



## Quick Start Guide for Apple iOS

AdColony Version 1.9.7

Updated October 10, 2011

### Table of Contents

#### **1. Introduction**

*A brief introduction to AdColony and its capabilities*

#### **2. Updating Applications and Changes to the Library**

*Instructions for users of previous versions of AdColony*

#### **3. AdColony SDK Integration**

*How to add AdColony to your application and link it with an account on [adcolony.com](http://adcolony.com)*

#### **4. Adding Video Ads**

*Detailed steps explaining how to prepare and display video ads at any point in your app*

#### **5. Adding Videos-For-Virtual-Currency™ (V4VC™)**

*Documents the usage of Videos-For-Virtual-Currency and how to integrate them within an existing virtual currency system*

#### **6. Advanced AdColony**

*Documents finely controlling video ad playback and advanced server-side controls*

#### **7. Integration With 3rd Party Networks and Aggregators**

*A note on integrating AdColony in apps with existing advertising systems*

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

---

## 1. Introduction

AdColony 1.9.7 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](http://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 E-mail: [support@adcolony.com](mailto:support@adcolony.com)

---

## 2. Updating Applications and Changes

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

### Drag and Drop Upgrading with AdColony

Many versions of AdColony for iOS can be upgraded simply by dragging the updated .a and .h files directly into your x-code project. In the case of certain older versions, such as those before 1.9.6, some ways of interacting with the AdColony SDK have changed. Please refer to both the change log within this section, and the function documentation in the header file to be sure that your Application utilizes the AdColony SDK in the most correct and useful way possible.

Users who are upgrading an app that contains AdColony 1.9.6 should replace the app's copies of files AdColonyPublic.h and libAdColony.a with the copies enclosed with this Quick Start Guide. It is also necessary to add the `CoreTelephony.framework` and weak-link it. For detailed directions see Step 3 of the AdColony SDK Integration section. After performing those changes, applications should continue to function as they did before without any other changes.

#### *Changes in AdColonyAdministratorDelegate protocol*

##### — Change 1

The `AdColonyAdministratorDelegate` protocol has been renamed `AdColonyDelegate` for clarity. A macro has been added to that defines the old name as the new name for backwards compatibility.

##### — Change 2

A new method, `adColonySupplementalVCParametersForZone:` was added. This method allows the developer to append parameters of his or her choice to the url used to query the server that awards virtual currency after playing a video.

#### *Changes in AdColonyAdministratorPublic class*

##### — Change 1

The `AdColonyAdministratorPublic` class has been renamed `AdColony` for clarity. A macro has been added to that defines the old name as the new name for backwards compatibility.

##### — Change 2

The `initAdministratorWithDelegate:` method has been renamed `initAdColonyWithDelegate:` for clarity. A macro has been added to that defines the old name as the new name for backwards compatibility.

#### — Change 3

Two methods, `zoneStatusForZone:` and `zoneStatusForSlot:` have been added so that one can determine the availability of video ads with a finer resolution. Please see `AdColonyPublic.h` for descriptions of the return values.

#### — Change 4

Two methods, `didVideoFinishLoadingForZone:` and `didVideoFinishLoadingForSlot:` have been deprecated, but remain fully functional. For future proofing, replace these methods with appropriate calls to the methods from Change 3.

#### — Change 5

Four methods have been added in order to implement a new feature: the ability of the developer to pause a playing AdColony video, either keeping it still on the screen or backgrounding it, exposing the app, and resuming the playback. The methods are `pauseVideoAdForZone:andGoIntoBackground:`, `pauseVideoAdForSlot:andGoIntoBackground:`, `unpauseVideoAdForZone:`, and `unpauseVideoAdForSlot:`.

#### *Changes in AdColonyTakeoverAdDelegate protocol*

#### — Change 1

Callback methods for pausing and unpausing the video ad have been added. The methods are `adColonyVideoAdPausedInZone:` and `adColonyVideoAdResumedInZone:`.

#### *Other Changes*

Methods that were previously deprecated, including everything that had to do with banner ads, have been removed.

---

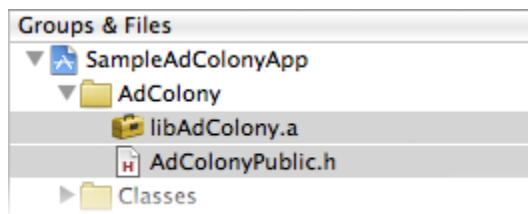
## 3. AdColony SDK Integration

### — Step 1: Choose a Project

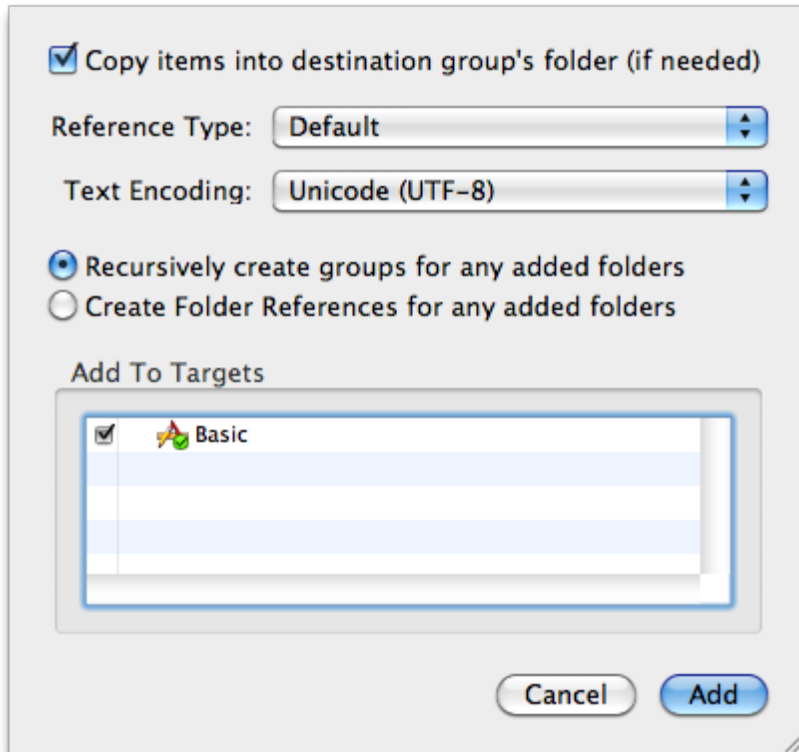
Use Xcode to open the existing project into which you want to integrate AdColony, or to create a new iOS project. AdColony requires you to select a Base iOS SDK of version 4.0 or greater.

### — Step 2: Add Library Files

Download the SDK and drag the `AdColonyPublic.h` and `libAdColony.a` files into the Xcode project.

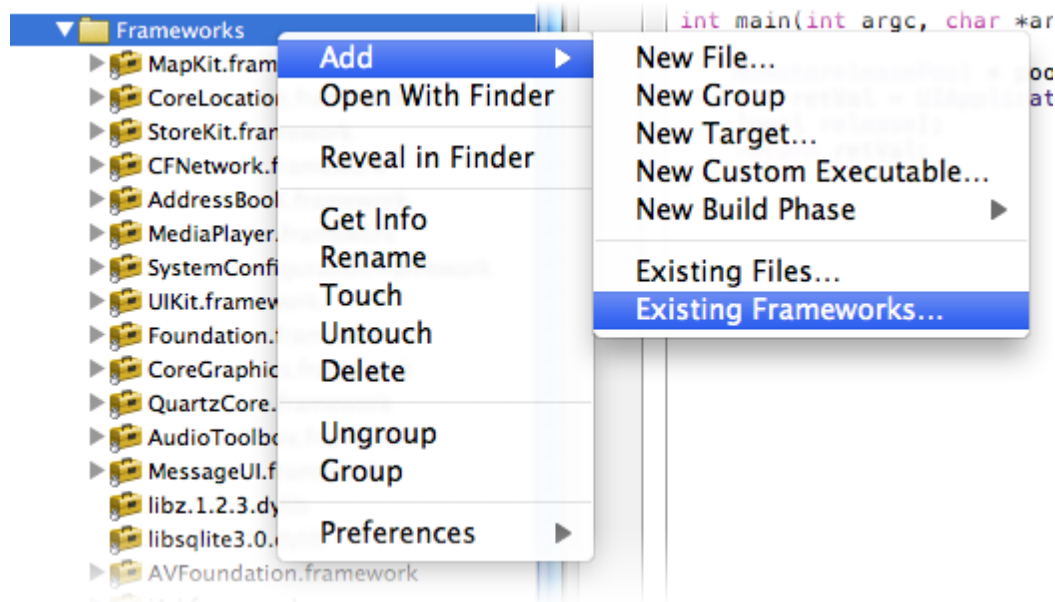


Ensure that they are copied into the project folder and added to all Targets which will utilize AdColony.

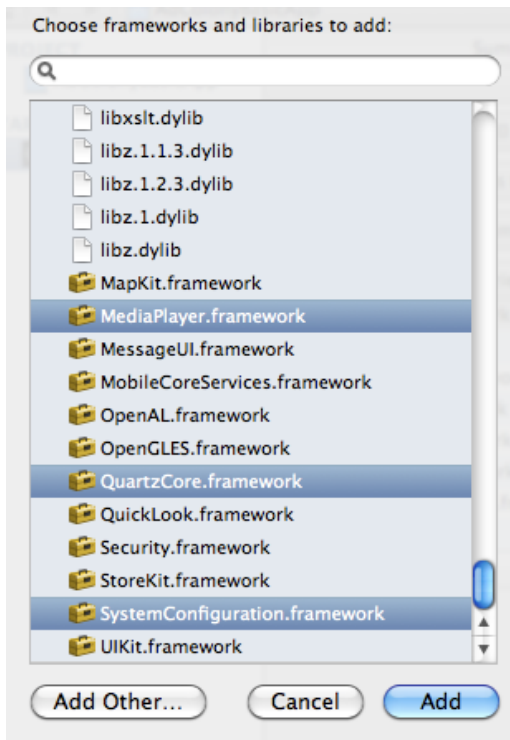


### — Step 3: Add Required Frameworks

Inside **Xcode**, right click on your Frameworks folder and select **Add > Existing 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 **"Weak"**.

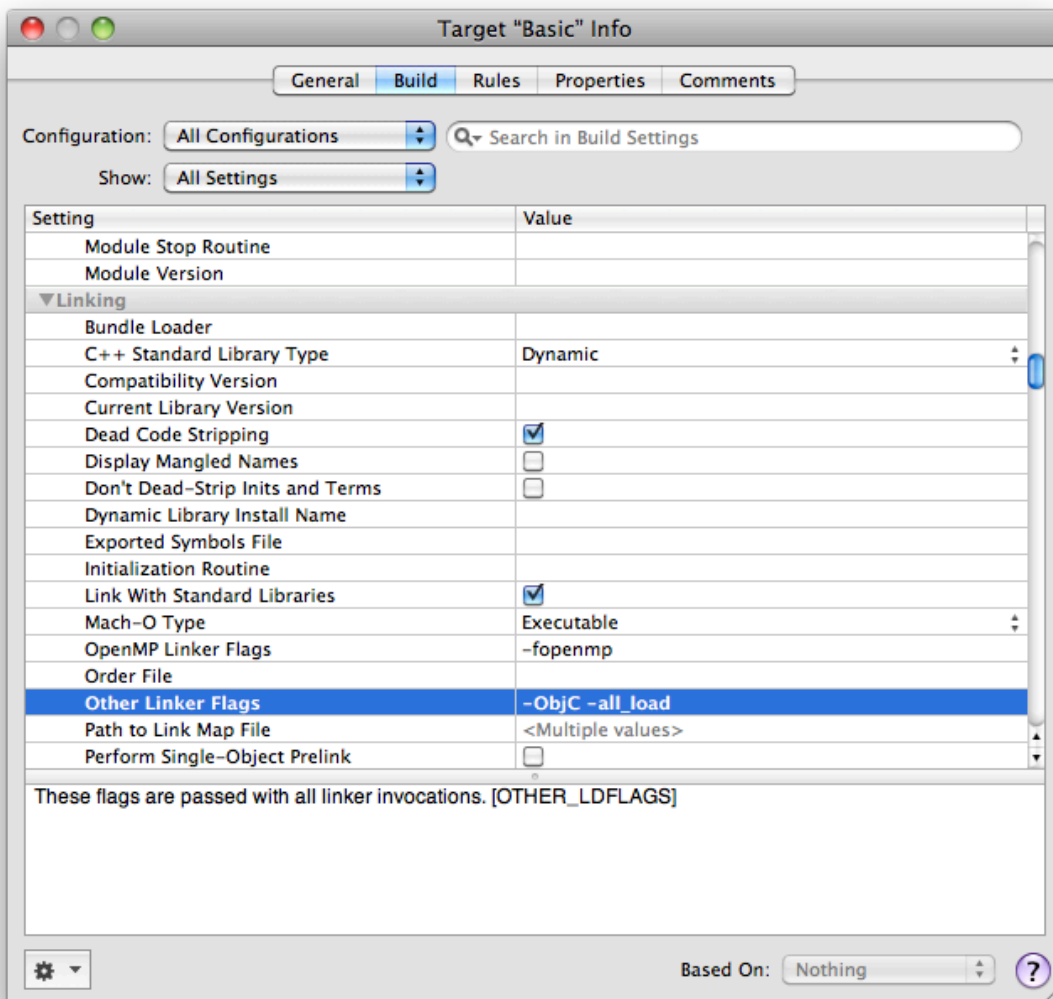
#### — Step 4: Set Linker Flags

Double click on your targets. Set the **Configuration** drop down menu to **All Configurations**. Select the **Build** tab, scroll to the **Linking** section, find the **Other Linker Flags** entry, then add the **"-all\_load"** and **"-ObjC"** flags.

#### NOTE:

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.

**PLEASE NOTE:** These flags must be added for your project to run successfully with AdColony. Omitting these flags will cause Run-Time Exceptions.



### — 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"
```

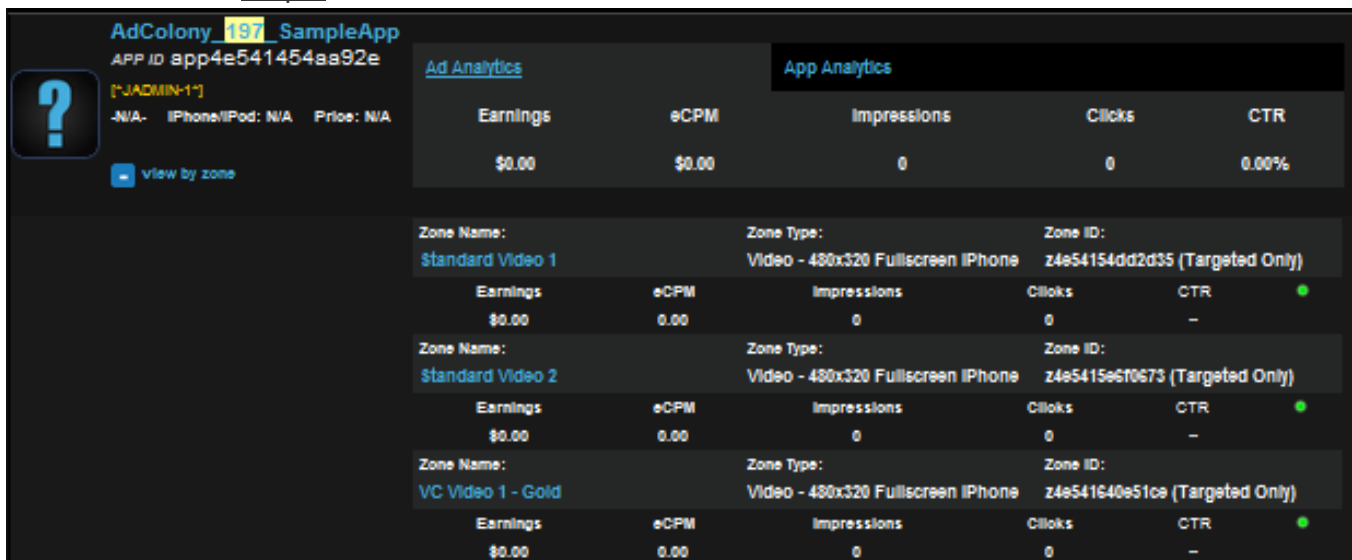
```
@interface AppDelegate : NSObject <UIApplicationDelegate, AdColonyDelegate> {
```

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];
}
```

### — Step 6: Gather Information from Your AdColony Account

Login to [adcolony.com](http://adcolony.com). If you have not already done so, create an app and needed zones on the website, following the instructions provided there. To receive test ads before an application is live on the app store, click the Edit Zone button on each zone and click the checkbox to Show Only Test Ads. Then retrieve your **app ID** and your corresponding **zone IDs** from the `AdColony` website and make note of them for use in [Step 7](#).



The screenshot shows the AdColony dashboard for an app named 'SampleApp' with ID 'app4e541454aa92e'. It displays a table of analytics for the app and a list of zones with their respective analytics.

| App Analytics |  | Earnings | eCPM   | Impressions | Clicks | CTR   |
|---------------|--|----------|--------|-------------|--------|-------|
|               |  | \$0.00   | \$0.00 | 0           | 0      | 0.00% |

| Zone Name         | Zone Type                         | Zone ID                        | Earnings | eCPM | Impressions | Clicks | CTR |
|-------------------|-----------------------------------|--------------------------------|----------|------|-------------|--------|-----|
| Standard Video 1  | Video - 480x320 Fullscreen iPhone | z4e54154dd2d35 (Targeted Only) | \$0.00   | 0.00 | 0           | 0      | —   |
| Standard Video 2  | Video - 480x320 Fullscreen iPhone | z4e5415e6f0673 (Targeted Only) | \$0.00   | 0.00 | 0           | 0      | —   |
| VC Video 1 - Gold | Video - 480x320 Fullscreen iPhone | z4e541640e51ce (Targeted Only) | \$0.00   | 0.00 | 0           | 0      | —   |

### — Step 7: Associate Your App With Your AdColony Info

Implement the `AdColonyDelegate` protocol methods within your chosen delegate class's (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](http://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: 1.9.7” to the console, indicating it has been initialized. If you encounter any problems, double check the previous steps.

*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 account.*

---



## 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. 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. 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

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 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**

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. In addition to the standard setup for video ads, one must set up 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.

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

### Modifying Your Server to Reward Virtual Currency Users

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.

In order to reward your users with the virtual currency rewarded by 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.

#### — Step 1: Create a URL

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. Once you have chosen this URL, you should input it in the video zone configuration page on [adcolony.com](http://adcolony.com) for your virtual currency zone. You will want to create this URL in a directory that can execute server-side code such as PHP.

#### — 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=[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 (iOS device 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 |

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.

```
<?php

$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";

?>

```

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](http://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=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<br>Duplicate transaction which was already rewarded | AdColony finishes transaction |
| vc_decline | uid was not valid<br>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.

## Configuring a Video Zone for Virtual Currency on [adcolony.com](http://adcolony.com)

### — Step 1

Sign into your [adcolony.com](http://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 callback URL field with the URL on your server that you created in the previous section.

## Using Videos for Virtual Currency 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`.

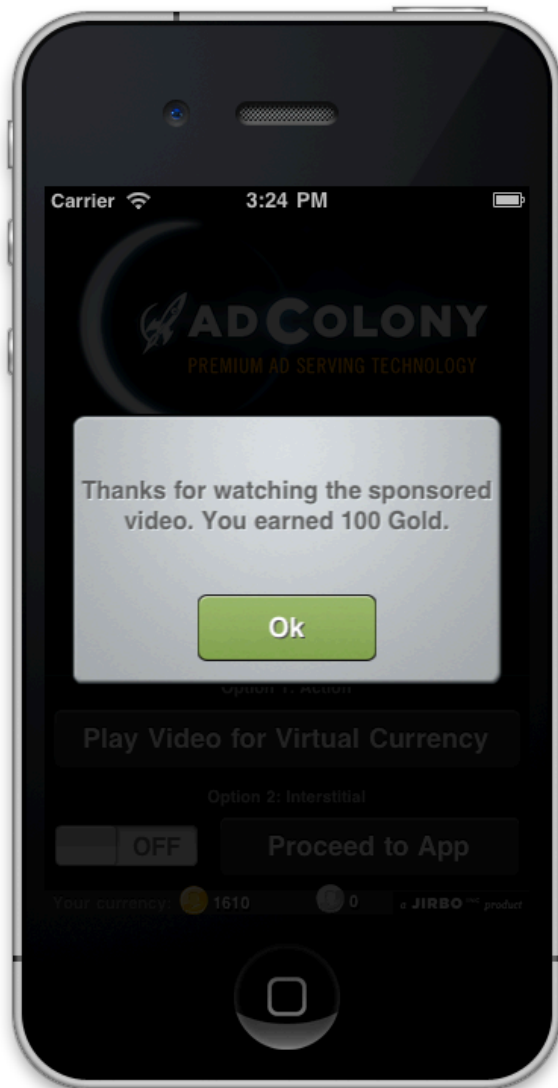
### — Step 2: Videos for Virtual Currency Using AdColony Popups

AdColony provides two default popups to provide the user information about V4VC. 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. These popups include information you entered on [adcolony.com](http://adcolony.com), informing users of the name and amount of currency they will receive. You may choose to use these popups or to ignore them. Many apps implementing V4VC implement their own custom popups to match the app's look.

The pre-popup currently has the following appearance:



The post-popup currently has the following appearance:



To use 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 Virtual Currency on [adcolony.com](http://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 */
    withV4VCPopup:/* 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 */  
    withV4VCPopup:/* 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

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 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

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](http://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](http://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. 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 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

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. Apps may want to display an alert to 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
}

```

---

## 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 unpaue 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 */
    withV4VCPopup:/* YES or NO */];

[AdColony playVideoAdForSlot:/* insert your slot number int here */
    withDelegate:/* nil or a reference to a delegate */
    withV4VCPrePopup:/* YES or NO */
    withV4VCPopup:/* 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](http://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.

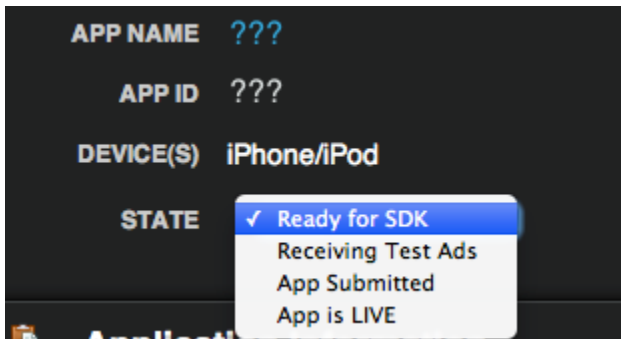
The **adColonyNoVideoFillInZone:** callback will be called by AdColony in the event that a zone has been disabled on the [adcolony.com](http://adcolony.com) control panel or if the server is unable to be contacted. AdColony will call the method with an **NSString\*** which contains the zone ID of the 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 State

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



The following is a detailed explanation of each state:

| <u>STATE</u>       | <u>DESCRIPTION</u>  |
|--------------------|---|
| 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.  |

---

## How to Receive Test Ads

On the AdColony Publisher Portal:

1. Publishers->Click on your app's link->**Edit App**.
2. Ensure your App's **STATE** is 'Ready for SDK' (if not, select 'Ready for SDK' from dropdown menu and click **Update Application**).
3. Click on **Zones**; next, click on the link for the video zone in which you'd like to show test ads.
4. Tick the, **Only Show Test Ads** box and click **Update My Zone**. Do this for each video zone in your app.

Your app will now receive AdColony test ads.

## How to Receive Live Ads

When you app goes live, please make sure to follow the steps below to receive live ads:

1. Publishers->Click on app link->**Edit App**.
2. Paste your app store URL in the **App Store URL Link** field and select 'App is Live' from the **App STATE** dropdown menu; click **Update Application**.
3. Click on **Zones**; next, click on the link for the video zone in which you'd like to receive live ads.
4. Un-check **Only Show Test Ads** box, review zone caps and V4VC settings and click **Update My Zone**. **Important: Do this for each video zone in your app.**

You will now receive live ads.

## 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.millennialmedia.com>)
  - Mobclix [version 4.1.6] (<http://www.mobclix.com>)
- 

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

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

**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](mailto:support@adcolony.com)

| Issue                  | Resolution  |
|------------------------|---|
| Assorted Linker Errors | Ensure your iPhone Base SDK version is at least 4.0<br>Check for missing frameworks as per Step 3 of AdColony SDK Integration   |
| Missing Video Ads      | Implement the optional <code>adColonyVideoAdNotServed</code> callbacks in your <code>AdColonyTakeoverAdDelegate</code> and log messages from within them. The AdColony servers are occasionally unable to provide ads, in which case AdColony will notify you with callbacks. |



|                   |  |
|-------------------|--|
| Missing Video Ads | <p>Ensure that the <code>AdColonyVideoAdPresenterPublic</code>'s <code>didVideoFinishedLoading</code> method returns <b>YES</b> before attempting to play a video (for more details, see <a href="#">Step 4</a> of the <b>Adding Video Ads</b> section). If the method call returns <b>NO</b> indefinitely, implement the <code>AdColonyDelegate</code> protocol's <code>noVideoFillInZone:</code> callback. If AdColony calls the callback, then the server is unable to provide video ads. If that happens, check to ensure that you have set a valid app ID and video zone ID in <a href="#">Step 7</a> of the <b>AdColony SDK Integration</b> section.</p> |
|-------------------|--|