



# OOYALA SDK FOR IOS GUIDE

# CONTENTS

<b>LATEST RELEASE NOTES: MOBILE SDK FOR IOS V2.4.0</b>	<b>5</b>
<b>ARCHIVED RELEASE NOTES: MOBILE SDK FOR IOS</b>	<b>6</b>
<b>ABOUT THE MOBILE SDK FOR IOS</b>	<b>7</b>
<b>GETTING STARTED: SETTING UP YOUR ENVIRONMENT</b>	<b>9</b>
<b>GETTING STARTED: A SAMPLE APPLICATION</b>	<b>11</b>
<b>SAMPLE APPLICATIONS FOR THE IOS MOBILE SDK</b>	<b>13</b>
<b>LAYOUT CONTROLS</b>	<b>14</b>
Implementing Custom Controls for iOS	14
<b>WORKING WITH EVENTS</b>	<b>15</b>
<b>DEALING WITH IOS MOBILE ERRORS</b>	<b>16</b>
<b>WORKING WITH CLOSED CAPTIONS</b>	<b>18</b>
Required Closed Caption Setup in Backlot	18
Fundamental Closed Caption Functions on iOS	18
iOS FCC 708 Closed Captions Compliance	19
<b>LOCALIZING THE USER INTERFACE</b>	<b>20</b>
<b>INTEGRATION WITH FREEWHEEL ON IOS</b>	<b>21</b>
See the Freewheel Sample App in Action on iOS	22
A Closer Look at the iOS Sample App and Integration Touchpoints	24
<b>INTEGRATION WITH GOOGLE IMA ON IOS</b>	<b>28</b>
See the IMA Sample App in Action on iOS	29
A Closer Look at the iOS Sample App and Integration Touchpoints	30
Overriding the Ad Tag URL on iOS	34



<b>INTEGRATION WITH OMNITURE ON IOS</b>	<b>35</b>
<b>WORKING WITH MULTI-RESOLUTION STREAMS ON WIDEVINE</b>	<b>40</b>
<b>CROSS-DEVICE RESUME (XDR) WITH THE MOBILE SDK</b>	<b>41</b>
<b>XTV CONNECT SDK FOR IOS</b>	<b>42</b>
Create an iOS Development Project for XTV	43
Using XBMC to Configure and Test DLNA Playback	47



# COPYRIGHT NOTICE

---

Copyright Ooyala 2008-2014

Ooyala, Backlot, Ooyala Actionable Analytics, the Ooyala logo, and other Ooyala logos and product and service names are trademarks of Ooyala, Inc. ("Ooyala Marks"). Company agrees not to remove any Ooyala Marks that are contained within and/or affixed to the Services as provided to Company. Except with respect to the foregoing, Company agrees not to display or use in any manner the Ooyala Marks without Ooyala's prior written permission. All contents of the Ooyala website and Services are: Copyright 2008-2014. Ooyala, Inc. All rights reserved. Ooyala and Backlot are registered trademarks of Ooyala, Inc. in the United States, Japan, and European Community. All rights reserved.

For complete information on terms of service, see: <http://www.ooyala.com/tos>

All other trademarks are the property of their respective companies.

This content was last updated on 2014-06-12.



# LATEST RELEASE NOTES: MOBILE SDK FOR IOS V2.4.0

---

Release notes for the mobile SDK v2.4.0 for iOS include API changes, Closed Caption updates, Freewheel and Google IMA fixes, and updated support for the Adobe Pass SDK and x86\_64 architecture.

## NOTABLE CHANGES IN VERSION 2.4.0

**Note:** These changes may require you to update your integration.

Version 2.4.0 of the Ooyala Mobile SDK for iOS has been released. Version 2.4.0 includes:

- Major Changes to Closed Captions UI and Closed Captions API.
- To comply with FCC Regulation, we require the inclusion of `QuartzCore.framework` and `MediaAccessibility.framework`. You may mark `MediaAccessibility.framework` as optional.
- A new `OOPlayerDomain` class that must be used for all instances of `OOOoyalaPlayerViewController`.
- The domain you provide to the `PlayerDomain` class now must begin with `http://` to match domain analytics ingestion.

## FULL RELEASE NOTES FOR VERSION 2.4.0

### API Changes

- Extracted 'player domain' as a concrete type and put validation enforcement in there, and updated APIs using such. Fixed up sample projects to use `OOPlayerDomain`.
- Now throw an uncaught exception if domain string looks bad.
- Pass all available resolutions to iOS devices, as opposed to filtering.
- Added `setControlsViewControllers` method.
- Added `setLiveScrubberShowing` method.
- Major Closed Captions UI Updates to adhere to FCC regulations. Make sure `QuartzCore.framework`, and `MediaAccessibility.framework` ('optional') are included in your applications.

### Freewheel/IMA Bug fixes

- Fixed the bad-ad-infinite-loop when FW returns an actual error via `FW_NOTIFICATION_REQUEST_COMPLETE`.
- Fix for view frame sizing when fullscreen for both iOS6 and iOS7.
- Fixed IMA postroll for iOS.
- Now pass current item duration into Freewheel instead of Stream Player duration.
- Now avoid adjusting the Learn More UI for Google IMA ads.
- Attempt to fix race where FW comes back before we've set up our player as delegate.

### Other Changes

- Updated Adobe Pass SDK to 1.7.6.
- Renamed JSONKit category methods to avoid name collisions with other categories.
- Added support for x86\_64 as a valid architecture.
- Removed iOS 4.3-specific code.
- Fixed memory leak with Ooyala Player/Closed Captions View Controllers.

### Documentation Changes

- Removed reference to FreeWheel libraries folder. It no longer exists.
- Changed description of FreeWheel Headers file indicating it is an Ooyala file not a FreeWheel file.



# ARCHIVED RELEASE NOTES: MOBILE SDK FOR IOS

---

Archived release notes for the mobile SDK for iOS.

## NOTABLE CHANGES IN VERSION 2.1.0

Version 2.1.0 of the Ooyala Mobile SDK for iOS has been released. Version 2.1.0 includes:

- Support for Ooyala's Cross-device Resume (XDR): the ability for a viewer to stop viewing on one device and resume on a different one.
- Support for Apple iOS7 user interface improvements.
- Customer-reported and unreported optimizations and bugfixes.

**Note:** In this SDK with version 2.1.0, the `BaseMoviePlayer` class been renamed `BaseStreamPlayer`.



# ABOUT THE MOBILE SDK FOR IOS

---

The Ooyala Mobile SDK for Apple iOS is a set of Objective C classes and layout controllers.

This documentation is for the current and prior supported versions of the Mobile SDK for iOS.

With the SDK you can rapidly develop rich, custom video application experiences for mobile devices. The SDK includes methods/classes to play videos, display VAST and Ooyala ads, query Ooyala content, and more, including but not limited to APIs for

- Playback
- Playlists/Channels - Thumbnails
- Info
- Related Media
- Live (New)
- Ads
- Analytics
- Fully integrated analytics to understand mobile usage
- Closed Captions
- APIs for loading and displaying closed captions through DFXP files

## DOWNLOADING THE MOBILE SDK

You can download the Mobile SDK from <http://support.ooyala.com/users/resources/mobile-and-client-sdks>. Click the appropriate version to download a zipfile of the Mobile SDK package.

## SUPPORTED CONFIGURATIONS

The Ooyala Mobile SDK for iOS supports:

- All devices with iOS 5 or later:
  - iPhone 3GS and later
  - iPod Touch, 3rd generation and later
  - iPad 1 and later
- HTTP Live Streaming (HLS)
- MP4 video container (subject to Apple's cautions and requirements)

## REQUIRED SKILLS

Ooyala assumes that you have the necessary programming skill or prior development experience to work productively with and with curiosity about the Mobile SDK. You should be able to investigate, analyze, and understand the SDK's source code.

## GENERAL DESIGN OF THE SDK

In general, the SDK has two parts:

1. "Content management" functions, which include loading a video and its associated metadata from embed code (content identifiers), as well as querying for more videos, and "video playback," which covers controlling video playback on a predefined "skin".

These functions are essential for playback.



2. "Layout controls" of the user interface (UI), the provided skins, and positioning of the "video surface" within the application.

These controls can be replaced entirely, either from scratch or with the provided controls as a starting point.

## REFERENCE DOCUMENTATION FOR THE MOBILE SDK

The Mobile SDK comes with complete reference documentation.

To see the programming documentation for the Ooyala Mobile SDK, after unzipping the package, with your browser or other tool, open the file `Documentation/index.html`

## ABOUT ACCESS CONTROL

The SDK is designed to work a minimum of security constraints. If you want to restrict access, you have the following options. Any other access control you might require is beyond the scope of the SDK but is available through Ooyala, such as Digital Rights Management (DRM) systems or other configurations.

- You can restrict the playback of videos to an Internet domain you have configured in Ooyala Backlot, either through the Backlot UI's Syndication Controls detailed in the *Backlot User Guide* or with the Backlot API's publishing rule routes detailed in the *Backlot API Reference*. This is discussed in the tutorial.
- You can use Ooyala Player Token (OPT) to control access to individual assets (videos). (For documentation on Ooyala Player Token, see the Ooyala documentation site.)

If you want to use OPT, you need to implement the SDK's `OOEmbedTokenGenerator` interface. For an example, see any of the sample applications, described briefly in [Sample Applications for the iOS Mobile SDK](#) on page 13. For documentation about OPT, see [Ooyala Player Token](#) in the *Ooyala Content Protection Developer Guide*.

## INTEGRATION WITH FREEWHEEL

Ooyala offers an add-on to the Mobile SDK to integrate with the Freewheel ad service. For more details, see [Integration with FreeWheel on iOS](#) on page 21.

## INTEGRATION WITH OMNITURE

Ooyala offers an add-on to the Mobile SDK to integrate with Omniture analytics. For more details, see [Integration with Omniture on iOS](#) on page 35.





# GETTING STARTED: SETTING UP YOUR ENVIRONMENT

---

A few short steps are all that's needed to setup the SDK in the popular Xcode development environment.

You need the following to get started with the SDK:

- The Mobile SDK for iOS itself

You can download the Mobile SDK from <http://support.ooyala.com/users/resources/mobile-and-client-sdks>. Click the appropriate version to download a zipfile of the Mobile SDK package.

- The Xcode 4.2 or later development environment for iOS applications and Apple iOS SDK 5 available at <http://developer.apple.com/xcode/>
- Your Ooyala-supplied provider code (pcode). To see your pcode, login to the Backlot UI, and go to the **ACCOUNT** tab, **Developers** subtab. The pcode is in the upper left.
- The embed code (content ID or asset ID) of a video you want to play

To get started:

1. Unzip the downloaded OoyalaSDK-iOS.zip file.
2. Locate the libOoyalaSDK.a from within the OoyalaSDK-iOS.zip and choose an installation location.
3. Start Xcode (default: /Developer/Applications/Xcode.app).
4. Click **Create a new Xcode project**.
5. When prompted, click **Single View Application**.  
The Choose options for your new project dialog box appears.
6. Do the following:
  - a) Enter a name in the Product Name field.
  - b) Enter your identifier in the Company Identifier field.
  - c) Select iPhone from the Device Family list box.
  - d) Deselect the Use Storyboards and Include Unit Tests check boxes.
  - e) Click Next.
7. Choose a location for your project and click **Next**.  
The project is created.
8. Select **Add Files** from the **File** menu.  
You are prompted to select a file.
9. Navigate to the directory where you unzipped the OoyalaSDK-iOS.zip file.
10. Do the following:
  - a) Select the libOoyalaSDK-iOS.a file and the Headers directory.
  - b) Select the **Copy Items into Groups Destination Folder** (if needed) check box.
  - c) Click **Add**.
11. In the left pane, click the name of the project. Then, click the name of the target in the **Targets** list. The target configuration page appears with the **Summary** tab displayed.
  - a) Click the **Build Phases** tab.
  - b) Expand **Link Binary with Libraries**.
  - c) Click the plus symbol (+) to add the following frameworks:

```
AVFoundation.framework
CFNetwork.framework
CoreMedia.framework
CoreText.framework
MediaAccessibility.framework (optional for FCC regulation compliance)
```



```
MediaPlayer.framework
Security.framework
SystemConfiguration.framework
QuartzCore.framework (for FCC regulation compliance)
```

- d) Click the **Build Settings** tab.
- e) Double-click **Other Linker Flags**. A dialog box opens.
- f) Click the plus (+) in the lower left corner and enter: `-lstdc++`

12. In the left pane, click `ViewController.h`.

13. Add the following beneath the import statement: `@class OOOoyalaPlayerViewController;`

14. Add the following beneath the interface: `@property (nonatomic, strong) OOOoyalaPlayerViewController *ooyalaPlayerViewController;`

15. In the left pane, click `ViewController.m`.

16. Add the following beneath the import statements: `#import "OOOoyalaPlayerViewController.h" #import "OOOoyalaPlayer.h"`

17. Add the synthesize command below the ViewController implementation: `@synthesize ooyalaPlayerViewController;`

18. Add the following beneath the synthesize statement, substituting your own credentials:

```
NSString * const PCODE = @"<pcode> ";
NSString * const APIKEY = @"<api_key> ";
NSString * const SECRETKEY = @"<secret_key> ";
NSString * const PLAYERDOMAIN = @"http://<domain> ";
```

To get this information, open Backlot, click the **Account** tab, and click the **Developers** subtab.

**Note:** The `<domain>` specifies which domain from which the content can play. The domain is a string match; if the domain where the content is hosted contains the string, playback is allowed.

19. Add the following after the `[super viewDidLoad];` line, replacing the embed code with the embed code of one of your assets:

```
ooyalaPlayerViewController = [[OOOoyalaPlayerViewController alloc]
initWithAPIKey:APIKEY secret:SECRETKEY pcode:PCODE
domain:[[OOPlayerDomain alloc] initWithString:PLAYERDOMAIN]];
[ooyalaPlayerViewController.view setFrame:self.view.bounds]; [self
addChildViewController:ooyalaPlayerViewController]; [self.view
addSubview:ooyalaPlayerViewController.view];
[ooyalaPlayerViewController.player setEmbedCode:@"<embed_code>"];
```

20. Click **Run**

A simulated iPhone appears with an embedded video player.

#### Notes:

- If you have problems viewing a video, make sure the syndication settings are set to allow iPhone and iPad devices. For more information, go to the Publishing Rules section of the Ooyala Backlot User Guide.
- If the video drops out and the audio continues to play, you might have a low bandwidth connection. This ensures that viewers can still experience the content and is the expected behavior designed by Apple.
- If you are using Xcode 4.5 and you are encountering compiling errors, remove `armv7s` from the **Valid Architectures** on the **Build Settings** tab.



# GETTING STARTED: A SAMPLE APPLICATION

A few short steps are all that's needed to setup the SDK in the popular Xcode development environment.

You need the following to get started with the SDK:

- The Mobile SDK for iOS itself

You can download the Mobile SDK from <http://support.ooyala.com/users/resources/mobile-and-client-sdks>. Click the appropriate version to download a zipfile of the Mobile SDK package.

- The Xcode 4.2 or later development environment for iOS applications and Apple iOS SDK 5 available at <http://developer.apple.com/xcode/>
- Your Ooyala-supplied provider code (pcode). To see your pcode, login to the Backlot UI, and go to the **ACCOUNT** tab, **Developers** subtab. The pcode is in the upper left.
- The embed code (content ID or asset ID) of a video you want to play

The best way to become familiar with the Mobile SDK for iOS is to run one of the sample applications.

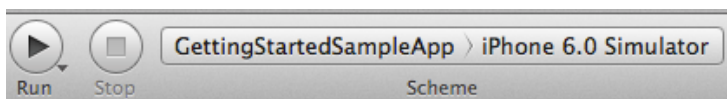
1. Unzip the downloaded OoyalaSDK-iOS.zip file.
2. Open the file OoyalaSDK-iOS/SampleApps/GettingStartedSampleApp/GettingStartedSampleApp.xcodeproj  
The source code is displayed in Xcode. On the upper left, switch to hierarchical view, expand ViewController, and select the ~viewDidLoad member.
3. On the appropriate lines, between the quotation marks for each constant or variable, paste your provider code (pcode) and your embed code.

The pertinent lines should look similar to the following

```
NSString * const PLAYERDOMAIN = @"http://www.ooyala.com";
NSString * const EMBED_CODE = @"your embed code here";
NSString * const PCODE = @"your pcode here";

.
.
.
[ooyalaPlayerViewController.player setEmbedCode:@"EMBED_CODE"];
```

- EMBEDCODE: The identifier for one of your video assets.
  - PCODE: Your Ooyala-supplied provider code
  - PLAYERDOMAIN: Leave unchanged as <http://www.ooyala.com>. This constant works in conjunction with **Syndication Controls** (publishing rules) in Backlot. If you have set Internet domain restrictions on videos in Backlot (see the *Backlot User Guide*), the constant here can be set to one of those allowed domains. If you have not set these **Syndication Controls**, the constant has no effect.
  - setEmbedCode( ) function: The embed code of the video you want to play as the argument. This function is the primary "work horse" of the SDK.
4. Save your changes.
  5. In the upper left, select **iPhone 6.0 Simulator**.



6. In the upper left, click **Run**.

The iPhone Simulator appears and your video is loaded. Click the play button to view the video.



**Note:**

1. If you have problems viewing a video, make sure the syndication settings are set to allow iPhone and iPad devices. For more information, go to the "Publishing Rules" section of the Ooyala *Backlot User Guide*.
2. If the video drops out and the audio continues to play, you might have a low bandwidth connection. This ensures that viewers can still experience the content and is the expected behavior designed by Apple.



# SAMPLE APPLICATIONS FOR THE IOS MOBILE SDK

---

The Mobile SDK comes with several sample applications, located in `OoyalaSDK-iOS/SampleApps`.

Name	Location in SDK	Description
DeviceManagementSampleApp	<code>DeviceManagementSampleApp</code>	A sample to illustrate the SDK device handling.
Getting Started	<code>SampleApps/GettingStartedSampleApp</code>	A basic program to illustrate the SDK. See <a href="#">Getting Started: A Sample Application</a> on page 11



# LAYOUT CONTROLS

---

The `DefaultControlsSource` directory contains the default Ooyala player skin.

This directory includes source C programs (\*.m) and header files (\*.h) that control the layout of the video player on the device, both for the default and for inline or fullscreen display, including the following:

- `OOControlsViewController`: The primary program that invokes the desired layout controls
- `OOFullscreenViewController`: Controller for fullscreen video mode
- `OOFullscreenControlsView`: Controller for fullscreen video mode
- `OOImages`: Control the shapes of icons displayed in the player
- `OOInlineControlsView`: Controls for inline video mode
- `OOInlineViewController`: Controller for inline video mode
- `OoyalaPlayerViewController`: The primary program that invokes the desired layout controls
- `OOUIProgressSlider`: Set characteristics of the progress slider (also called the "scrubber"), such as the image for the slider itself, the display of time elapsed or remaining, and more
- `OOUIUtils`: Utility methods
- `OOVolumeView`: Set characteristics of the volume slider, such as the color or shape

## IMPLEMENTING CUSTOM CONTROLS FOR IOS

---

Ooyala's iOS SDK provides you with tools for overriding the default fullscreen and inline controls. The following steps explain how to run custom controls over the `OoyalaPlayer`.

1. Download [the Ooyala SDK](#).
2. Open the `DefaultControlsSource` folder provided with the SDK.
3. Copy the entire `DefaultControlsSource` folder into your application.
4. Rename `OOInlinelOS7ViewController` and `OOInlinelOS7ControlsView` (for example, `CustomControlsViewController` and `CustomControlsView`).
5. instantiate an `OoyalaPlayerViewController` and add the following line of code:

```
[_ooyalaPlayerViewController setInlineViewController:
[[CustomControlsViewController alloc] init]];
//or
[_ooyalaPlayerViewController setFullscreenViewController:
[[CustomControlsViewController alloc] init]];
```

Your custom controls will now be used in the Ooyala Player. You can confirm this by adding/removing views from your custom controls or by adding logging. **Note:** Do not simply make changes to the provided source code; such changes might be lost in upgrade of future versions of the SDK.



# WORKING WITH EVENTS

---

You need to listen for event notifications on the player object.

First you need to set up notification objects, as shown below. Events are enumerated in the header file `Headers/OOYalaPlayer.h` with explanatory comments:

```
ios Notifications:
// notifications
extern NSString const OOYalaPlayerTimeChangedNotification;
/* Fires when the Playhead Time Changes */
extern NSString const OOYalaPlayerStateChangedNotification;
/* Fires when the Player's State Changes */
extern NSString const OOYalaPlayerContentTreeReadyNotification;
/* Fires when the content tree's metadata is ready and can be accessed */
extern NSString const OOYalaPlayerAuthorizationReadyNotification;
/* Fires when the authorization status is ready and can be accessed */
extern NSString const OOYalaPlayerPlayStartedNotification;
/* Fires when play starts */
extern NSString const OOYalaPlayerPlayCompletedNotification;
/* Fires when play completes */
extern NSString const OOYalaPlayerCurrentItemChangedNotification;
/* Fires when the current item changes */
extern NSString const OOYalaPlayerAdStartedNotification;
/* Fires when an ad starts playing */
extern NSString const OOYalaPlayerAdCompletedNotification;
/* Fires when an ad completes playing */
extern NSString const OOYalaPlayerAdSkippedNotification;
/* Fires when an ad is skipped */
extern NSString const OOYalaPlayerErrorNotification;
/* Fires when an error occurs */
extern NSString const OOYalaPlayerAdErrorNotification;
/* Fires when an error occurs while trying to play an ad */
extern NSString const OOYalaPlayerMetadataReadyNotification;
/* Fires when content metadata is ready to be accessed */
```

For an example, see [Dealing with iOS Mobile Errors](#) on page 16.



# DEALING WITH IOS MOBILE ERRORS

---

You need to listen for error notifications on the player object.

Note: When dealing with iOS Mobile Errors, you also need to remember to provide and display useful error messages for your app to your users.

To deal with iOS mobile SDK errors, you need to first declare the following:

```
.  
.   
.   
extern NSString *const OOOoyalaPlayerErrorNotification ;  
/** Fires when an error occurs */  
.   
.   
.
```

After an error notification is received, to determine the exact error, read the `OOOoyalaError` property of the player object:

```
/** Returns current error if it exists */  
@property(readonly, nonatomic, strong) OOOoyalaError *error;
```

The errors are enumerated in the header file `Headers/OOOoyalaError.h` with explanatory comments:

```
/**  
 * Error Codes  
 */  
typedef enum {  
    OOOoyalaErrorCodeAuthorizationFailed,  
    /** Authorization Failed */  
    OOOoyalaErrorCodeAuthorizationInvalid,  
    /** Authorization Response invalid */  
    OOOoyalaErrorCodeHeartbeatFailed,  
    /** Heartbeat failed */  
    OOOoyalaErrorCodeContentTreeInvalid,  
    /** Content Tree Response invalid */  
    OOOoyalaErrorCodeAuthorizationSignatureInvalid,  
    /** The signature of the Authorization Response is invalid */  
    OOOoyalaErrorCodeContentTreeNextFailed,  
    /** Content Tree Next failed */  
    OOOoyalaErrorCodePlaybackFailed,  
    /** AVPlayer Failed */  
    OOOoyalaErrorCodeAssetNotEncodedForIOS,  
    /** The asset is not encoded for iOS */  
    OOOoyalaErrorCodeInternalIOS,  
    /** An Internal iOS Error. Check the error property. */  
    OOOoyalaErrorCodeMetadataInvalid,  
    /** Metadata Response invalid */  
} OOOoyalaErrorCode;
```





## AUTHORIZATION ERROR CODES

You can examine the field `authCode` on your player object's current item (`player.currentItem.authCode`) for possible errors, as shown below.

Error	Code
AUTHORIZED	0
UNAUTHORIZED_PARENT	1
UNAUTHORIZED_DOMAIN	2
UNAUTHORIZED_LOCATION	3
BEFORE_FLIGHT_TIME	4
AFTER_FLIGHT_TIME	5
OUTSIDE_RECURRING_FLIGHT_TIMES	6
BAD_EMBED_CODE	7
INVALID_SIGNATURE	8
MISSING_PARAMS	9
MISSING_RULE_SET	10
UNAUTHORIZED	11
MISSING_PCODE	12
UNAUTHORIZED_DEVICE	13
INVALID_TOKEN	14
TOKEN_EXPIRED	15
UNAUTHORIZED_MULTI_SYND_GROUP	16
PROVIDER_DELETED	17
TOO_MANY_ACTIVE_STREAMS	18
MISSING_ACCOUNT_ID	19
NO_ENTITLEMENTS_FOUND	20
NON_ENTITLED_DEVICE	21
NON_REGISTERED_DEVICE	22



# WORKING WITH CLOSED CAPTIONS

---

With the Mobile SDK, there are many ways you can work with closed captions, from the most basic features that need no programming to more advanced programming features.

## REQUIRED CLOSED CAPTION SETUP IN BACKLOT

---

A setup in Backlot is prerequisite to closed captions (CC) on mobile devices.

A prerequisite to displaying language-specific closed captions for any video on mobile devices is that, for each video, you must first setup a [Distribution Exchange Format Profile \(DFXP\)](#) file that includes the captions in the desired languages. This file must be uploaded to Backlot and associated with the video itself.

No programming with the Ooyala Mobile SDK is required for this most basic setup.

### DFXP FORMAT

Multiple languages can be included in a single DFXP file; for details on format, see [DFXP format](#).

### UPLOADING

After you have prepared the DFXP file, it can be uploaded to Backlot either with the Backlot UI, as described in [Uploading DFXP data](#), or with the Backlot API `/v2/assets` route and the `/closed_captions` [qualifier](#).

### EFFECT ON MOBILE DEVICES

This most basic prerequisite setup displays a button with which the viewer can select the desired closed caption language.

## FUNDAMENTAL CLOSED CAPTION FUNCTIONS ON IOS

---

A simple programming example shows the basic functions on iOS to get and set the closed caption (CC) language.

**Note:** Be sure you have setup the required DFXP files for all your videos, as detailed [Required Closed Caption Setup in Backlot](#) on page 18. This simple example shows some of the basic function calls for programming with closed captions on iOS:

- Shown first here is a check for the `OOOoyalaPlayerCurrentItemChangedNotification` event, which in this context indicates that the video and its associated DFXP file have been loaded. (Event programming is not required for working with closed captions, but is shown here only as a useful feature. If you are interested in more details about event programming, see [Working with Events](#) on page 15.)
- To find out what languages are available for the video, use the `availableClosedCaptionsLanguages` function, which returns an array named `availableLanguages` in this example.



- Then, with the `setClosedCaptionsLanguage` function you can actually set the desired language. This example checks if English ("en") is available and sets the captions to that language.

```
- (void) notificationReceived:(NSNotification*)notification {
    if ([notification.name
        isEqualToString:OOOyalaPlayerCurrentItemChangedNotification]) {
        NSArray* availableLanaguegs =
        [ooyalaPlayerViewController.player availableClosedCaptionsLanguages];
        for (NSString *language in availableLanaguegs) {
            if ([language isEqualToString:@"en"]) {

                [ooyalaPlayerViewController.player setClosedCaptionsLanguage:language];
            }
        }
        return;
    }
}
```

## IOS FCC 708 CLOSED CAPTIONS COMPLIANCE

---

iOS FCC 708 compliance has been applied using device level Accessibility settings. You need to enable Closed Caption (CC) settings on the device itself for it to be rendered in preferred styles on the video app. This one time setting, applies to any Ooyala SDK app. For more information, please see the [Apple user guide](#). Caveats: Supported on iOS 7.x+ only

**Note:** Ooyala FCC 708 compliance is only supported on iOS 7.x+ versions. HTML5 and Hook are not currently supported.

To comply with FCC Regulation, we require the inclusion of `QuartzCore.framework` and `MediaAccessibility.framework`. You may mark `MediaAccessibilty.framework` as optional. For more information, see [Getting Started: Setting Up Your Environment](#).



# LOCALIZING THE USER INTERFACE

---

You can set up custom localization strings for the UI text.

The Mobile SDK uses the Apple-provided, default iOS Localization mechanism. You can add a file named `Localizable.strings` to the SDK package to translate all UI text to whatever language you want.

Follow the guidelines from Apple at [Internationalize Your App](#). The information and steps that you need are under the headings **Configure a Localizable.strings File**, subheading **To add a Localizable.strings file for each localization**.

Add the following key/value pairs to your `Localizable.strings` file. You can then customize each value as you like:

```
"OO_LIVE" = "LIVE";
"OO_SUBTITLES_TITLE" = "Subtitles";
"OO_CLOSED_CAPTIONS_TITLE" = "Closed Captions";
"OO_SUBTITLES_NONE" = "None";
"OO_SUBTITLES_CC" = "Use Closed Captions";
"OO_SUBTITLES_EN" = "English";
```

For each desired language, specify the two-character ISO 639-1 language code on the `OO_SUBTITLES_` key. For instance, the following is for Spanish: `OO_SUBTITLES_ES`.



# INTEGRATION WITH FREEWHEEL ON IOS

With the Ooyala SDK for integrating Freewheel, the same advertising experience you have created on the desktop can be created on mobile devices when supported.

Some key design principles behind Ooyala's SDK for Freewheel include:

- Ooyala's Freewheel Manager object, which incorporates in it all of the necessary functions, so that you do not need to be concerned with working with a wide variety of calls.
- You can continue to use the basic functions of Ooyala's Mobile SDK, on which Ooyala Freewheel SDK is built.

## WHAT YOU NEED

To get started with Ooyala's SDK for FreeWheel on iOS, download the following:

1. The [Ooyala Mobile SDK for iOS](#)
2. The [Ooyala Freewheel SDK for iOS](#) (Ooyala FreeWheel iOS)

**Note:** You must have static ad tags associated with your video assets in Ooyala Backlot, on which the Mobile SDK relies. If these are not present in your production Backlot account, you must load them in your app.

3. The [AdManager.framework](#) from FreeWheel website (you need FreeWheel credentials to download the zip)

## STRUCTURE OF THE OOOYALA FREEWHEEL SDK

The SDK has the following directories and files.

Directory/File	Description
Documentation	Reference docs for the OoyalaFreewheelManager
FreewheelHeaders	Header files for the Ooyala-FreeWheel integration.
FreewheelSampleApp	A sample application implementing the Ooyala SDK
libOoyalaFreewheelSDK.a	The Ooyala SDK library file you need to add to any new app
VERSION	Version number of the Ooyala SDK for FreeWheel

## ESSENTIAL PARAMETERS AND FREEWHEEL OPF MODULE AD SET

To make use of FreeWheel in the Mobile SDK, you must create an ad set of type **FreeWheel OPF Module** in Backlot. See the Backlot User Guide for details. Ooyala allows you to store Freewheel-ad-related parameters in a variety of locations. In order of precedence, FreeWheel parameters and their values can be defined in:

1. Your app itself.
2. Internal Ooyala configuration, which you can set by way of your Customer Success Manager.
3. In Backlot's **MONETIZE** tab, **Ad Sets** subtab for the **FreeWheel OPF Module** type of ad set.

The sample app included with the FreeWheel intergration for Ooyala's Mobile SDK includes the following lines, which show two essential parameters that must be set in your app (the other parameters are commented).



**Note:** These parameters must be set either directly in your app or internal-to-Ooyala by your Customer Success Manager:

- fw\_ios\_ad\_server: The URL for serving ads on iOS
- fw\_ios\_player\_profile: The defined profile for the player on iOS

```
//Set Freewheel parameters. Note that these are optional, and override
configurations set in Backlot or in Ooyala internals
[fwParameters setObject:@"90750" forKey:@"fw_ios_mrm_network_id"];
[fwParameters setObject:@"http://demo.v.fwmrm.net/"
forKey:@"fw_ios_ad_server"];
[fwParameters setObject:@"90750:ooyala_ios"
forKey:@"fw_ios_player_profile"];
[fwParameters setObject:@"ooyala_test_site_section"
forKey:@"fw_ios_site_section_id"];
[fwParameters setObject:@"ooyala_test_video_with_bvi_cuepoints"
forKey:@"fw_ios_video_asset_id"];
[fwParameters
setObject:@"channel=TEST;subchannel=TEST;section=TEST;mode=online;player=ooyala;beta=n"
forKey:@"FRMSegment"];
```

## SEE THE FREEWHEEL SAMPLE APP IN ACTION ON IOS

To get started, follow these steps to run the sample app Ooyala provides.

1. Download the [\[HOLD need real link\] Ooyala Google Freewheel SDK for iOS](#).
2. Unzip the OoyalaFreewheelSDK-iOS.zip file.
3. Go to the subdirectory OoyalaFreewheelSDK-iOS/FreewheelSampleApp.
4. Open the file FreewheelSampleApp.xcodeproj. This starts Xcode.
5. In Xcode, open the folder FreewheelSampleApp.
6. Edit the ViewController.m file, as shown below.

```
//
//  ViewController.m
//  Freewheel Sample App
//
//  Created by Sumin Kang on 11/27/13.
//  Copyright (c) 2013 Ooyala, Inc. All rights reserved.
//
//  This Sample App demonstrates how to integrate Freewheel Ad Manager
//  with the Ooyala player
//

#import "ViewController.h"
#import "OOOoyalaPlayerViewController.h"
#import "OOOoyalaPlayer.h"
#import "OOFreewheelManager.h"

@interface ViewController ()
@property (nonatomic, strong) OOOoyalaPlayerViewController
    *ooyalaPlayerViewController;
@property (nonatomic, strong) OOFreewheelManager *fwAdManager;
@end

@implementation ViewController

@synthesize ooyalaPlayerViewController;
```



```

NSString *const EMBEDCODE      = @"VmMnA4ZzoSrZEiptnTUbroi7BqpNqotP";
NSString *const PCODE          = @"BidTxOqebpNklrVsjs2sUJSTOZc";
NSString *const PLAYERDOMAIN   = @"http://www.ooyala.com";

- (void)viewDidLoad {
    [super viewDidLoad];
    ooyalaPlayerViewController = [[OOOoyalaPlayerViewController alloc]
    initWithPcode:PCODE domain:PLAYERDOMAIN];

    //Setup video view
    [ooyalaPlayerViewController.view setFrame:self.videoView.bounds];
    [self addChildViewController:ooyalaPlayerViewController];
    [self.videoView addSubview:ooyalaPlayerViewController.view];

    //Setup Freewheel
    self.fwAdManager = [[OOFreewheelManager alloc]
    initWithOoyalaPlayerViewController:ooyalaPlayerViewController];
    NSMutableDictionary *fwParameters = [[NSMutableDictionary alloc] init];

    //Set Freewheel parameters. Note that these are optional, and override
    configurations set in Backlot or in Ooyala internals
    [fwParameters setObject:@"90750" forKey:@"fw_ios_mrm_network_id"];
    [fwParameters setObject:@"http://demo.v.fwmrm.net/"
    forKey:@"fw_ios_ad_server"];
    [fwParameters setObject:@"90750:ooyala_ios"
    forKey:@"fw_ios_player_profile"];
    [fwParameters setObject:@"ooyala_test_site_section"
    forKey:@"fw_ios_site_section_id"];
    [fwParameters setObject:@"ooyala_test_video_with_bvi_cuepoints"
    forKey:@"fw_ios_video_asset_id"];

    [self.fwAdManager overrideFreewheelParameters:fwParameters];

    [ooyalaPlayerViewController.player setEmbedCode:EMBEDCODE];
    [ooyalaPlayerViewController.player play];
}

- (void)didReceiveMemoryWarning {
    [super didReceiveMemoryWarning];
    //Dispose of any resources that can be recreated
}

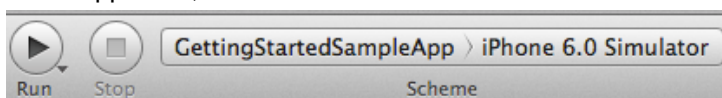
@end

```

- EMBEDCODE: The identifier for one of your video assets that has associated Freewheel ads.
- PCODE: Your Ooyala-supplied provider ID code, displayed in your Backlot account on the **ACCOUNTS** tab, **Developer** subtab.
- PLAYERDOMAIN: Leave unchanged as `www.ooyala.com`. This constant works in conjunction with **Syndication Controls** (publishing rules) in Backlot. If you have set Internet domain restrictions on videos in Backlot (see the *Backlot User Guide*), the constant here can be set to one of those allowed domains. If you have not set these **Syndication Controls**, the constant has no effect.

7. Save your changes.

8. In the upper left, select **iPhone 6.0 Simulator**.



9. In the upper left, click **Run**.

The iPhone Simulator appears and your video is loaded. Click the play button to view the video.



## A CLOSER LOOK AT THE IOS SAMPLE APP AND INTEGRATION TOUCHPOINTS

---

A closer look at the source code of the sample app highlights the touchpoints you need focus on to build your own app:

- The Imports
- ViewController Implementation, with Ad Manager
- The Constants
- View Setup and OOFreewheelManager Initialization
- Set Freewheel Ad Parameters
- Play the Video

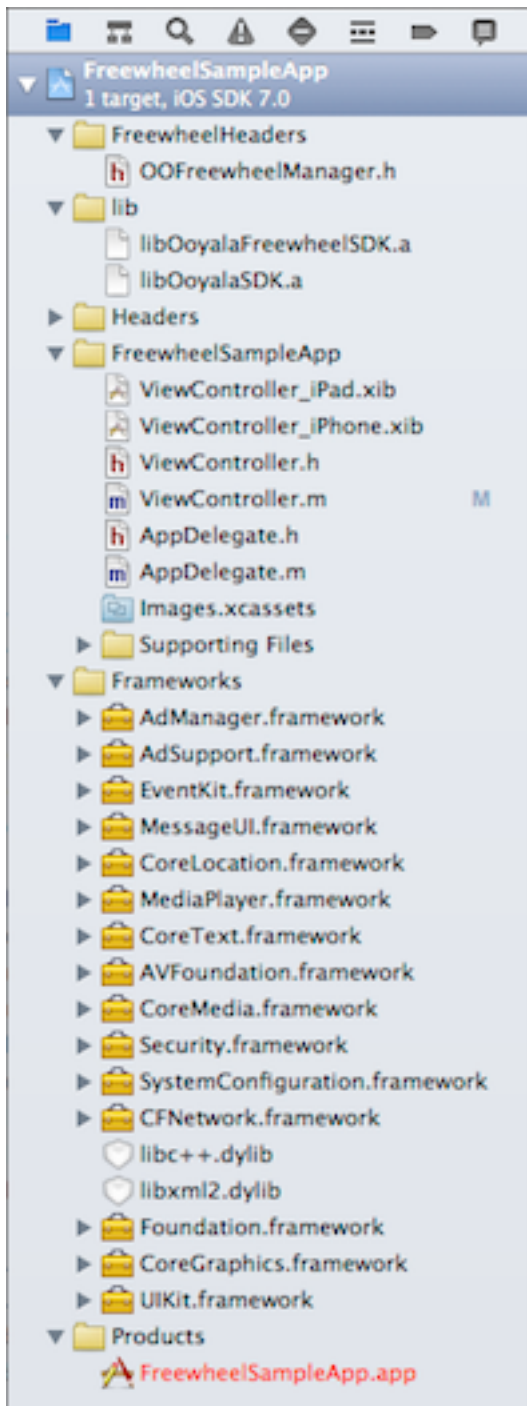
To begin, open the `FreewheelSampleApp/ViewController.m` file.

### THE HEADER AND LIBRARY FILES

For your Xcode project's **Build Phase**, be sure to include the following in the **Link Binaries With Libraries** (click **Add Other...**):







- OoyalaFreewheelSDK-iOS/  
libOoyalaFreewheel.a
- From the baseline Ooyala Mobile SDK, the libOoyalaSDK.a library. (For your convenience, this is also included in the Freewheel sample app directory.)
- The OoyalaFreewheelSDK-iOS/  
FreewheelLibraries/AdManager.framework folder
- The following frameworks (unless you already have them in your project). To add these, select them from the list of Xcode frameworks:
  - libxml2.dylib
  - UIKit.framework
  - CoreGraphics.framework
  - QuartzCore.framework
  - MediaPlayer.framework
  - CoreLocation.framework
  - MessageUI.framework
  - EventKit.framework
  - CoreMedia.framework
  - AVFoundation.framework
  - AdSupport.framework (optional)

To your Xcode project itself, add Ooyala's the OoyalaFreewheelSDK-iOS/FreewheelHeaders folder.

## THE IMPORTS

The first three import statements pull in header definitions from the baseline Ooyala Mobile SDK.

The fourth import pulls in the Ooyala Freewheel Manager definitions.

```
#import "ViewController.h"
#import "OOOoyalaPlayerViewController.h"
```



```
#import "OOOoyalaPlayer.h"
#import "OOFreewheelManager.h"
.
.
.
```

## VIEWCONTROLLER IMPLEMENTATION, WITH AD MANAGER

These lines define the ViewController implementation. Notice the definition of the `fwAdManager` property as an `OOFreewheelManager`-type interface.

```
.
.
.
@interface ViewController ()
@property (nonatomic, strong) OOOoyalaPlayerViewController
    *ooyalaPlayerViewController;
@property (nonatomic, strong) OOFreewheelManager *fwAdManager;
@end

@implementation ViewController

@synthesize ooyalaPlayerViewController;
.
.
.
```

For reference documentation about the `OOFreewheelManager`, see the `OoyalaFreewheelSDK-iOS/Documentation` subdirectory.

## THE CONSTANTS

Whereas the sample app defines constants `PCODE`, `EMBEDCODE`, and `PLAYERDOMAIN` (see [See the Freewheel Sample App in Action on iOS](#) on page 22), you probably want to define variables, especially for the `EMBEDCODE` constant (asset ID or content ID).

## VIEW SETUP AND OOFREEWHEELMANAGER INITIALIZATION

In the sample app, the view is setup and the `fwAdManager` object is initialized as an `OOFreewheelManager`:

```
.
.
.
- (void)viewDidLoad {
    [super viewDidLoad];
    ooyalaPlayerViewController = [[[OOOoyalaPlayerViewController
        alloc] initWithPcode:PCODE domain:[OOPlayerDomain alloc]
        initWithString:PLAYERDOMAIN];

    //Setup video view
    [ooyalaPlayerViewController.view setFrame:self.videoView.bounds];
    [self addChildViewController:ooyalaPlayerViewController];
    [self.videoView addSubview:ooyalaPlayerViewController.view];

    //Setup Freewheel
    self.fwAdManager = [[OOFreewheelManager alloc]
        initWithOoyalaPlayerViewController:ooyalaPlayerViewController];
.
.
```



```
.  
.
```

## SET FREEWHEEL AD PARAMETERS

To set the ad parameters you need, change the values of the parameters included in the app. First, after the `OoyalaPlayerView` controller has been initialized (in the previous step) but before attempting to play the video, create an `NSMutableDictionary` named `fwParameters`; then set the desired parameter values to your own settings (replacing the sample values shown below).

The parameters highlighted below must be set in the app itself; the others can be set as explained in [Essential Parameters and FreeWheel OPF Module Ad Set](#) on page 21.

```
.  
.  
.  
    //Set Freewheel parameters. Note that these are optional, and override  
    configurations set in Backlot or in Ooyala internals  
    NSMutableDictionary *fwParameters = [[NSMutableDictionary alloc] init];  
  
    [fwParameters setObject:@"90750" forKey:@"fw_ios_mrm_network_id"];  
    [fwParameters setObject:@"http://demo.v.fwmrm.net/"  
    forKey:@"fw_ios_ad_server"];  
    [fwParameters setObject:@"90750:ooyala_ios"  
    forKey:@"fw_ios_player_profile"];  
    [fwParameters setObject:@"ooyala_test_site_section"  
    forKey:@"fw_ios_site_section_id"];  
    [fwParameters setObject:@"ooyala_test_video_with_bvi_cuepoints"  
    forKey:@"fw_ios_video_asset_id"];  
    [fwParameters  
    setObject:@"channel=TEST;subchannel=TEST;section=TEST;mode=online;player=ooyala;beta=n"  
    forKey:@"FRMSegment"];  
  
    [self.fwAdManager overrideFreewheelParameters:fwParameters];  
.  
.  
.
```

## PLAYING THE VIDEO

Finally, on the following lines, the `setEmbedCode` method is from the baseline Ooyala Mobile SDK. These lines invokes the Ooyala player with the desired video identifier. This is another touchpoint where your own app probably needs to use a variable for the identifier (rather than the `EMBEDCODE` constant).

```
.  
.  
.  
    [ooyalaPlayerViewController.player setEmbedCode:EMBEDCODE];  
    [ooyalaPlayerViewController.player play];  
.  
.  
.
```



# INTEGRATION WITH GOOGLE IMA ON IOS

---

With the Ooyala SDK for integrating Google IMA (Interactive Media Ads), the same advertising experience you have created on the desktop can be created on mobile devices.

**Note:** This software and documentation is for Google IMA with Ooyala's Mobile SDK for use on mobile devices.

Some key design principles behind Ooyala's SDK for Google IMA include:

- Ooyala's IMA Manager object, incorporates all of the necessary functions of the Google IMA SDK, so that you do not need to be concerned with working directly with Google's calls.
- We built the Ooyala SDK for Google IMA on top of our mobile SDKs, so you can continue to use their basic functions.

## WHAT YOU NEED

To get started with Ooyala's Google IMA for iOS SDK integration, download the following:

1. [Ooyala Mobile SDK for iOS](#)
2. [Ooyala Google IMA SDK for iOS](#) (Ooyala Google IMA iOS)

**Note:** Our SDK contains a version of the Google IMA SDK so you do not need to download it from Google.

3. [Apple Xcode](#)

## PREREQUISITE: WORKING GOOGLE IMA SETUP, WITH ASSOCIATED VIDEO ASSETS

Before you start working with the Ooyala Google IMA SDK, make sure you have a working Google IMA setup, as described at <https://developers.google.com/interactive-media-ads/>.

**Note:** You must have static ad tags associated with your video assets in Ooyala Backlot, on which the Mobile SDK relies. If these are not present in your production Backlot account, you must load them in your app.

If Google IMA is already working correctly on your desktop, you should have no difficulty getting it to work on mobile devices.

## DIRECTORIES AND FILES IN THE DISTRIBUTION

The following folders and files are included in the distribution.

Folder/File Name	
Documentation	Reference documentation for the distribution
GoogleIMALibraries	Library ( . a ) files from Google: <ul style="list-style-type: none"><li>• <code>libGoogleIMA3.a</code>, for normal use</li><li>• <code>libGoogleIMA3ForAdMob.a</code>, for use with Google IMA for AdMob</li></ul>
IMAHeaders	Program headers you need to add to any new Xcode project to work with the Ooyala Google IMA SDK
IMASampleApp	A sample app using the distribution, described below.
<code>libOoyalaIMA.a</code>	The compiled Ooyala Google IMA SDK library you need to add to any new Xcode project



## Folder/File Name

VERSION                      Version number of the distribution

## SEE THE IMA SAMPLE APP IN ACTION ON IOS

To get started, follow these steps to run the sample app Ooyala provides.

1. Download the [Ooyala Google IMA SDK for iOS](#).
2. Unzip the OoyalaIMASDK-iOS.zip file.
3. Go to the subdirectory OoyalaIMASDK-iOS/IMASampleApp.
4. Open the file GoogleIMASampleApp.xcodeproj. This starts Xcode.
5. In Xcode, open the folder GoogleIMASampleApp.
6. Edit the ViewController.m file, as shown below.

```
//
//  ViewController.m
//  GoogleIMASampleApp
//
//  Created on 7/23/13.
//  Copyright (c) 2013 Ooyala, Inc. All rights reserved.
//

#import "ViewController.h"
#import "OOOoyalaPlayerViewController.h"
#import "OOOoyalaPlayer.h"
#import "OOIMAManager.h"

@interface ViewController ()
@property (nonatomic, strong) OOOoyalaPlayerViewController
    *ooyalaPlayerViewController;
@property (nonatomic, strong) OOIMAManager *adsManager;
@end

@implementation ViewController

@synthesize ooyalaPlayerViewController;

NSString *const PCODE          = @"Vpd3E6BNabnn09G72IWye5O2RzN1";
NSString *const EMBEDCODE     = @"Rpa2liNzoiVehWQ6ARsJcwEGY5M4Ozrl";
NSString *const PLAYERDOMAIN = @"http://www.ooyala.com";

- (void)viewDidLoad
{
    [super viewDidLoad];
    ooyalaPlayerViewController = [[[OOOoyalaPlayerViewController
    alloc] initWithPcode:PCODE domain:[OOPlayerDomain alloc]
    initWithString:PLAYERDOMAIN];

    // Setup video view
    [ooyalaPlayerViewController.view setFrame:self.videoView.bounds];
    [self addChildViewController:ooyalaPlayerViewController];

    [self.videoView addSubview:ooyalaPlayerViewController.view];
    [ooyalaPlayerViewController.player setEmbedCode:EMBEDCODE];

    self.adsManager = [[OOIMAManager alloc]
    initWithOoyalaPlayerViewController:ooyalaPlayerViewController];
```



```

// Setup a companion ad view
if (UI_USER_INTERFACE_IDIOM() == UIUserInterfaceIdiomPad) {
    [self.adsManager addCompanionSlot:_largerCompanionSlot];
}
[self.adsManager addCompanionSlot:_smallerCompanionSlot];
}

- (void)didReceiveMemoryWarning
{
    [super didReceiveMemoryWarning];
    // Dispose of any resources that can be recreated.
}

@end

```

7. Focus on the lines highlighted above and replace the values of the following constants:

```

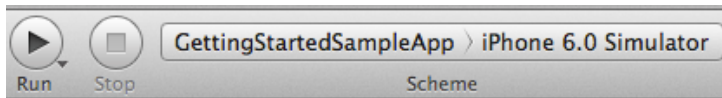
NSString *const PCODE          = @"Vpd3E6BNabnn09G72IWye5O2RzN1";
NSString *const EMBEDCODE     = @"Rpa2linZoiVehWQ6ARsJcwEGY5M4Ozrl";
NSString *const PLAYERDOMAIN = @"http://www.ooyala.com"

```

- PCODE: Your Ooyala-supplied provider ID code, displayed in your Backlot account on the **ACCOUNTS** tab, **Developer** subtab.
- EMBEDCODE: The identifier for one of your video assets that has associated IMA ads.
- PLAYERDOMAIN: Leave unchanged as "**http://www.ooyala.com**". This constant works in conjunction with **Syndication Controls** (publishing rules) in Backlot. If you have set Internet domain restrictions on videos in Backlot (see the *Backlot User Guide*), the constant here can be set to one of those allowed domains. If you have not set these **Syndication Controls**, the constant has no effect.

8. Save your changes.

9. In the upper left, select **iPhone 6.0 Simulator**.



10. In the upper left, click **Run**.

The iPhone Simulator appears and your video is loaded. Click the play button to view the video.

## A CLOSER LOOK AT THE IOS SAMPLE APP AND INTEGRATION TOUCHPOINTS

A closer look at the source code of the sample app highlights the touchpoints you need focus on to build your own app.:

- The Header and Library Files
- The Imports
- ViewController Implementation, with Ads Manager
- The Constants
- View Set Up and OOIMAManager Initialization
- Append Ad Tag Parameters to Ad Tag URL
- Companion Ad Slot
- Playing the Video

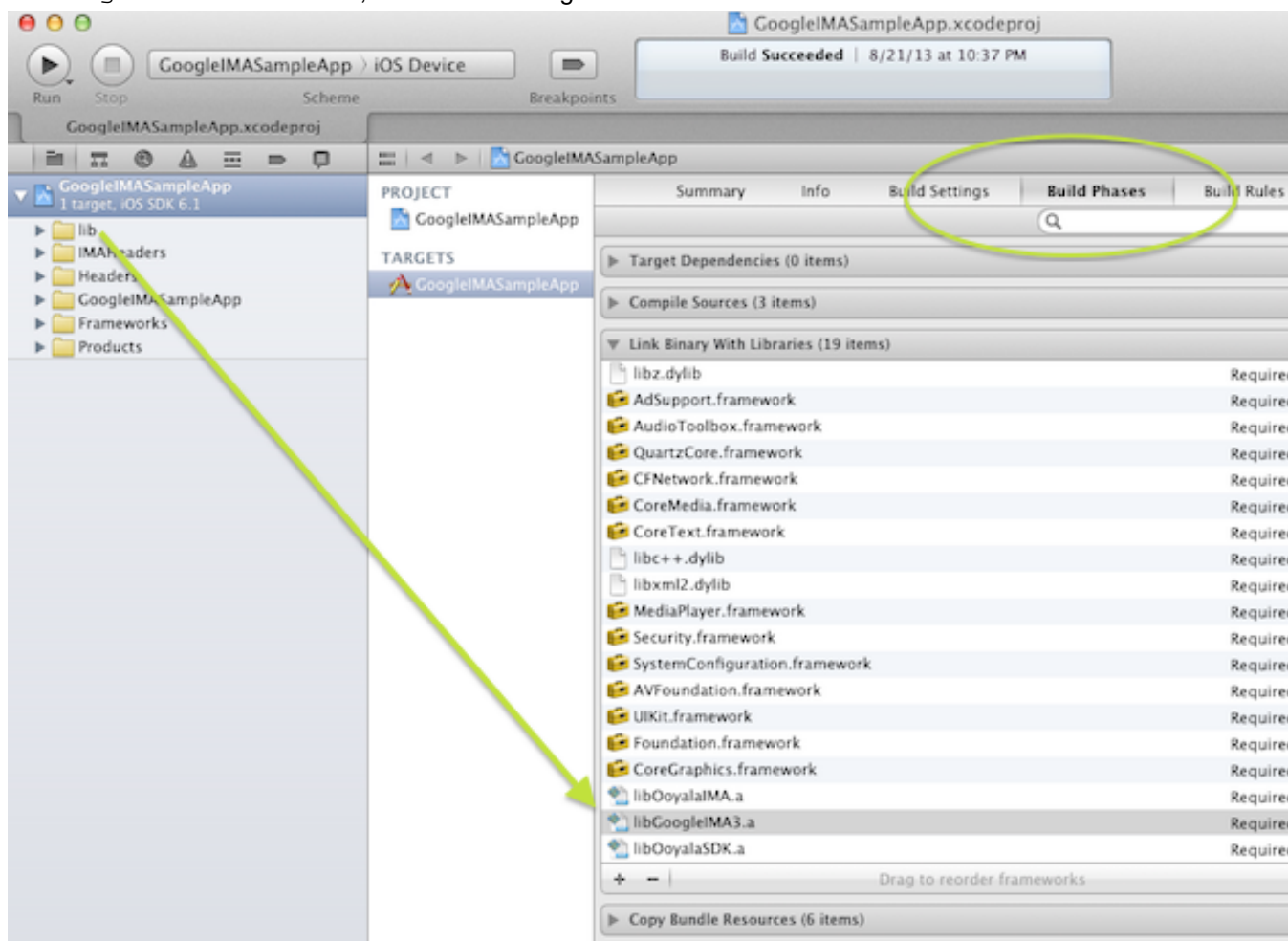


To begin, open the `GoogleIMASampleApp/ViewController.m` file.

## THE HEADER AND LIBRARY FILES

For your Xcode project's Build Phase, be sure to include the following libraries, as shown in the graphic below:

- `OoyalaIMASDK-iOS/libOoyalaIMA.a`
- From the baseline Ooyala Mobile SDK, the `libOoyalaSDK.a` library. (For your convenience, this is also included in the IMA sample app directory.)
- Either one of the following Google libraries from the `GoogleIMALibraries` directory, depending on if you are implementing Google IMA for Ad Mob or not:
  - `libGoogleIMA3.a`, for normal use. (The Ooyala-provided sample app uses this library.)
  - `libGoogleIMA3ForAdMob.a`, for use with Google IMA for AdMob



For any new project to implement the Ooyala Google IMA SDK, you need to include all the header files from the `OoyalaIMASDK-iOS/IMAHeaders` subdirectory.

## THE IMPORTS

The first three import statements pull in header definitions from the baseline Ooyala Mobile SDK.

The fourth import pulls in the Ooyala IMA Manager definitions.



```
#import "ViewController.h"
#import "OOOoyalaPlayerViewController.h"
#import "OOOoyalaPlayer.h"
#import "OOIMAManager.h"
```

## VIEWCONTROLLER IMPLEMENTATION, WITH ADS MANAGER

These lines define the ViewController implementation. Notice the definition of the `adsManager` property as an `OOIMAManager`-type interface.

```
@interface ViewController ()
@property (nonatomic, strong) OOOoyalaPlayerViewController
    *ooyalaPlayerViewController;
@property (nonatomic, strong) OOIMAManager *adsManager;
@end

@implementation ViewController

@synthesize ooyalaPlayerViewController;
```

For reference documentation about the `OOIMAManager`, see the `OoyalaIMASDK-iOS/Documentation` subdirectory.

## THE CONSTANTS

Whereas the sample app defines constants `PCODE`, `EMBEDCODE`, and `PLAYERDOMAIN` (see [See the IMA Sample App in Action on iOS](#) on page 29), you probably want to define variables, especially for the `EMBEDCODE` constant (asset ID or content ID).

## VIEW SET UP AND OOIMAMANAGER INITIALIZATION

Various view controllers combined with the `OOIMAManager` are included with the SDK. You can initialize the view with or without the Ooyala player. These and other definitions are in the header file `IMAHeaders/OOIMAManager.h`.

In the sample app, the view is setup and the `adsManager` object is initialized as an `OOIMAManager`:

```
- (void)viewDidLoad
{
    [super viewDidLoad];
    ooyalaPlayerViewController = [[OOOoyalaPlayerViewController
    alloc] initWithPcode:PCODE domain:[[OOPlayerDomain alloc]
        initWithString:PLAYERDOMAIN];

    [self.videoView addSubview:ooyalaPlayerViewController.view];

    //Create the IMA Ad Manager
    self.adsManager = [[OOIMAManager alloc]
    initWithOoyalaPlayer:ooyalaPlayerViewController.player];
```





## COMPANION AD SLOT

A companion ad slot is defined for the `adsManager` object.

```
// Setup a companion ad view
if (UI_USER_INTERFACE_IDIOM() == UIUserInterfaceIdiomPad) {
    [self.adsManager addCompanionSlot:_largerCompanionSlot];
}
[self.adsManager addCompanionSlot:_smallerCompanionSlot];
```

## APPEND AD TAG PARAMETERS TO AD TAG URL

To append your own ad tags to your Google-supplied ad tag URL, you can use the `setAdTagParameters()` method.

1. First, create an `NSMutableDictionary` named `adTagParameters`.

```
.
.
.
    //Create adTagParameters dict, fill it, and give it to the IMA Ad
    Manager
    NSMutableDictionary *adTagParameters = [NSMutableDictionary dictionary];
    [adTagParameters setObject: EMBEDCODE forKey:@"vid" ];
    [adTagParameters setObject: @"[referrer_url]" forKey: @"url" ];
    [adTagParameters setObject: @"2" forKey: @"pod"];
    [adTagParameters setObject: @"2" forKey: @"ppos"];
    [adTagParameters setObject: @"preroll" forKey:@"vpos"];
    [adTagParameters setObject: @"2" forKey: @"mridx"];

    [self.adsManager setAdTagParameters: adTagParameters];
.
.
.
```

2. Then for each of the ad tags you want to append, use `setObject` to define its value and `forKey` to define its name. Notice in the sample app that the ad tag `vid` is set to the `EMBEDCODE` constant (the video ID).
3. Finally, after you have populated the dictionary, pass the `adTagParameters` dictionary to the `setAdTagParameters` function of the `adsManager` object (of type `OOIMAManager`).

As an example, suppose you have an `adTagUrl` that looks like the following (this URL is broken across several lines for readability):

```
http://pubads.g.doubleclick.net/gampad/ads?sz=400x300&iu=
%2F6062%2Fhanna_MA_group%2Fwrapper_with_comp
&ciu_szs=728x90&impl=s&gdfp_req=1&env=vp&output=xml_vast2&unviewed_position_start=1
&m_ast=vast&correlator=[timestamp]
```

After setting up your ad tag parameters as shown in the sample app, the ad tag URL would look like this, with the extra tags highlighted:

```
http://pubads.g.doubleclick.net/gampad/ads?sz=400x300&iu=
%2F6062%2Fhanna_MA_group%2Fwrapper_with_comp
```



```
&ciu_szs=728x90&impl=s&gdfp_req=1&env=vp&output=xml_vast2&unviewed_position_start=1
&m_ast=vast&correlator=[timestamp]
&mridx=2&vpos=preroll&ppos=2&vid=h5OWFoYTrG4YIPdrDKrIz5-VhobsuT-
M&pod=2&url=[referrer_url]
```

You can also override the base value of the ad tag URL itself. See the discussion in [Overriding the Ad Tag URL on iOS](#) on page 34.

### PLAYING THE VIDEO

Finally, on the following line, the `setEmbedCode()` method is from the baseline Ooyala Mobile SDK. This line invokes the Ooyala player with the desired video identifier. This is another touchpoint where your own app probably needs to use a variable for the identifier (rather than the `EMBEDCODE` constant).

```
[ooyalaPlayerViewController.player setEmbedCode:EMBEDCODE];
```

## OVERRIDING THE AD TAG URL ON IOS

If you need a value for your base Google IMA ad tag URL that is different than the value defined in Backlot, use the `adUrlOverride` function, as shown in the following example. Consider:

- You must override *before* you set the video identifier with `setEmbedCode`.
- The example below does not show a complete, valid ad tag URL value.
- After the special case URL, be sure to nullify the override, as shown below.
- More details about `adUrlOverride` are in the reference documentation in the SDK package.

```
.
.
.
// To override the ad URL, do this *before* setting the embed code:
_adsManager.adUrlOverride = @"http://pubads.g.doubleclick.net/gampad/ads?
sz=640x480...";
.
. set the embed code
.
// Remember to 'nil' it out later if need be: As long as it is non-null, the
value will override any IMA settings from Backlot.
_adsManager.adUrlOverride = nil;
.
.
.
```



# INTEGRATION WITH OMNITURE ON IOS

---

The Ooyala Omniture Integration App demonstrates how you can integrate Omniture analytics capabilities into your iOS SDK-based apps. Omniture analytics, now called The Adobe® Marketing Cloud Mobile libraries after Adobe's acquisition of Omniture, allows you to capture native app activity (user, usage, behavior, gestures, etc.) and send that information to Adobe servers for ingestion and use by SiteCatalyst® reporting. You can integrate Ooyala's Mobile SDK with Omniture SDKs through a step-by-step integration process using our sample app as a model.

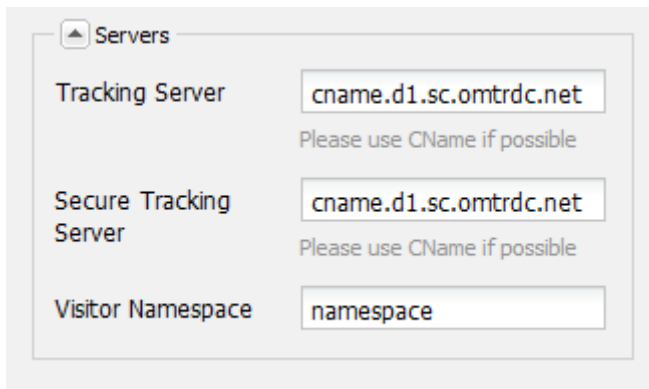
## WHAT YOU NEED

To get started with Ooyala's Omniture for iOS SDK Integration, you need to download the following items:

1. The [Ooyala Mobile SDK for iOS](#).
2. The [Ooyala Omniture Integration App for iOS](#).
3. The [Omniture SDK for iOS](#). (Omniture was purchased by Adobe and is now marketed as The Adobe® Marketing Cloud Mobile).
4. [Apple Xcode](#). Note that in our guide, we use Apple Xcode to illustrate the integration steps.

As part of its capacity to capture analytics, Omniture draws from and provides information to the Adobe SiteCatalyst website. Before starting, you will also need to do the following:

1. Have or get an account with login credentials for Adobe's SiteCatalyst.
2. Login to [SiteCatalyst](#).
3. Get the following information:
  - Report Suite ID
  - Tracking Server. The following image shows sample tracking server information from SiteCatalyst.

A screenshot of a web-based configuration window titled "Servers". It contains three rows of input fields. The first row is labeled "Tracking Server" and contains the text "cname.d1.sc.omtrdc.net" with a note below it saying "Please use CName if possible". The second row is labeled "Secure Tracking Server" and also contains "cname.d1.sc.omtrdc.net" with the same note. The third row is labeled "Visitor Namespace" and contains the text "namespace".

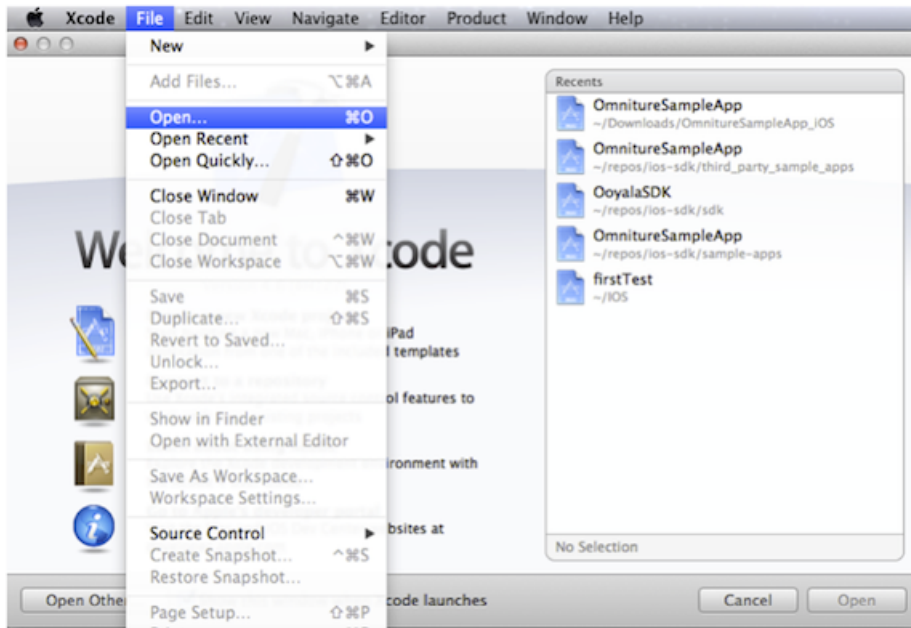
You will use this information in [Edit TrackingHelper.m](#) on page 38.

## OPEN THE IOS SAMPLE APP

To get started, all you need to do is open our sample app and integrate a few files into your project. In the following procedure, we are using the Xcode IDE. The Apple Xcode tool will help with your iOS development effort. To get started with your development project, launch Xcode.

Click **File > Open > Browse** to `OmnitureSampleApp.xcodeproj`.

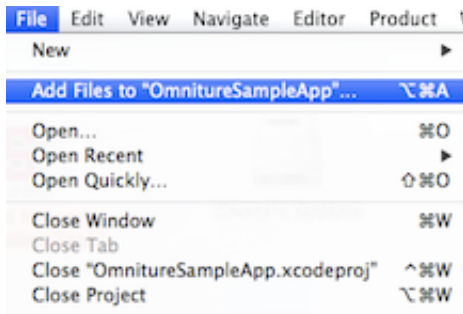




## IMPORT REQUIRED LIBRARIES

Your next step is to import some required libraries from the Omniture iOS SDK into your Omniture Integration app.

1. Click **File > Add Files to “OmnitureSampleApp”**.

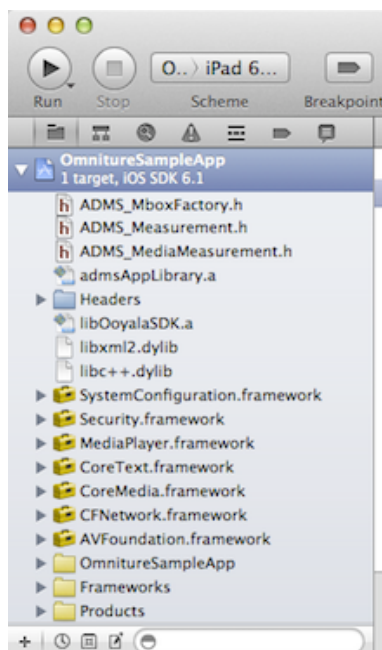
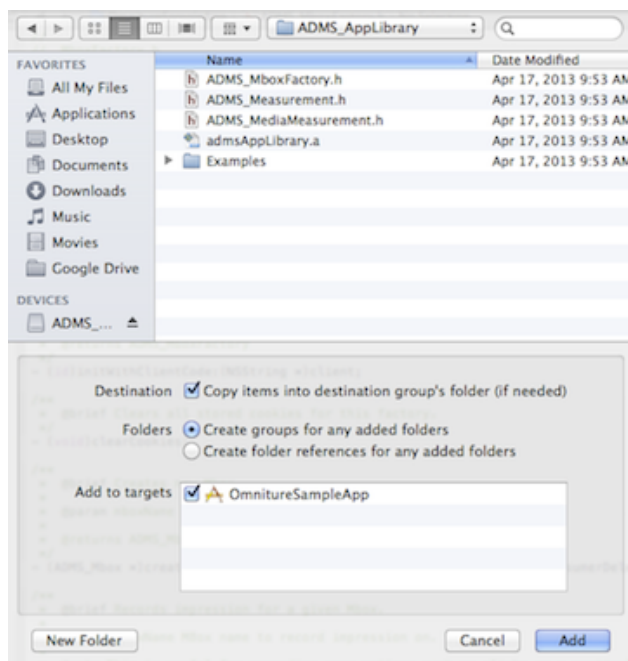


2. Add the following files from the Omniture iOS SDK (ADMS\_AppLibrary-3.X.X-iOS.dmg) to the project:

- ADMS\_Measurement.h
- ADMS\_MediaMeasurement.h
- ADMS\_MboxFactory.h (Optional)
- admsAppLibrary.a

**Note:** Because the file is a .dmg file, you might need to open it as a device, select the individual files and add the files from the device.

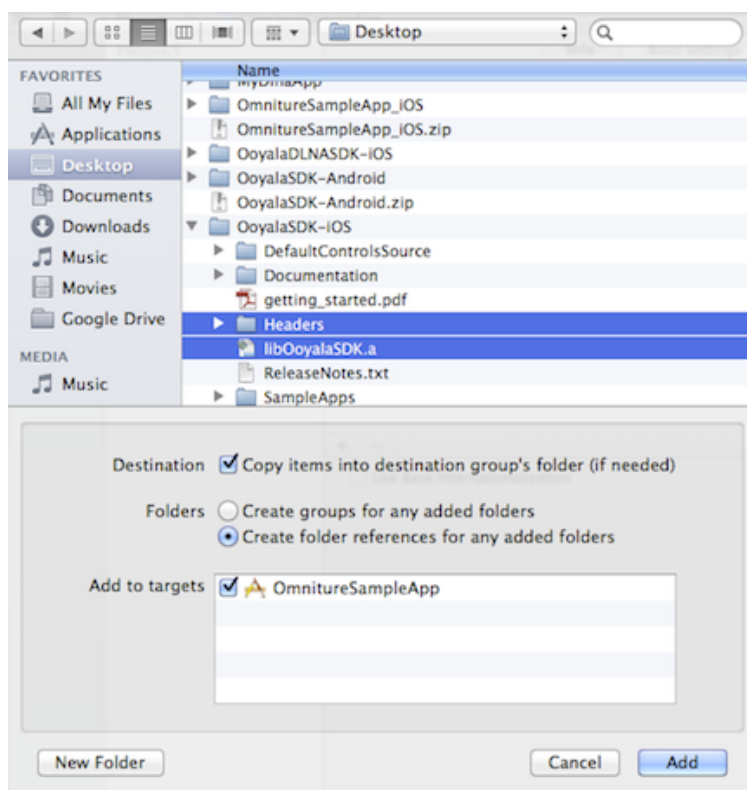




3. You also need to some files from the [Ooyala Mobile SDK for iOS](#) zipfile. As shown in the following screenshot, when you add the following files, for **Destination**, you must select **Copy items into destination group's folder**:

- libOoyala-SDK.a
- Headers (folder)





## EDIT TRACKINGHELPER.M

The next step in setting up your environment is to add some code to the ViewController. You will need to add the following lines.

1. Open the `TrackingHelper.m` file. You will make some modifications to this file with the information you saved from SiteCatalyst, as described in [What You Need](#) on page 35.

```
//
// TrackingHelper.m
// Adobe Digital Marketing Suite
//

#import "TrackingHelper.h"
#import "ADMS_Measurement.h"
#import "ADMS_MediaMeasurement.h"

NSString * const TRACKING_RSID = @"YOUR_REPORT_SUITE_ID_HERE";
NSString * const TRACKING_SERVER = @"YOUR_SERVER_HERE";

@implementation TrackingHelper
```

1. In your `TrackingHelper.m` file, change:
  - `YOUR_REPORT_SUITE_ID_HERE` to match the one you got from SiteCatalyst.
  - `YOUR_SERVER_HERE` so that it matches the one from SiteCatalyst.
2. In the `TrackingHelper.m` file, you also need to change the following configuration variables so that they match the equivalent variables in SiteCatalyst:
  - `(s.eVarN)`
  - `s.props - (s.propN)`



- `s.events – (s.events)`

```
+ (void)configureMediaMeasurement{
    ADMS_MediaMeasurement *mediaMeasurement = [ADMS_MediaMeasurement
sharedInstance];

    //need to use SiteCatalyst to map eVars and props to the following
    configs:

    // (required) configure ContextDataMapping
    mediaMeasurement.contextDataMapping = [:@{
        @"a.media.name":@"eVar29,prop29",
        @"a.media.segment":@"eVar55",
        @"a.contentType":@"eVar5",
        @"a.media.timePlayed":@"event26",
        @"a.media.view":@"event8",
        @"a.media.segmentView":@"event25",
        @"a.media.complete":@"event12"
    } mutableCopy];

    //Enable MPMoviePlayer Autotracking (iOS only)
    [mediaMeasurement
setAutoTrackingOptions:ADMS_MediaAutoTrackOptionsMPMoviePlayer];

    //configure optional settings
    mediaMeasurement.trackMilestones = @"25,50,75";
    mediaMeasurement.segmentByMilestones = YES;
}

@end
```

1. Although the sample contains an embed code and pcode, replace these with your own embed code and pcode before running the build.
2. Run the build!
3. When you run your build, successfully, the iOS Simulator is invoked.

You now have everything in place to run your build and test your app.

## BUILD YOUR PROJECT

After you have copied all the necessary components in your development environment, select **Run** to build your project. If successful, you will see Omniture analytics displayed on the SiteCatalyst web page.

## TROUBLESHOOTING

If you have any trouble with your build or build results:

1. Check the SiteCatalyst web page to see if the analytics information is displaying properly.
2. Review your logs to look for any potential issues. Logs are your friend!
3. Remember that the Omniture code is developed by Adobe. If you find an issue with that code, you need to contact your Adobe documentation or representative.



# WORKING WITH MULTI-RESOLUTION STREAMS ON WIDEVINE

---

To support ABR for video content with multiple resolutions on Google WideVine, you can set the maximum dimensions of the video stream.

To allow your multi-resolution videos to be played at maximal size on a mobile device, you can set the maximum width and height of the video stream, as shown in the code fragment below. The stream selected will have a resolution smaller or equal to the maximum dimensions you define.

1. Extend the `OODefaultPlayerInfo` class.

The `maxHeight` and `maxWidth` should accommodate the highest resolution you have.

2. Set the static `StreamPlayer.OODefaultPlayerInfo` to an instance of your new class, in the code snippet, `CustomStreamInfo`.

```
// iOS
@interface CustomStreamPlayer : OODefaultPlayerInfo
@end

@implementation CustomStreamPlayer
- (int)maxHeight { return 500; }
- (int)maxWidth { return 1280; }
@end

- (void)viewDidLoad
{
    [super viewDidLoad];

    [OOStreamPlayer setDefaultPlayerInfo:[CustomStreamPlayer alloc]];

    ...

    ooyalaPlayerViewController = [[OOOoyalaPlayerViewController alloc]
initWithPcode:PCODE domain:[[OOPlayerDomain alloc]
initWithString:PLAYERDOMAIN];

    ...
```





# CROSS-DEVICE RESUME (XDR) WITH THE MOBILE SDK

---

Cross-device Resume is the ability for a viewer to resume playback a video on a different device at the last position viewed on a previous device.

Cross-Device Resume (XDR) gives viewers the flexibility to start watching a video on one device and continue watching it on the same or different device at a later time, automatically resuming where the viewer left off. A secure server architecture is required. This architecture, REST API requests, and programmatic calls in JavaScript for Player v3, Objective C for the Ooyala Mobile SDK for iOS, and Java for the Ooyala Mobile SDK for Android are fully detailed in [Cross-device Resume \(XDR\)](#).



# XTV CONNECT SDK FOR IOS

---

To increase engagement with your content, Ooyala provides SDKs, code samples, and other tools to enable you to view your content on connected TVs.

XTV Connect SDK for iOS (formerly Mobile Connect SDK for iOS) allows a viewer to send content through a mobile application (on a smartphone or tablet, for example) directly to a connected television (which we call an *XTV*), while continuing to use the mobile device as a second screen. This is available as a plugin to our mobile SDKs. Viewers can stream content from any Android or iOS application to any supported XTV (including AppleTV).

## Requirements

To get started with the XTV Connect SDK for iOS Plug-in, you need the following items:

- An integrated development environment (IDE) such as Apple's Xcode or other similar IDE
- The [Ooyala iOS SDK](#)
- The Ooyala DLNA SDK for iOS.

**Note:** Contact your Customer Success Manager to get a copy of this SDK.

- An appropriate DLNA-enabled TV or set top box. You can also use the XBMC media center client (a DLNA renderer). You can use this tool for testing and debugging your projects. (DLNA is the Digital Living Network Alliance, hosted at [www.dlna.org](http://www.dlna.org).)

XTV Connect allows customization of the XTV apps based on additional features that customers require. Customers use the XTV Connect app, plug-in and iOS SDK to modify and include advanced functionality like subscriptions. Viewers can control the XTV app using a TV remote, or choose to do so through the second screen app.

Supported	Description
Connected TVs, set-top boxes, and Blu-ray Players	Ooyala supports the following: <ul style="list-style-type: none"><li>• Apple TV (iOS 4.3, 5.0, 6.0)</li><li>• LG (such as model: 47LM860V)</li><li>• Samsung (such as model: UE-46ES8007)</li></ul> You can also check to see if other LG and Samsung models and different types of connected TVs, set-top boxes, and Blu-ray Players, such as Toshiba, Panasonic, Roku and Xbox, support video playback via DLNA at this <a href="#">DLNA organization website</a> .
Formats	mp4
Analytics	Yes
DRM	No
Auth and entitlement	Partial. Security must be controlled by your mobile app)
App required on TV	None
Content types	VOD
Ads	Pre-Roll, Mid-Roll and Post-Roll



## GETTING STARTED

See the information appropriate to the platform:

- [Create an iOS Development Project for XTV](#) on page 43
- [Configuring an Android Project for XTV](#)

## CREATE AN IOS DEVELOPMENT PROJECT FOR XTV

You need to set up a project for iOS development.

In our example, we are using the Xcode IDE. The Apple Xcode tool will help with your iOS development effort. To get started with your development project, launch your Xcode app. **Note:** You need to be able to configure for DLNA Playback and be able to test your setup. An example is provided in Appendix A.

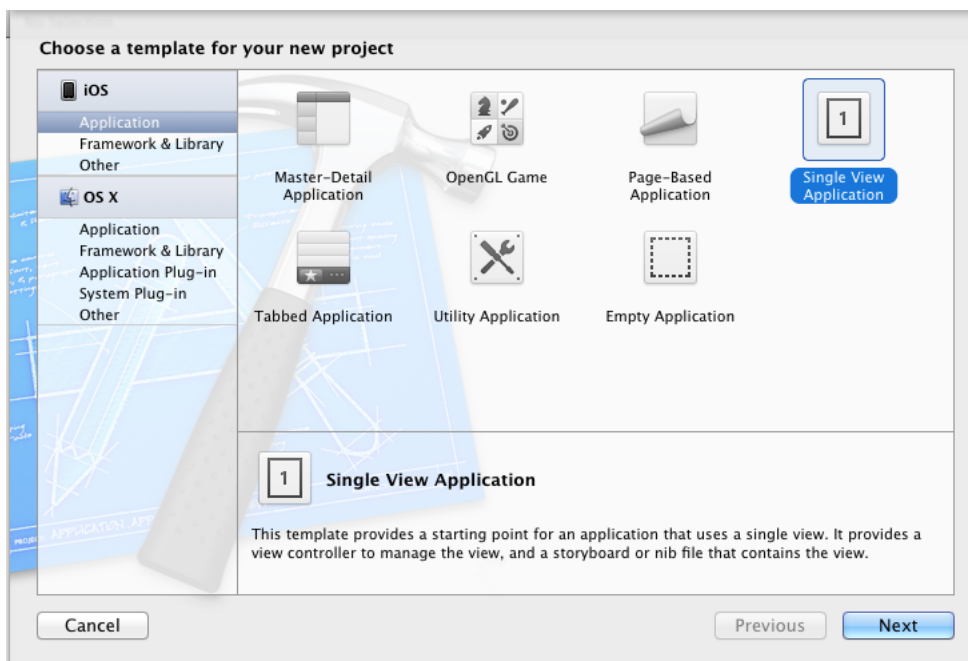
### Create a New XTV Project

1. Select **Create a new Xcode project**.



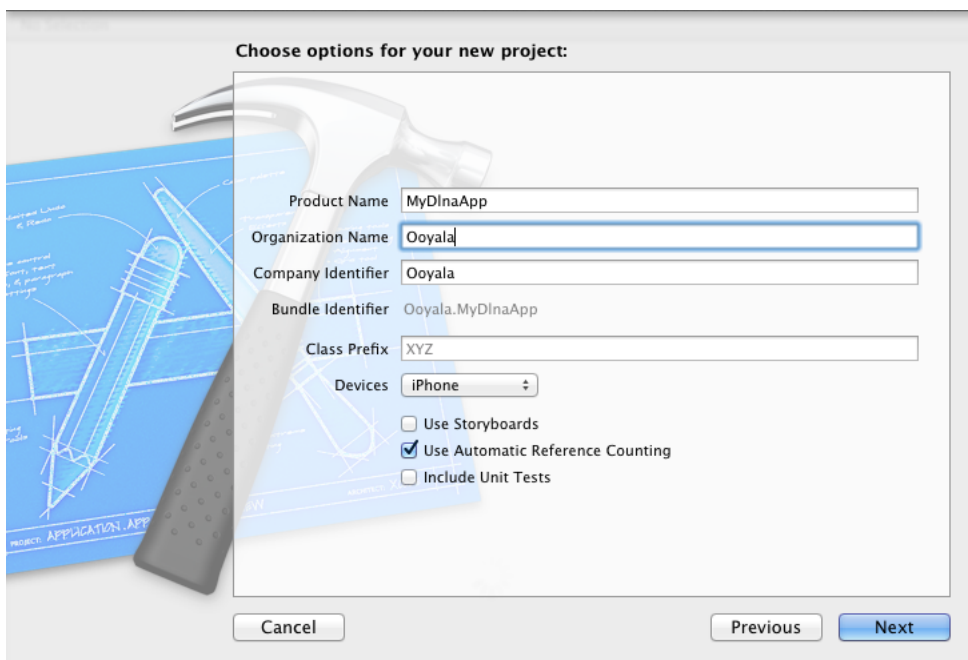
2. Select a template for your new application. In the following example, we selected the Single View Application.





### Name Your Project

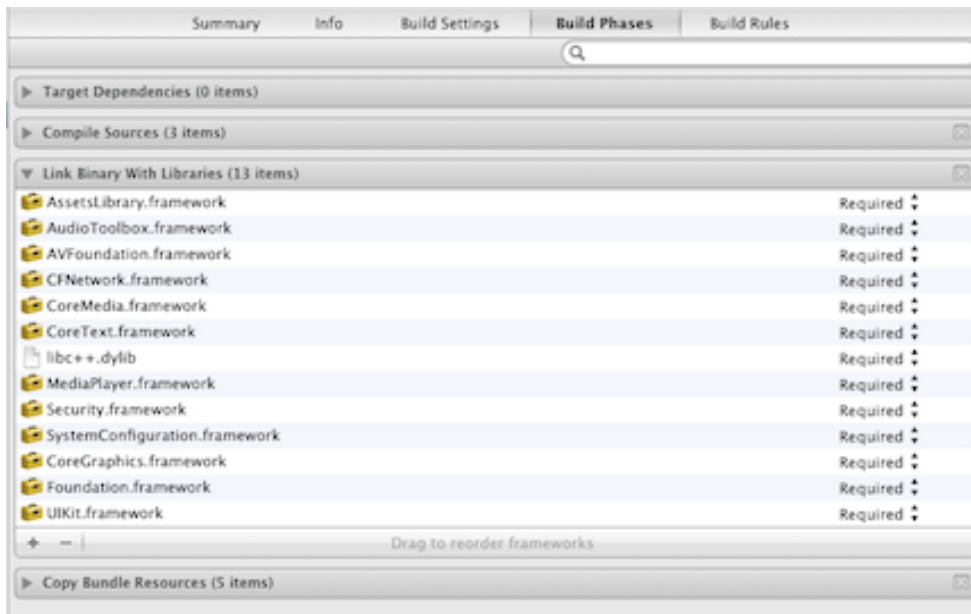
3. Enter the options for your project. Provide a Product Name (your application name), select the device (iPhone) and any other relevant option.



### Import Required Libraries

4. Your next step is to import some required libraries from the iOS SDK and XTV Connect DLNA for iOS SDK.





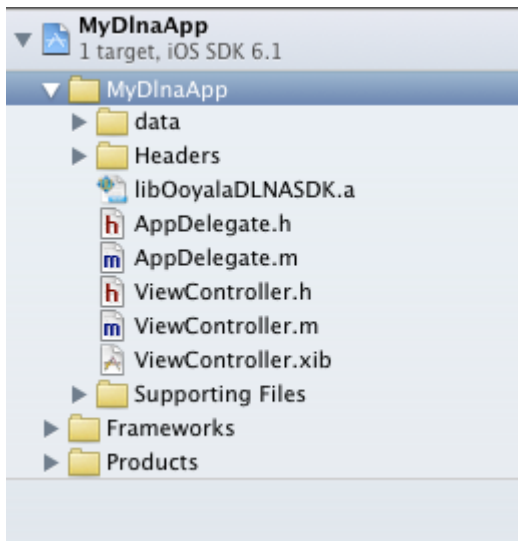
Here is a list of all of the files that you need to import into your project.

- AssetsLibrary.framework
- AudioToolbox.framework
- AVFoundation.framework
- CFNetwork.framework
- CoreGraphics.framework
- CoreMedia.framework
- CoreText.framework
- Foundation.framework
- MediaPlayer.framework
- Security.framework
- SystemConfiguration.framework
- UIKit.framework
- libc++.dylib
- libxml2.dylib

### **Copy Library, Data, Headers**

5. Open the Ooyala iOS SDK.
6. Drag and drop the iOS SDK Headers folder into your Xcode development tool.
7. Open the Ooyala XTV Connect DLNA plug-in SDK.
8. Drag and drop the XTV Connect DLNA SDK plug-in Headers folder into your Xcode development tool.





### Edit the View Controller

9. You will add the following lines of code to the ViewController. Obtain your provider ID (PCODE) from your Backlot account. For more information about this, see the topic [Your API Keys](#) in the **Ooyala Support Center > Documentation > Backlot Developer site**.
10. Add your PCODE to and your player's domain (such as `http://www.ooyala.com`) to the line:

```
[[ OODlnaPlayerViewController alloc ] initWithPcode:PCODE domain:
[[OOPlayerDomain alloc]
initWithString:PLAYERDOMAIN];
```

11. You need to specify your embed code (you can get this from Backlot) for this statement:
`[ooyalaPlayerViewController.player setEmbedCode:EMBED_CODE];`
12. You can cut and paste the lines of code in the following example as needed. You do need to make sure that you provide your specific pcode and embed code.

```
#import "ViewController.h"

// Create Ooyala ViewController
ooyalaPlayerViewController = [[ OODlnaPlayerViewController
alloc ] initWithPcode:PCODE domain:[[OOPlayerDomain alloc]
initWithString:PLAYERDOMAIN];

// Attach it to current view
[self addChildViewController:ooyalaPlayerViewController];

// Set player frame && attach the view
[ooyalaPlayerViewController.view setFrame:self.view.frame];
[self.view addSubview:ooyalaPlayerViewController.view];

// Load the video
[ooyalaPlayerViewController.player setEmbedCode:EMBED_CODE];
```

You now have everything in place to run your build.

### Build Your Project

13. After you have copied all the necessary components into your development environment, select **Run** to build your project. If successful, you will be able to use XBMC to see your results.

### Troubleshooting



If you have any trouble with your build or build results, try running through the following troubleshooting tips.

- Verify that your devices are all on the same wifi network and that the following ports are open for the indicated protocols:
  - UDP: port 1900
  - TCP: port 2869
- Make sure the data directory is properly bundled with your app. If this is incorrectly configured, a log message is generated:

```
Device DB not installed
```

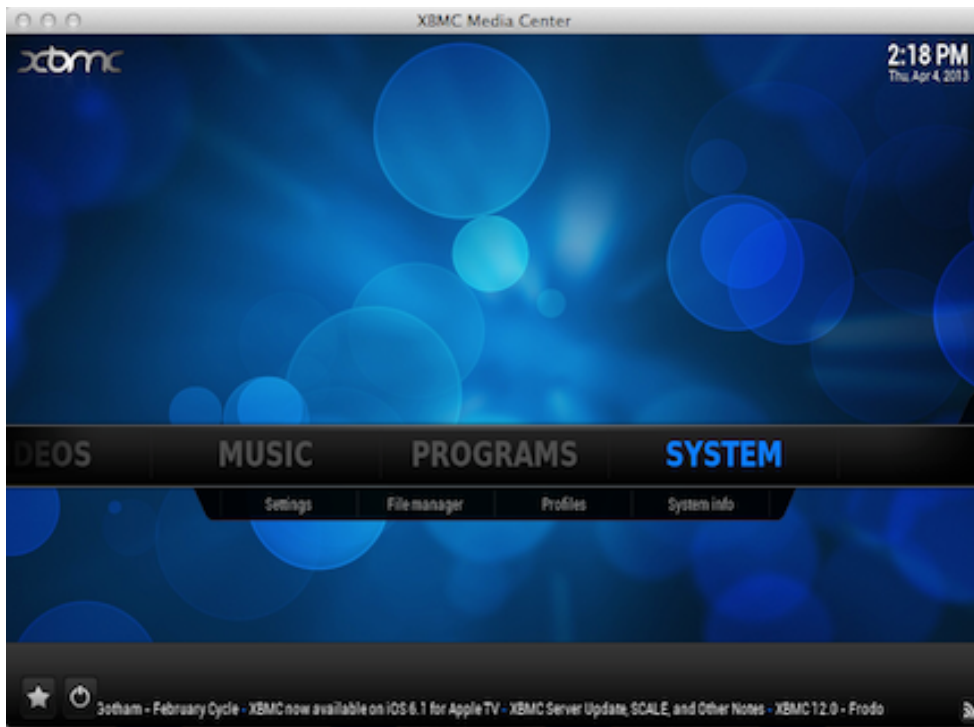
- Verify your setup with XBMC DLNA/Airplay renderers. See [Using XBMC to Configure and Test DLNA Playback](#) on page 47.

## USING XBMC TO CONFIGURE AND TEST DLNA PLAYBACK

You can use XBMC to turn your computer into an XTV-compatible device, so you can test your apps.

These are the rudimentary steps to setup XBMC.

1. Download XBMC from <http://xbmc.org/> (where you can also consult the documentation or other resources), install it, and start it.



2. Select **System**
3. Select **Services** (located at the bottom of the left navigation pane).
4. Click **UPnP** (located at the top of the left navigation pane).
5. Select the **Allow control of XBMC via UPnP** radio button.
6. Select **AirPlay** (located at the bottom of the left navigation pane).



7. Click on the **Allow XBMC to receive airplay** radio button. If you have any questions or need to troubleshoot your XBMC setup, you should consult the XBMC documentation for further assistance.

