Quick Start Guide for Apple iOS

AdColony Version 1.9.11
Updated June 26, 2012

Table of Contents

1. Introduction

AdColony 1.9.11 delivers high-definition (HD), Instant-Play™ video advertisements that can be played anywhere within your application. Video ads may require a brief waiting time before the first attempt to play; afterwards, 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 contains step-by-step instructions to easily integrate AdColony into your 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 Email: support@adcolony.com

---

2. Changes to the Library and Updating Applications

You may skip this section if you are adding AdColony to an application for the first time.

Overall Changes
The largest change in AdColony 1.9.11 from 1.9.9 is the addition of new methods of uniquely identifying devices. Apple UDID and ODIN identifiers are now in use, in addition to previous identifiers. This change has a potential impact on any applications using our server-side V4VC system.

Updating from AdColony 1.9.9
Copy the new versions of libAdColony.a and AdColonyPublic.h packaged with this Quick Start Guide into your Xcode project, overwriting the old files.

If your application uses our server-side V4VC system, please review section 5 of this document titled Adding Videos-For-Virtual-Currency.

Updating from AdColony 1.9.8 and 1.9.7
Follow the steps for updating from AdColony 1.9.9.

Changes in AdColony class since 1.9.9
— Change 1
A new method, `getODIN1:` was added. This method allows the developer to access the device's ODIN1 identifier. You are free to include and link your own copy of ODIN1, which should return the same identifier. For more details about ODIN1, please see its Google Code page, available at http://

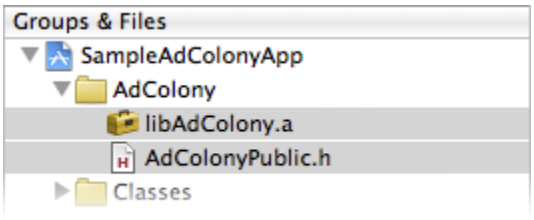
code.google.com/p/odinmobile/. For server-side V4VC integrations, the ODIN1 returned by this method is now included in V4VC callback URLs as the URL parameter named ‘odin1’ and is part of the hashed message body.

---

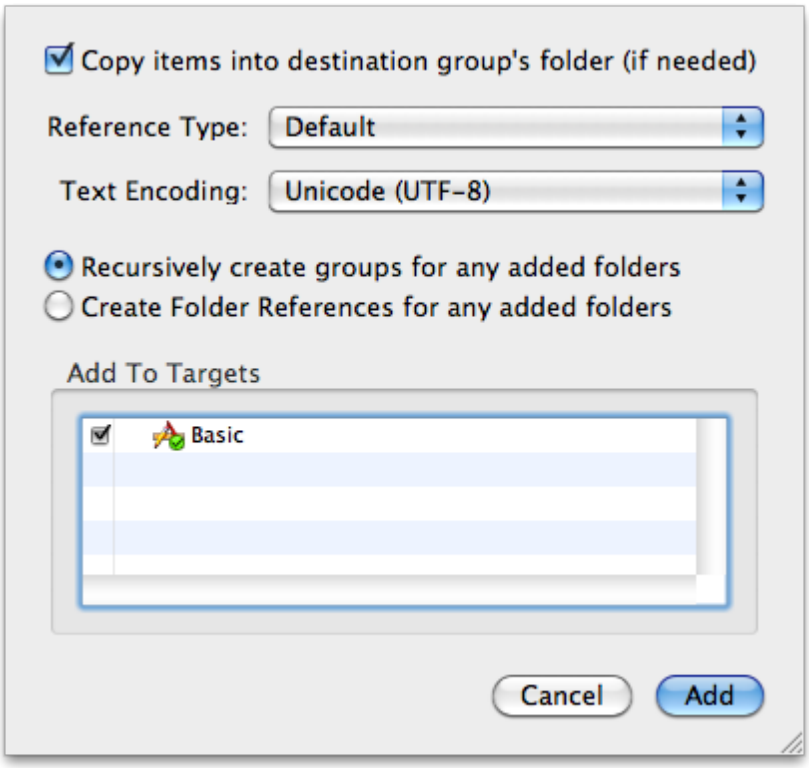
3. AdColony SDK Integration

— Step 1: Choose a Project
Use Xcode to open the existing project into which you want to integrate AdColony, or to create a new iOS project. AdColony requires you to select a Base iOS SDK of version 4.0 or greater.

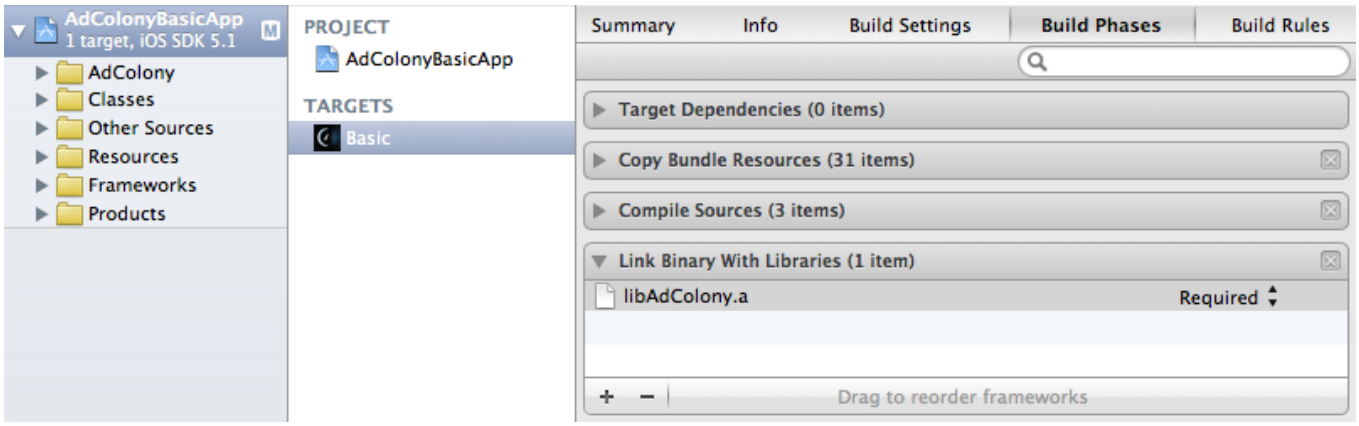
— Step 2: Add Library Files
Download the SDK and copy the `AdColonyPublic.h` and `libAdColony.a` files into the Xcode project.



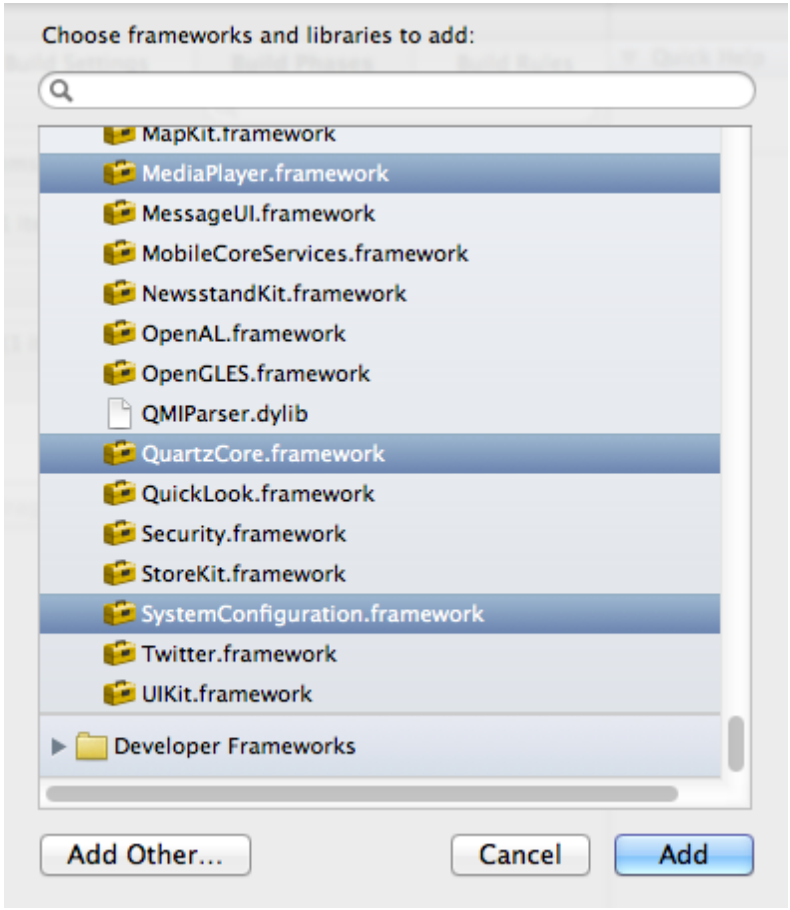
Ensure that they are copied into the project folder and added to all Targets which will utilize AdColony.



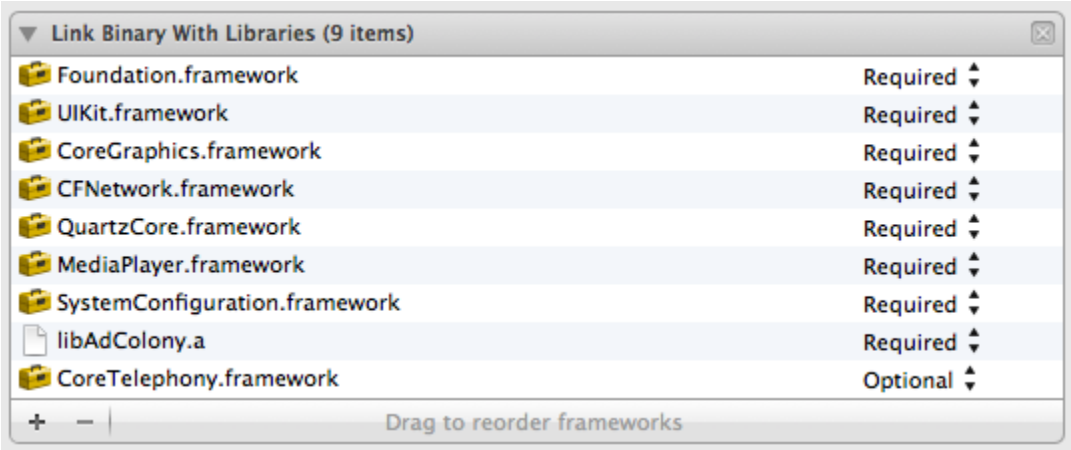
— Step 3: Add Required Frameworks
Inside **Xcode**, select your **Target**, select its **Build Phases** tab, then under the **Link Binary With Libraries** section click the plus sign to add AdColony's required frameworks.



Then select all of the following frameworks from the list and click **Add**: `MediaPlayer.framework`, `CFNetwork.framework`, `CoreGraphics.framework`, `CoreTelephony.framework`, `SystemConfiguration.framework`, and `QuartzCore.framework`.



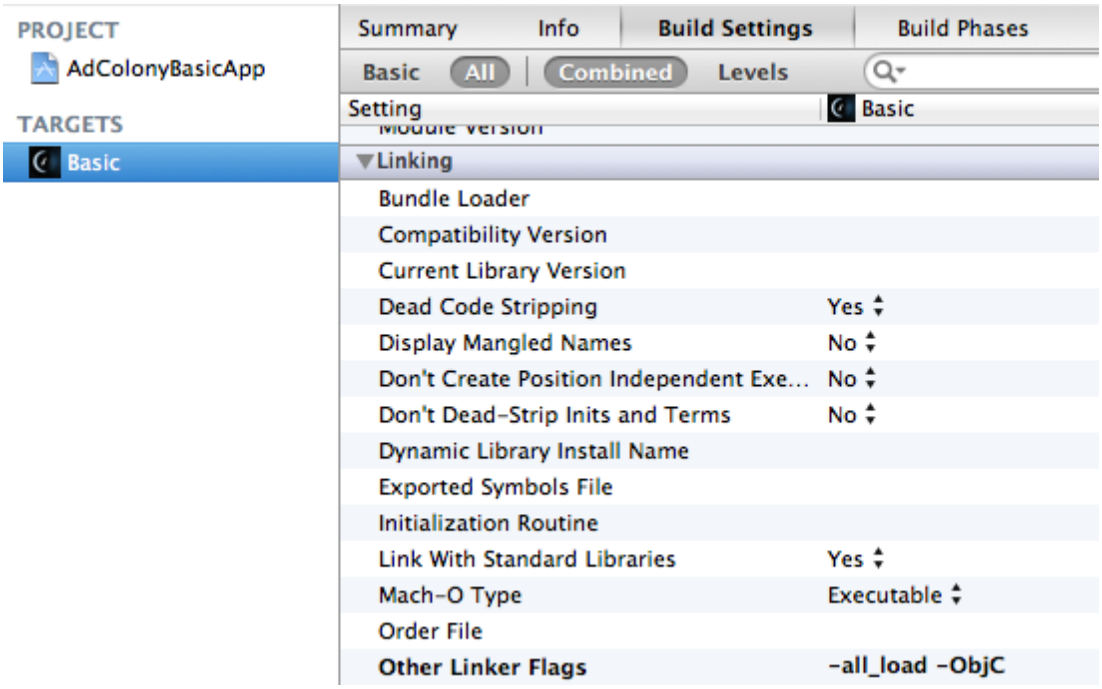
After adding the required frameworks, weak-link the `CoreTelephony` framework by selecting your application's target, then find `CoreTelephony.framework` in the list of included resources and set its **Role** to "**Optional**".



— Step 4: Set Linker Flags

IMPORTANT: Select on your **Target**, select its **Build Settings** tab. In the **Linking** section find the **Other Linker Flags** entry, then and add the **"-all_load"** and **"-ObjC"** flags. Omitting these flags will cause runtime exceptions.

Some apps that use static libraries fail to compile when the **-all_load** flag is set. If this happens to your app, instead of the **-all_load** flag, use **-force_load PATH/TO/LIBRARY/libAdColony.a** instead.



— Checkpoint 1

At this point, build and run your application to ensure that AdColony correctly compiles, links, and executes with your program. If you encounter any problems, double check the previous steps.

— Step 5: Initialize AdColony

In order to show video ads at any point in your application, AdColony must be initialized at every entry point to your application and requires a delegate to handle general callbacks. Typically, the **`UIApplicationDelegate`** is the best choice to be the **`AdColonyDelegate`**; however, if required, the **`AdColonyDelegate`** can be an instance of any Objective-C class which will persist in memory for the lifetime of the application.

To use the **`UIApplicationDelegate`**, which is a recommended practice, open your **`AppDelegate.h`** file, import **`AdColonyPublic.h`** and add the **`AdColonyDelegate`** protocol to your `AppDelegate` class interface.

```
#import "AdColonyPublic.h"

@interface AppDelegate : NSObject <UIApplicationDelegate, AdColonyDelegate> {
```

Then, within your **`AppDelegate.m`** file, call the **`AdColony`**'s static method **`initAdColonyWithDelegate`**: in your application's entry points, which in many cases is the **`application:didFinishLaunchingWithOptions`**: method of your AppDelegate. Your application may use this or other entry points, so if this is the case, be sure to initialize AdColony at every entry point. For the method's parameter, choose the object that will receive general AdColony callbacks, and pass it as the delegate.

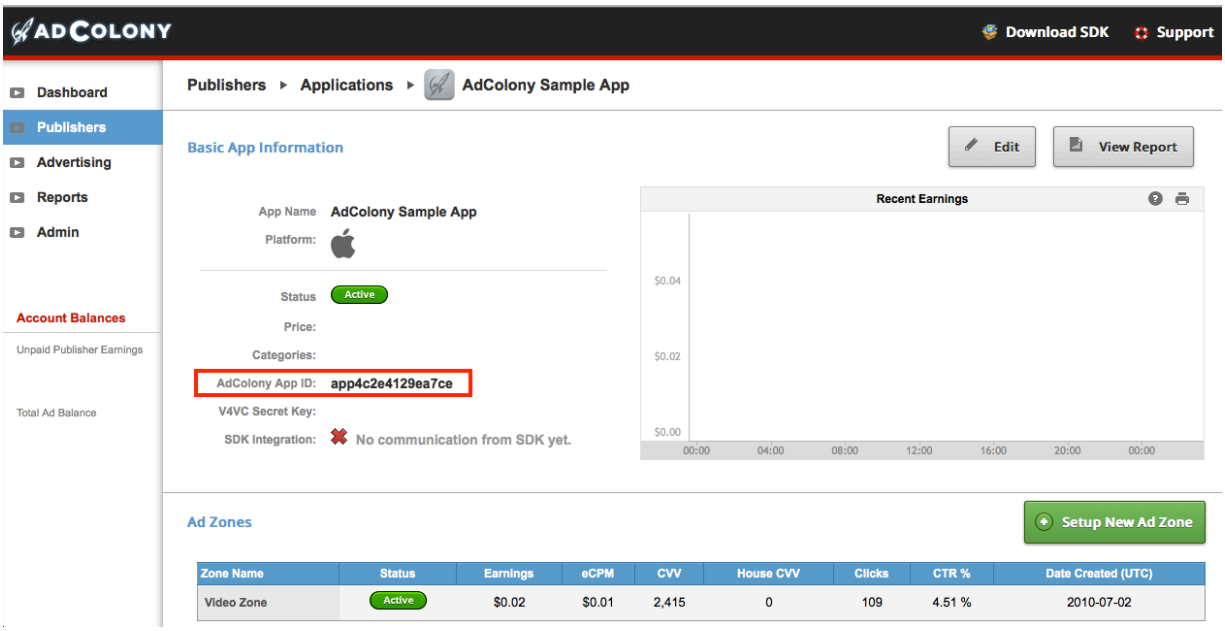
```
- (void)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
   [AdColony initAdColonyWithDelegate:self];
```

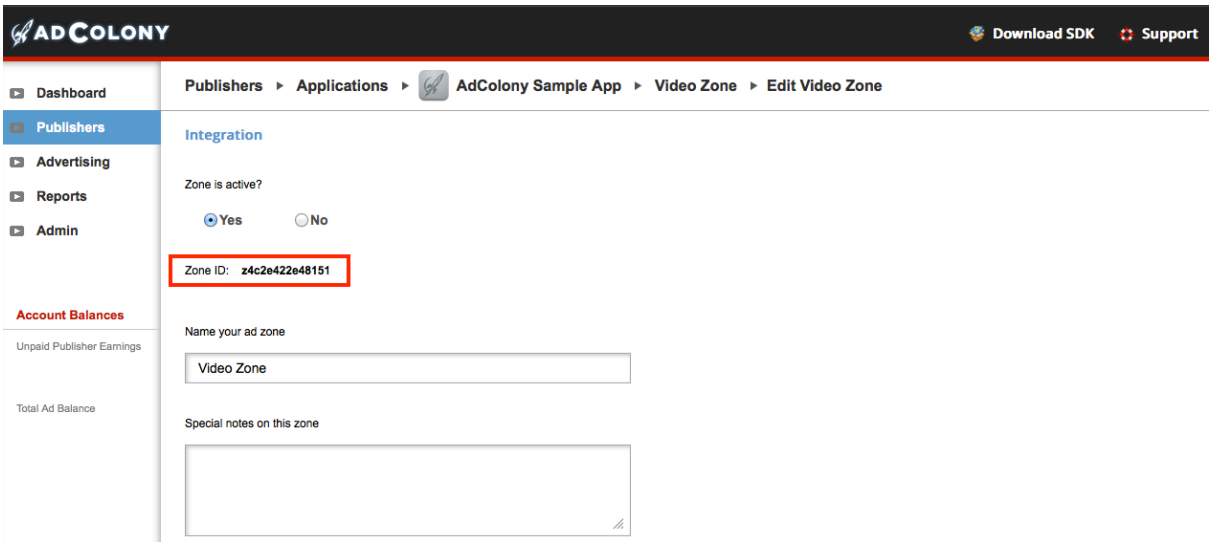
IMPORTANT

In any one execution of your application, AdColony should only be initialized once. For example, this means you should not re-initialize AdColony when your app returns from multitasking.

— Step 6: 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. 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 7. Please reference the screenshots below on locations of the **app ID** and **zone IDs**.





— Step 7: Associate Your App With Your AdColony Info

Implement the `AdColonyDelegate` protocol methods within your chosen delegate class' (usually the `AppDelegate.m`) implementation file. Return your AdColony **app ID** as an NSString* from the `adColonyApplicationID` callback, and return a dictionary of your **zone IDs** from the `adColonyAdZoneNumberAssociation` delegate callback. Each **zone ID** should map to a unique integer-valued NSNumber which you will use internally to refer to zones. These integers are referred to as **slot numbers**, and are a shorthand way to refer to zones.

```
//use the app id provided by adcolony.com
-(NSString*)adColonyApplicationID{
    return @"app4dc1bc42a5529";
}
```

```
//use the zone numbers provided by adcolony.com
-(NSDictionary*)adColonyAdZoneNumberAssociation{
    return [NSDictionary dictionaryWithObjectsAndKeys:
                @"z4dc1bc79c5fc9", [NSNumber numberWithInt:1], //video zone 1
                @"z4dc1bd434abc9", [NSNumber numberWithInt:2], //video zone 2
                nil];
}
```

AdColony uses these two callbacks during its initialization process to properly associate your application with your adcolony.com account.

— Checkpoint

At this point, build and run your application to ensure that AdColony correctly compiles, links, and executes with your program. AdColony should log the version string “AdColony library version: 1.9.9” to the console, indicating it has been initialized. If you encounter any problems, double check the previous steps.

The following sections explain how to add video advertisements at specific places within your app.

---

4. Adding Video Ads

AdColony video ads can be displayed programmatically at any point after you call AdColony's **`initAdColonyWithDelegate`**: method and the video ads have finished loading. If you intend to reward your users with virtual currency for watching an ad, please read the section entitled “Adding Videos-For-Virtual-Currency”.

You must now decide when you want video ads to play, and if desired, which object will receive callbacks from AdColony about the ad. This section of the document contains the minimum steps necessary to play a video ad, as well as optional steps that may be necessary, depending on your application. If your application plays audio besides its use of AdColony, be sure to also read the section entitled “Advanced AdColony: the **`AdColonyTakeoverAdDelegate`**”.

— Step 1: Choose a Class to Play the Videos
Choose your Objective-C class from which you want to launch a full screen video ad; from this point forward, we will refer to this class as the `VideoPlayer`. In this example, we chose a UIViewController that is on screen when we want the ad to play. Please note that any class can act as a video player--it is not necessary for it to be a UIViewController. Open the header file of the `VideoPlayer` and import **`AdColonyPublic.h`**.

```
#import "AdColonyPublic.h"
@interface BasicViewController : UIViewController {
```

— Step 2: Play the Videos
In the implementation file of your `VideoPlayer` class, choose an execution point from where you want to begin playing an AdColony video. Now choose which video ad zone you want to use to play a video at this point, and retrieve either its zone ID or slot number. You previously setup your zones in your **`AdColonyDelegate`** within Step 7 of the previous section, and you should use the same information you entered in that step.

If you are referencing a zone directly by its zone ID, insert the following code and replace the comment with your zone ID string:

```
[AdColony playVideoAdForZone:/* insert your zone ID string here */];
```

If you are referencing a zone using its slot number, insert the following code and replace the comment with your slot number:

```
[AdColony playVideoAdForSlot:/* insert your slot number int here */];
```

IMPORTANT
If your app uses audio or music at any point, you must use the **`AdColonyTakeoverAdDelegate`** callbacks to pause and unpause your audio or music for the duration of video ads. Please refer to the section of this document entitled “Advanced AdColony: The **`AdColonyTakeoverAdDelegate`**” for

detailed instructions.

IMPORTANT
The playVideoAd* group of methods in the **`AdColony`** class will return immediately when you call them, and a video ad is not guaranteed to play after you call them. If you need to perform a specific action when the video begins playing, is finished playing, or does not play, you must use the **`AdColonyTakeoverAdDelegate`**. Please refer to the section of this document entitled “Advanced AdColony: The **`AdColonyTakeoverAdDelegate`**” for detailed instructions.

IMPORTANT
If your app changes the `UIWindow` hierarchy after initialization, use the **`AdColonyTakeoverAdDelegate`** to ensure that it does not happen while an ad is playing. Please refer to the section of this document entitled “Advanced AdColony: The **`AdColonyTakeoverAdDelegate`**” for detailed instructions.

— Checkpoint
Your app is now ready to play video ads! Build and run your app in an iOS simulator or on a device. After your app begins running, give AdColony time to prepare your ads with an active network connection 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.

---

5. 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. 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.

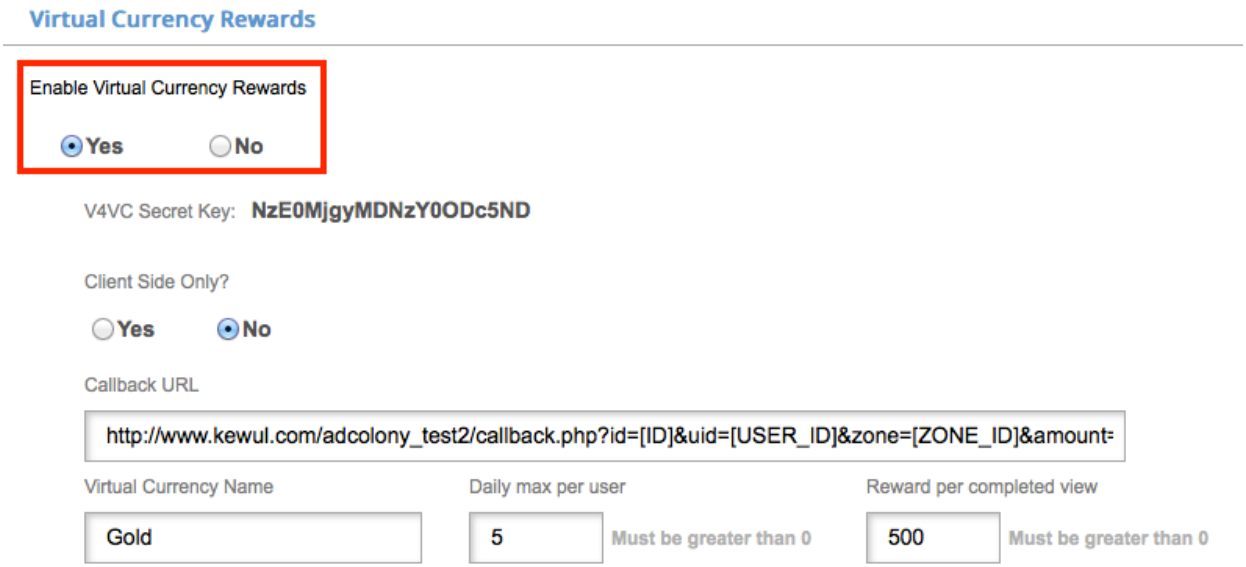
Please reference this best practices document for more information. Our best practices detail recommended usages and settings for V4VC.

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 adcolony.com

— Step 1
Sign into your 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?** V4VC Security 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.

— 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.

Setting Up V4VC in Your App

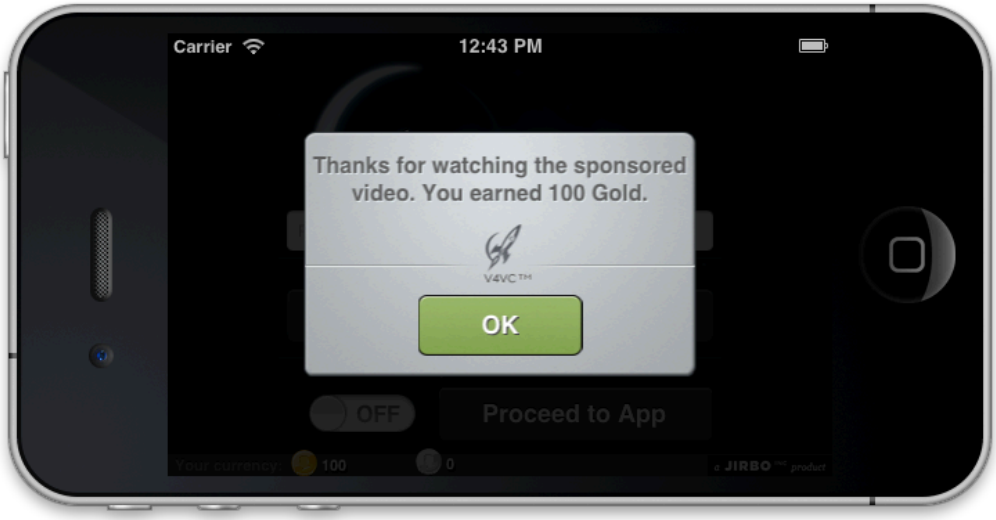
— Step 1: Choose a Class to Play Videos
Perform Step 1 of the previous section, Adding Video Ads, and remember which class you choose as your `VideoPlayer`.

— Step 2: Videos for Virtual Currency Using AdColony Popups
AdColony provides two default popups to provide the user information about V4VC. 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.

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 the pre-popup or post-popup, open the implementation file of the class acting as your `VideoPlayer` and find the execution point where you want a video ad to play. Now retrieve the zone ID or slot number of the video zone that you want to use for V4VC. You previously setup a zone for V4VC in the last section of this document, entitled “Configuring a Video Zone for V4VC on adcolony.com”.

If you are referencing a zone directly by its zone ID, insert the following code and replace the comments with desired values.

```
[AdColony playVideoAdForZone:/* insert your zone ID string here */
                   withDelegate:/* nil or a reference to a delegate */
                withV4VCPrePopup:/* YES or NO */
               withV4VCPostPopup:/* YES or NO */];
```

If you are referencing a zone using its slot number, insert the following code and replace the comments with desired values.

```
[AdColony playVideoAdForSlot:/* insert your slot number int here */
                   withDelegate:/* nil or a reference to a delegate */
                withV4VCPrePopup:/* YES or NO */
               withV4VCPostPopup:/* YES or NO */];
```

If you will use an **`AdColonyTakeoverAdDelegate`** to receive callbacks with information about the video play, pass a pointer to it for the withDelegate: parameter.

IMPORTANT
If your app uses audio at any point, you must use the **`AdColonyTakeoverAdDelegate`** callbacks to pause and unpause your audio for the duration of video ads. Please refer to the section of this document entitled “Advanced AdColony: The **`AdColonyTakeoverAdDelegate`**” for detailed instructions.

IMPORTANT
The playVideoAd* group of methods in the **`AdColony`** class will return immediately when you call them, and a video ad is not guaranteed to play after you call them. If you need to perform a specific

action when the video begins playing, is finished playing, or does not play, you must use the **`AdColonyTakeoverAdDelegate`**. Please refer to the section of this document entitled “Advanced AdColony: The **`AdColonyTakeoverAdDelegate`**” for detailed instructions.

IMPORTANT
If your app changes the `UIWindow` hierarchy after initialization, use the **`AdColonyTakeoverAdDelegate`** to ensure that it does not happen while an ad is playing. Please refer to the section of this document entitled “Advanced AdColony: The **`AdColonyTakeoverAdDelegate`**” for detailed instructions.

— Optional Step 1: Videos-For-Virtual-Currency™ Using Custom Popups

The following step explains how to implement custom popups before and after AdColony V4VC™ ads. It involves querying AdColony for server-configurable information about V4VC rewards, and details precautions you should take to make sure that your user interface reflects the behavior that users will experience. The default, optional AdColony popups have this logic included already. If you are comfortable using the default AdColony popups, skip this step; we recommend trying them out and returning to this step later if desired.

AdColony provides methods that return the currency name and reward amount associated with a zone. You should ask AdColony about the currency name and reward amount that is set for a zone on adcolony.com and incorporate it in your custom popups. Using the information passed back by AdColony ensures that your custom popups will reflect changes you make on adcolony.com to the currency name and reward amount, preventing user confusion. You should retrieve the zone ID or slot number for the zone you are using to play a V4VC ad, and pass it into the following methods.

If using zone ID:

```
    NSString* currencyName = [AdColony getVirtualCurrencyNameForZone:/*zone ID*/];
    int currencyAmount = [AdColony getVirtualCurrencyRewardAmountForZone:/*zone ID*/];
```

If using slot number:

```
NSString* currencyName = [AdColony getVirtualCurrencyNameForSlot:/*slot int*/];
    int currencyAmount = [AdColony getVirtualCurrencyRewardAmountForSlot:/*slot int*/];
```

Check to ensure that videos are ready to play using the zone ID or slot number of the zone you are going to use to play a V4VC ad. You will want to change your user interface to reflect when videos are unavailable so that users are not confused. Please refer to the section of this document entitled “Advanced AdColony: Checking Video Readiness” for detailed instructions for how to create callbacks that AdColony will use to notify you of the availability of video ads.

Check to see if the user has hit their daily reward cap. You will want to change your popup user interface to reflect when the user is at their daily play cap to encourage them to return tomorrow and to prevent users from thinking a video will play, then seeing no video.

If using zone ID:

```
    if(![AdColony virtualCurrencyAwardAvailableForZone:/*zone ID*/]) {
          //notify the user in your popup that the V4VC cap has been hit for today
}
```

If using slot number:

```
   if(![AdColony virtualCurrencyAwardAvailableForSlot:/*slot int*/) {
         //notify the user in your popup that the V4VC cap has been hit for today
}
```

— Step 3: Respond to V4VC Rewards

Implement `AdColony's adColonyVirtualCurrencyAwardedByZone:currencyName:currencyAmount:` callback in your AppDelegate.m file. The actions you should perform in this callback depend on whether you are using a client-side or server-side V4VC integration.

Client-Side

In a client-side V4VC integration, this callback is executed immediately after a video view that is eligible for a currency reward is completed (after the user presses **Continue** from the ad's closing-frame). You must increase your user's currency identified by *name* by the passed in *amount* for this system to work properly.

```
-(void)adColonyVirtualCurrencyAwardedByZone:(NSString *)zone currencyName:(NSString *)name
currencyAmount:(int)amount {
         //Increase the user's virtual currency balance here by the amount passed in
         //NOTE: This callback can be executed by AdColony at any time
         //NOTE: This is the ideal place for an alert about the successful reward
}
```

Server-Side

In a server-side V4VC integration, this callback is executed when a 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.

```
-(void)adColonyVirtualCurrencyAwardedByZone:(NSString *)zone currencyName:(NSString *)name
currencyAmount:(int)amount {
         //Update virtual currency balance by contacting the game server here
         //NOTE: The currency award transaction will be complete at this point
         //NOTE: This callback can be executed by AdColony at any time
         //NOTE: This is the ideal place for an alert about the successful reward
   }
```

IMPORTANT: In the event of a various network problems, a server-side currency transaction will not be instantaneous, which can result in this callback being executed by AdColony at any point during the execution of your application.

— Step 4: Handle Reward Failure Case (Server-side Only)

Implement `AdColony's adColonyVirtualCurrencyNotAwardedByZone:currencyName:currencyAmount:reason:` callback. This callback is made when a video is played, but for some reason, the currency award fails--for instance, if the server managing the currency is down. This callback will not be triggered in a client-

side integration. Apps may want to display an alert to the user here to notify them that virtual currency rewards are unavailable. AdColony passes a reason string that may be useful for debugging; it is not recommended to present this string to the user.

```
-(void)adColonyVirtualCurrencyNotAwardedByZone:(NSString *)zone
currencyName:(NSString *)name currencyAmount:(int)amount reason:(NSString *)reason{
      //Update the user interface after calling virtualCurrencyAwardAvailable here
}
```

Server-side Setup

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 handler 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 reward themselves 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 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.

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. The callback URL will use 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 adcolony.com for your virtual currency zone. See Step 3 of the section titled "Configuring a Video Zone for V4VC on 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

   $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;
   }

   //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 adcolony.com during the verification process.

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=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

The `AdColonyTakeoverAdDelegate`

The following section explains the purpose of the **`AdColonyTakeoverAdDelegate`** and, how to use it when playing videos, and a list of cases in which its use is recommended. The **`AdColonyTakeoverAdDelegate`** contains callbacks for when video ads begin, end, or were not displayed. Apps may need this information, for example, to pause audio when a video begins playing and resume audio when the ad has completed.

If your app uses audio at any point, you must use the **`AdColonyTakeoverAdDelegate`** callbacks to pause and unpause your audio for the duration of video ads.

If your app changes the `UIWindow` hierarchy after initialization, use the **`AdColonyTakeoverAdDelegate`** to ensure that it does not happen while an ad is playing.

—Step 1: Choosing a Delegate Class
In order to use the delegate, choose a class that you want to receive the callbacks and an instance of which will remain alive in memory until after the video has completed. In this example, we use the same UIViewController from a previous section that acted as our `AdColonyPlayer`. From this point forward in the document, we will refer to the class acting as our delegate as the `VideoDelegate`.

Open the header file of the `VideoDelegate` and import **`AdColonyPublic.h`**, then add the **`AdColonyTakeoverAdDelegate`** protocol to the class declaration.

```
#import "AdColonyPublic.h"
@interface BasicViewController : UIViewController <AdColonyTakeoverAdDelegate> {
```

— Step 2: Implement Desired Callbacks
Open the implementation file of your `VideoDelegate` and implement the callback methods of the **`AdColonyTakeoverAdDelegate`** protocol that you want to receive.

The **`adColonyTakeoverBeganForZone:`** method is the last thing that your program will execute before the video begins to play. In apps that play audio, you must pause all of your audio systems within this method.

```
- (void) adColonyTakeoverBeganForZone:(NSString *)zone {
    NSLog(@"AdColony video ad launched for zone %@", zone);
}
```

The **`adColonyTakeoverEndedForZone:withVC:`** method will be the first thing executed when the video

has finished playing and the user has dismissed it. In apps that play audio, you should resume all of your paused audio systems within this method.

```
- (void) adColonyTakeoverEndedForZone:(NSString *)zone
    withVC:(BOOL)withVirtualCurrencyAward {
    NSLog(@"AdColony video ad finished for zone %@", zone);
}
```

The **`adColonyVideoAdNotServedForZone:`** method will be called immediately if AdColony is unable to serve a video ad for any reason.

```
- (void) adColonyVideoAdNotServedForZone:(NSString *)zone {
    NSLog(@"AdColony did not serve a video for zone %@", zone);
}
```

— Step 3: Use Your Delegate When Playing Video Ads

In your `VideoPlayer`, locate the calls you made to play video ads. Select one of the similar AdColony playVideo* methods that takes a delegate parameter then pass in a reference to your `VideoDelegate` object.

The following are method calls you can use which take a delegate parameter. They will all result in a request to play an AdColony video, but some give you the option to display V4VC popups if you are creating a V4VC zone. Some methods take zone IDs while some take slot numbers. Choose an appropriate method call from below and pass the appropriate parameters.

Standard Video Ads:

```
[AdColony playVideoAdForZone:/* insert your zone ID string here */
                withDelegate:/* pass a reference to your VideoDelegate here*/];

[AdColony playVideoAdForSlot:/* insert your slot number int here */
                 withDelegate:/* pass a reference to your VideoDelegate here*/];
```

V4VC:

```
[AdColony playVideoAdForZone:/* insert your zone ID string here */
                            withDelegate:/* nil or a reference to a delegate */
                            withV4VCPrePopup:/* YES or NO */
                            withV4VCPostPopup:/* YES or NO */];

[AdColony playVideoAdForSlot:/* insert your slot number int here */
                            withDelegate:/* nil or a reference to a delegate */
                            withV4VCPrePopup:/* YES or NO */
                            withV4VCPostPopup:/* YES or NO */];
```

In our example, because we chose the same object to act as our `VideoPlayer` and our `VideoDelegate`, we can pass a self pointer when playing a video, as follows:

```
[AdColony playVideoAdForZone:/* zone ID */ withDelegate:self];
```

— Checkpoint
Build and run your app. Navigate through it and trigger a video to be played, then check the console log for the NSLog messages we inserted in the callbacks. Verify that the appropriate messages appear when the video ad begins and ends playing, and that your callbacks behave as desired.

Checking Video Readiness

In many apps, it is acceptable for no video to play when AdColony is asked to play a video. AdColony may not play a video when asked to if a zone is disabled via the adcolony.com control panel, if no ads are available, or if AdColony has not yet finished preparing ads when asked to play a video ad. In some apps, it may be necessary to know exactly when videos are ready to be played. If this is required, one should implement the **`adColonyNoVideoFillInZone:`**, **`adColonyVideoAdsReadyInZone:`**, and **`adColonyVideoAdsNotReadyInZone:`** callbacks in your class that implements the **`AdColonyDelegate`** protocol.

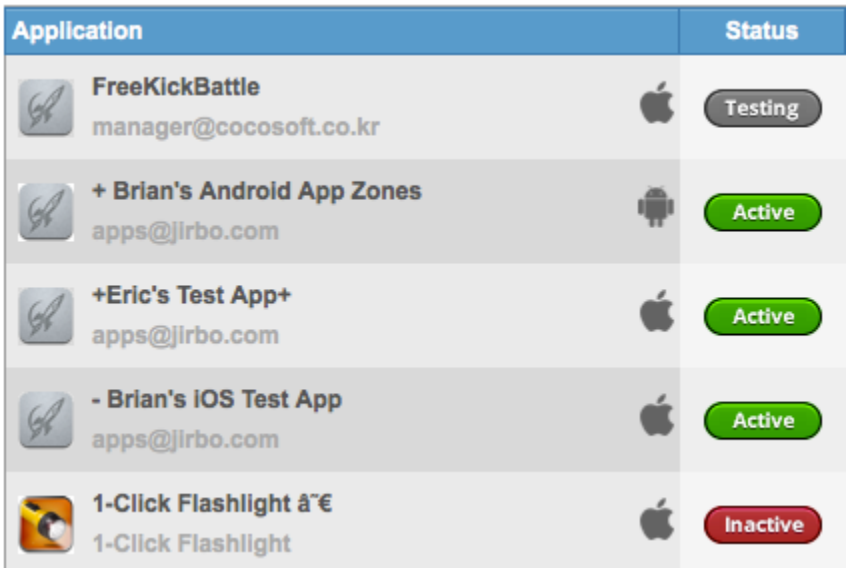
The **`adColonyNoVideoFillInZone:`** callback will be called by AdColony in the event that a zone has been disabled on the adcolony.com control panel or if the server is unable to be contacted. AdColony will call the method with an NSString* which contains the zone ID of the affected zone. If you receive this callback, the affected zone will not play video ads if you invoke a playVideoAd method on it.

The **`adColonyVideoAdsReadyInZone:`** callback will be called by AdColony immediately when video ads become available for a zone. AdColony will call the method with an NSString* which contains the zone ID of the affected zone. If you receive this callback, you can immediately invoke a playVideoAd method on the affected zone and expect a video ad to play.

The **`adColonyVideoAdsNotReadyInZone:`** callback will be called by AdColony immediately when video ads become unavailable in a zone. This can happen if the existing ads in a zone expire and AdColony is in the process of preparing new ads. AdColony will call the method with an NSString* which contains the zone ID of the affected zone. If you receive this callback, the affected zone will not play video ads until further notice by one of the other callbacks.

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 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 applications. They 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 build and link against the AdColonyPublic.h and libAdColony.a files present in the AdColonyLibrary folder distributed with the SDK.

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
Linker Errors	Ensure your iPhone Base SDK version is at least 4.0 Check for missing frameworks as per Step 3 of AdColony SDK Integration
Runtime Exception on Initialization of AdColony	Ensure that you have inserted the proper ‘Other Linker Flags’ as per Step 4 of AdColony SDK Integration
No Video Ads	For testing purposes, set your zone to receive test ads on 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.
Video Sound Present with Black Screen or Hidden Video	Determine which UIWindow you want the AdColony video to display within, and then call its makeKeyAndVisible method before telling AdColony to play a video. On an iPhone, the SDK identifies the UIWindow that is key and visible, then adds a UIView to it in front of all other UIViews. On an iPad, the SDK identifies the key UIWindow, creates its own UIWindow to use to display a video, and then when the video is completed makes the previously key UIWindow key and visible again.
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: The AdColonyTakeoverAdDelegate". AdColony does its best to avoid interference with your application's audio session, but playing audio during a video ad can cause problems.