones apps@jirbo.com	Active
+Eric's Test App+ apps@jirbo.com	Active
- Brian's iOS Test App apps@jirbo.com	Active
1-Click Flashlight â˜€ 1-Click Flashlight	Inactive

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.

Sample Applications and Android Caveats

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

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

AdColony Android Caveats

Here are some common pitfalls and obstacles that may be encountered when using AdColony for Android.

- **Limited memory on early Android devices**
  The 16 MB per-program limit on early Android devices means there's often very little room to spare. AdColony videos are automatically disabled on devices with a 16 MB memory class. If you have a simple app and believe the videos would work anyways, call "AdColony.enable(true)" after calling *AdColony.configure()* to renable videos.

- **Limited storage capacity of many devices**
  Many older Android devices have severely limited internal memory (~70MB total). Most of those users have SD Cards for additional storage, but those who do not have very little free space. Because AdColony typically needs a few MB of working space it will not show videos on devices that have less than 20 MB of available storage.

- **Sound system conflicts (no videos play)**
  Even some newer devices (for example, the Samsung Galaxy S) can appear to be working fine and yet they will not play AdColony videos correctly at first. The culprit turns out to be certain types of media files - for example, videos may not play on some phones if your game contains a mix of WAV and M4A sounds. The best solution that we've found is to encode all of your sound effects in the OGG format - those sound files seem to coexist more peacefully with other media types.

- **Device ID**
  The default device ID generated by AdColony (and available via *AdColony.getDeviceID()*) is technically an *installation id* that follows Google's best practices recommendations as outlined here:
  http://android-developers.blogspot.com/2011/03/identifying-app-installations.html

- **Zoom To Fill Screen on tablets**
  Android "Honeycomb" OS 3.2 adds a "Zoom to fill screen" option to the user interface that, when used with AdColony videos on tablets, results in an undesirable level of magnification and cropping. To disallow this behavior either have your app target API Level 11 or higher *or* add this line to your AndroidManifest.xml (along with target=android-10 in default.properties):

  ```
  <supports-screens android:xlargeScreens="true" />
  ```