



# **OOYALA SDK FOR ANDROID DEVELOPER GUIDE**

# CONTENTS

<b>LATEST RELEASE NOTES: MOBILE SDK FOR ANDROID V2.4.0</b>	<b>5</b>
<b>ARCHIVED RELEASE NOTES: MOBILE SDK FOR ANDROID</b>	<b>6</b>
<b>ABOUT THE MOBILE SDK FOR ANDROID</b>	<b>7</b>
<b>SETUP IN ECLIPSE</b>	<b>9</b>
<b>TUTORIAL: GETTING STARTED WITH THE MOBILE SDK</b>	<b>11</b>
<b>SAMPLE APPLICATIONS FOR THE ANDROID MOBILE SDK</b>	<b>13</b>
<b>LAYOUT CONTROLS</b>	<b>14</b>
Preventing Restart on Tilt	14
Preventing Screen Blackout/Flicker	14
Implementing Custom Controls for Android	15
<b>WORKING WITH EVENTS</b>	<b>16</b>
<b>DEALING WITH ANDROID MOBILE SDK ERRORS</b>	<b>18</b>
<b>WORKING WITH CLOSED CAPTIONS</b>	<b>20</b>
Required Closed Caption Setup in Backlot	20
Fundamental CC Methods on Android	20
<b>LOCALIZING THE USER INTERFACE</b>	<b>22</b>
<b>CROSS-DEVICE RESUME (XDR) WITH THE MOBILE SDK</b>	<b>23</b>
<b>WORKING WITH MULTI-RESOLUTION PLAYERS ON WIDEVINE</b>	<b>24</b>
<b>INTEGRATION WITH OMNITURE ON ANDROID</b>	<b>25</b>
<b>INTEGRATION WITH GOOGLE IMA ON ANDROID</b>	<b>29</b>



See the IMA Sample App in Action on Android	30
A Closer Look at the Android Sample App and Integration Touchpoints	31
Overriding the Ad Tag URL on Android	35

## **INTEGRATION WITH FREEWHEEL ON ANDROID** **37**

See the Freewheel Sample App in Action on Android	38
A Closer Look at the Android Sample App and Integration Touchpoints	40



# 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 ANDROID V2.4.0

---

These are the release notes for the mobile SDK v2.4.0 for Android.

## 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 Android has been released. Version 2.4.0 includes:

- Major Changes to Closed Captions UI and Closed Captions API.
- New PlayerDomain class (new PlayerDomain(DOMAIN)) that must be used for all instances of OoyalaPlayerLayoutController and OptimizedOoyalaPlayerLayoutController.
- 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

### Freewheel/IMA Bug fixes

- Updated Freewheel Binaries to 5.13.0.
- Fixed issue of running update before we had data.
- Addressed race conditions with FW ads.
- Pass current item duration into FreeWheel instead of Stream Player duration.
- Sync SDK helper with iOS, remove FreeWheel jar from OO+FW SDK.

### API Changes

- Major Closed Captions UI overhaul to adhere to FCC Regulations.
- Extracted 'player domain' as a concrete type and put validation enforcement in there, and updated APIs using such.
- Introduced onKeyUp event handler to show how Fire TV remote control (and other controls) can interact with OoyalaPlayer.

### Other Changes

- Fixed use of \_closedCaptionsView and \_closedCaptionsStyle to avoid NPEs.
- Use AsyncTask and Thread to respect thread requirements.
- Fixed bugs with OptimizedOoyalaPlayerLayoutController.



# ARCHIVED RELEASE NOTES: MOBILE SDK FOR ANDROID

---

Archived release notes for the mobile SDK for Android.

## NOTABLE CHANGES IN VERSION 2.1.0

Version 2.1.0 of the Ooyala Mobile SDK for Android and 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.
- Many customer-reported and unreported optimizations and bugfixes.

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



# ABOUT THE MOBILE SDK FOR ANDROID

---

The Ooyala Mobile SDK for Android is a set of Java classes and layout controllers.

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

## ABOUT THE MOBILE SDK

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.

## 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/com/ooyala/android/package-summary.html`.

## SUPPORTED CONFIGURATIONS

The Ooyala Mobile SDK for Android supports:

- Android OS version 2.2 or later
- HTTP Live Streaming (HLS) for Android 4.0 or later. If you need to support video on Android 2.2 or later, you need a third-party add-on for HLS. Contact Ooyala for details on this add-on.
- Videos in MP4 container with baseline H.264 format

The SDK is known to work with Google TV, but this is not a supported configuration.

## 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. To run custom controls over the Ooyala Player, see [Implementing Custom Controls for Android](#).

## 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 `EmbedTokenGenerator` interface. For an example, see any of the sample application, described briefly in [Sample Applications for the Android 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 Android](#) on page 37.

## 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 Android](#) on page 25.





# SETUP IN ECLIPSE

---

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

You need the following to get started with the SDK:

- The Mobile SDK for Android 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.
- A development environment for Android applications. This guide refers to the popular Eclipse development environment.

To set up Eclipse for the Ooyala Mobile SDK for Android, follow these steps to create a skeletal project. You can reuse this skeleton again and again for every Ooyala Mobile SDK application you develop.

1. Unzip the downloaded OoyalaSDK.zip file.
2. Copy the OoyalaSDK/OoyalaSDK.jar file to a suitable location of your preference.
3. Open Eclipse.
4. Click **File**, select **New**, and click **Android Project**. If **Android Project** does not appear, click **File**, select **New**, and click **Other**. Then, expand **Android** and click **Android Project**.
5. Follow the steps to create a new project.
6. Add the OoyalaSDK.jar to your Android application. For example, you might do the following:
  - Create a **lib** folder by selecting **File**, pointing to **New**, and clicking **Folder**.
  - Drag the OoyalaSDK.jar to the **lib** folder.
7. Add the OoyalaSDK.jar to the classpath of your Android application. For example, you might do the following:
  - a) Click **Project** and select **Properties**.
  - b) Select **Java Build Path** in the left pane.
  - c) Click **Add Jars**.
  - d) Navigate to and select the OoyalaSDK.jar file.
  - e) Click **OK**.
8. Select the AndroidManifest.xml file and do the following:
  - a) Click the **Permissions** tab.
  - b) Click **Add**.
  - c) Select **Uses Permission** and click **OK**.
  - d) From the **Name** list box, select **android.permission.INTERNET**.
  - e) Click the **AndroidManifest.xml** tab to verify that the following entry has been added:

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

9. Ensure the application will work with Android Version 2.2 and later by changing:

```
<uses-sdk android:minSdkVersion="15" />
```

to:

```
<uses-sdk android:minSdkVersion="8" />
```

10. This step depends on your creating an actual project, whose name is shown as *project*. Double-click *project/res/layout/main*.
11. Select the **main.xml** tab.



**12.** Add the following within the `LinearLayout` element:

```
<com.ooyala.android.OoyalaPlayerLayout android:id="@+id/ooyalaPlayer"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
</com.ooyala.android.OoyalaPlayerLayout>
```

**13.** Save the file.

You now have a reusable program skeleton.



# TUTORIAL: GETTING STARTED WITH THE MOBILE SDK

---

Let's take a close look at the "Getting Started" sample application.

For this tutorial, you need:

- 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 `SampleApps` directory in the SDK distribution contains a `GettingStartedSampleApp` directory. This basic application shows the rudimentary steps for creating a video player.

In Eclipse (or your own development environment), open the source file:

```
OoyalaSDK-Android/SampleApps/GettingStartedSampleApp/com/ooyala/android/sampleapp/GettingStartedSampleAppActivity.java
```

The application looks like this.

```
package com.ooyala.android.sampleapp;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import com.ooyala.android.OoyalaPlayer;
import com.ooyala.android.OoyalaPlayerLayout;
import com.ooyala.android.OoyalaPlayerLayoutController;

public class GettingStartedSampleAppActivity extends Activity {

    final String EMBED = "yourEmbedCodeHere"; //Embed Code, or Content ID
    final String PCODE = "yourPcodeHere";
    final String new PlayerDomain(DOMAIN) = "http://www.ooyala.com";

    /**
     * Called when the activity is first created.
     */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        OoyalaPlayerLayout playerLayout = (OoyalaPlayerLayout)
        findViewById(R.id.ooyalaPlayer);
        OoyalaPlayerLayoutController playerLayoutController =
        new OoyalaPlayerLayoutController(playerLayout, PCODE, new
        PlayerDomain(DOMAIN));
        OoyalaPlayer player = playerLayoutController.getPlayer();
        if (player.setEmbedCode(EMBED)) {
            player.play();
        } else {
            Log.d(this.getClass().getName(), "Something Went Wrong!");
        }
    }
}
```



1. The three imports are standard for use of the Ooyala SDK:

- `import com.ooyala.android.OoyalaPlayer;`
- `import com.ooyala.android.OoyalaPlayerLayout;` For more sophisticated uses, this import depends on which layout controller you want to use. For more about layouts, see [Layout Controls](#) on page 14.
- `import com.ooyala.android.OoyalaPlayerLayoutController;`

2. The sample application extends the standard Android Activity object.

3. Set the following constants:

```
final String EMBED = "yourEmbedCodeHere"; //Embed Code, or Content ID
final String PCODE = "yourPcodeHere";
```

Leave the new `PlayerDomain(DOMAIN)` constant unchanged as `http://www.ooyala.com`. This 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 *domain* variable here can be set to one of those allowed domains. If you have not set these **Syndication Controls**, the new `PlayerDomain(DOMAIN)` constant here has no effect.

4. Set the layout to the `OoyalaPlayerLayout`. For more about layouts, see [Layout Controls](#) on page 14.

5. Create a new `OoyalaPlayerController` object (named `playerLayoutController`).

6. Instantiate a player object with the `playerLayoutController.getPlayer()` method.

7. Start the video in an if test with `player.setEmbedCode()` function with the identifier of the video to play (`EMBED`). This function is the primary "work horse" of the SDK.

For your more sophisticated applications, rather than hardcoding the video identifier, you will want to pass a variable to the `setEmbedCode()` function.



# SAMPLE APPLICATIONS FOR THE ANDROID MOBILE SDK

---

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

Name	Location in SDK	Description
Adobe Pass DemoApp	<code>SampleApps/AdobePassDemoApp</code>	Illustrates how the Ooyala SDK can work in conjunction with Adobe Pass.
Channel Browser	<code>SampleApps/ChannelBrowserSampleApp</code>	This is a more fleshed-out example of a full-featured, including loading a list of videos to play, displaying a list of videos in the UI, and finally playing the video selected by user.
FragmentUISampleApp	<code>SampleApps/FragmentUISampleApp</code>	Illustrates a sample UI.
Getting Started	<code>SampleApps/GettingStartedSampleApp</code>	A basic program to illustrate the SDK. See <a href="#">Tutorial: Getting Started with the Mobile SDK</a> on page 11.
keys	<code>SampleApps/keys</code>	Keystore files for some of the sample applications. Note: This file is no longer available.
OOView	<code>SampleApps/OOView</code>	This is the most full-featured and complex sample, including custom navigation, loading of videos based on different parameters, different ways to play video and so on.



# LAYOUT CONTROLS

---

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

This directory includes source Java programs that control the layout of the video player on the device, both the default and for inline or fullscreen display, including the following:

- `OoyalaPlayerControls.java` is the program that actually implements the layout whose name you pass it.
- `OoyalaPlayerLayoutController.java` is a generic `LayoutController` that will work in most cases (regardless of the containing `Layout` type). It uses basic controls and allows additional overlays to be added. Fullscreening is done by opening a full screen `Dialog` and filling it with a dynamically created `OoyalaPlayerLayout`. Because of this, playback will be suspended and subsequently resumed during this process. As a result, fullscreening is slower than if the `OoyalaPlayerLayout` is embedded directly in the `Activity`'s base layout, that base layout is a `FrameLayout`, and the `LayoutController` used is `FastOoyalaPlayerLayoutController`.
- `DefaultOoyalaPlayerFullscreenControls.java`
- `DefaultOoyalaPlayerInlineControls.java`
- The recommended `OptimizedOoyalaPlayerLayoutController.java` is a faster `LayoutController` that will work only on one specific case: The `OoyalaPlayerLayout` it controls is a direct child of the `Activity`'s base layout which is a `FrameLayout`. This `LayoutController` uses basic controls and allows additional overlays to be added. Fullscreening is done by simply resizing the `OoyalaPlayerLayout` to fill the entire screen, which does not trigger a player reload thus causing this to be much faster at Fullscreening than `OoyalaPlayerLayoutController`.

**Note:** If you decide to use any layout other than the default, be sure to change your project's `LinearLayout` element to use the name of the layout you desire. See the final step in [Tutorial: Getting Started with the Mobile SDK](#) on page 11.

## PREVENTING RESTART ON TILT

---

This setting prevents the restarting of a video when the viewer changes the orientation of the device.

To prevent a video from restarting when the device changes orientation (vertical to horizontal or horizontal to vertical), do the following:

1. Open the application's `AndroidManifest.xml` file.
2. Add the following attribute to the `activity` element: `android:configChanges="orientation|keyboardHidden"`. For example:

```
<activity android:name=".TestOoyalaSDKActivity" android:label="@string/app_name"
    android:configChanges="orientation|keyboardHidden" >
```

## PREVENTING SCREEN BLACKOUT/FLICKER

---

This setting prevents the device's screen from blacking out or flickering when a video starts.

In some cases, the screen of your device might simply black out. This is not due to the SDK itself but rather the underlying Android system and is perhaps due to a change to the `SurfaceView`.



To prevent a blackout or flicker on the device, add the following XML to the first layout activity of your application:

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/ooyala_container"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent" >

    <SurfaceView android:layout_width="0dp" android:layout_height="0dp" />
    .
    . your first layout here
    .
```

## IMPLEMENTING CUSTOM CONTROLS FOR ANDROID

Ooyala's Android 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.

**Note:** If you use the `OptimizedOoyalaPlayerLayoutController` you only need to override the Inline controls.

1. Download the [Ooyala SDK](#).
2. Open the `DefaultControlsSource` folder provided with the SDK.
3. Copy the `Images.java`, `AbstractOoyalaPlayerControls.java`, and `DefaultOoyalaPlayerInlineControls.java` files into your project's or `GettingStartedSampleApp`'s source code.
4. Rename the `DefaultOoyalaPlayerInlineControls` (for example, `CustomOoyalaPlayerControls`).
5. Instantiate an `OoyalaPlayerLayoutController` and add the following line of code:

```
playerLayoutController.setInlineControls(new
    CustomOoyalaPlayerControls(player, playerLayout));
//or
playerLayoutController.setFullscreenControls(new
    CustomOoyalaPlayerControls(player, playerLayout));
```

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 extend the Activity object to include Ooyala's notifications.

1. Your activity also needs to implement the Observer object. For example:

```
public class OoyalaAndroidTestAppActivity extends Activity implements
    OnClickListener, Observer
```

2. Attach your activity to the player, like this:

```
player.addObserver(this);
```

3. Finally, in the activity, implement the update method:

```
@Override
public void update(Observable arg0, Object arg1) {
    Log.d(TAG, "Notification Recieved: " + arg1 + " - state: " +
        player.getState());
    if (arg1 == OoyalaPlayer.CONTENT_TREE_READY_NOTIFICATION) {
        metadataReady = true;
        Log.d(TAG, "AD - metadata true!");
    } else if (arg1 == OoyalaPlayer.METADATA_READY_NOTIFICATION) {
        Log.d(TAG, "Woot, here is the current metadata: " +
            player.getMetadata());
    }
    // if (((String)arg1).equals(OoyalaPlayer.STATE_CHANGED_NOTIFICATION)
    && ((OoyalaPlayer)arg0).getState()
    // == State.READY) {
    //     player.play();
    // }
}
```

- arg0 is always the player instance
- arg1 is the notification

Events are defined in the header file `OoyalaPlayer.java`:

```
public static final String TIME_CHANGED_NOTIFICATION = "timeChanged";
public static final String STATE_CHANGED_NOTIFICATION = "stateChanged";
public static final String BUFFER_CHANGED_NOTIFICATION = "bufferChanged";
public static final String CONTENT_TREE_READY_NOTIFICATION =
    "contentTreeReady";
public static final String AUTHORIZATION_READY_NOTIFICATION =
    "authorizationReady";
public static final String ERROR_NOTIFICATION = "error";
public static final String PLAY_STARTED_NOTIFICATION = "playStarted";
public static final String PLAY_COMPLETED_NOTIFICATION = "playCompleted";
public static final String CURRENT_ITEM_CHANGED_NOTIFICATION =
    "currentItemChanged";
public static final String AD_STARTED_NOTIFICATION = "adStarted";
public static final String AD_COMPLETED_NOTIFICATION = "adCompleted";
public static final String AD_SKIPPED_NOTIFICATION = "adSkipped";
public static final String AD_ERROR_NOTIFICATION = "adError";
public static final String METADATA_READY_NOTIFICATION = "metadataReady";
```





---

See also [Dealing with Android Mobile SDK Errors](#) on page 18.



# DEALING WITH ANDROID MOBILE SDK ERRORS

---

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

First declare the following:

```
.  
.   
.   
public static final String ERROR_NOTIFICATION = "error" ;  
/** Fires when an error occurs */  
.   
.   
.
```

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

```
public String getError() {  
    return _error;  
}
```

The errors are as follows, with explanatory comments:

```
public enum OoyalaErrorCode {  
    /** Authorization Response invalid */  
    ERROR_AUTHORIZATION_INVALID,  
    /** Content Tree Response invalid */  
    ERROR_CONTENT_TREE_INVALID,  
    /** Authorization failed */  
    ERROR_AUTHORIZATION_FAILED,  
    /** The signature of the Authorization Response is invalid */  
    ERROR_AUTHORIZATION_SIGNATURE_INVALID,  
    /** Content Tree Next failed */  
    ERROR_CONTENT_TREE_NEXT_FAILED,  
    /** An Internal Android Error. Check the Throwable properties. */  
    ERROR_INTERNAL_ANDROID,  
    /** Playback failed */  
    ERROR_PLAYBACK_FAILED,  
    /** Authorization Heartbeat failed. Check properties. */  
    ERROR_AUTHORIZATION_HEARTBEAT_FAILED,  
    /** Metadata fetch failed*/  
    ERROR_METADATA_FETCH_FAILED,  
  
};
```

## AUTHORIZATION ERROR CODES

You can examine the `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 CC METHODS ON ANDROID

---

A simple programming example shows the basic methods on Android 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 20. This simple example shows some of the basic method calls for programming with closed captions on Android:

- Shown first here is a check for the `CURRENT_ITEM_CHANGED_NOTIFICATION` 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 16.)
- To find out what languages are available for the video, use the `getAvailableClosedCaptionsLanguages()` method, which returns the information in a string named `languages` in this example..



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

```
if (notification == OoyalaPlayer.CURRENT_ITEM_CHANGED_NOTIFICATION) {  
    Set<String> languages = player.getAvailableClosedCaptionsLanguages();  
    if (languages.contains("en")) {  
        player.setClosedCaptionsLanguage("en");  
    }  
}
```



# LOCALIZING THE USER INTERFACE

---

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

To change the language of the text strings in the Mobile SDK's user interface, you can add the following key/value pairs to the resource file `res/values/strings.xml`. You can then customize each value as you like:

```
<string name="oo_live">Live</string>
<string name="oo_subtitles_title">Subtitles</string>
<string name="oo_subtitles_none">None</string>
<string name="oo_subtitles_cc">Closed Captions</string>
<string name="oo_subtitles_en">English</string>
<string name="oo_subtitles_es">Spanish</string>
```

For each desired language, specify the two-character ISO 639-1 language code on the `oo_subtitles_` key. For instance, as shown above for Spanish: `oo_subtitles_es`.

To assist in your debugging, the Mobile SDK looks up values from the resource file by the classes `getContext().getResources().getIdentifier(name, "string",`  
`getContext().getPackageName().`



# 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 \(XDR\)](#).



# WORKING WITH MULTI-RESOLUTION PLAYERS 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 `DefaultPlayerInfo` class.

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

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

```
// Android
class CustomPlayerInfo extends DefaultPlayerInfo {
    public int getMaxWidth() { return 1200; }
    public int getMaxHeight() { return 700; }
}

public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    StreamPlayer.defaultPlayerInfo = new CustomPlayerInfo();

    ...

    OoyalaPlayerLayoutController playerLayoutController =
    new OoyalaPlayerLayoutController(playerLayout, PCODE, new
    PlayerDomain(DOMAIN));

    ...
```





# INTEGRATION WITH OMNITURE ON ANDROID

---

The Ooyala Omniture Integration App demonstrates how you can integrate Omniture analytics capabilities into your Android 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 in SiteCatalyst® reporting. You can integrate Ooyala mobile SDK with Omniture SDKs through a step-by-step integration process using our sample app as a model.

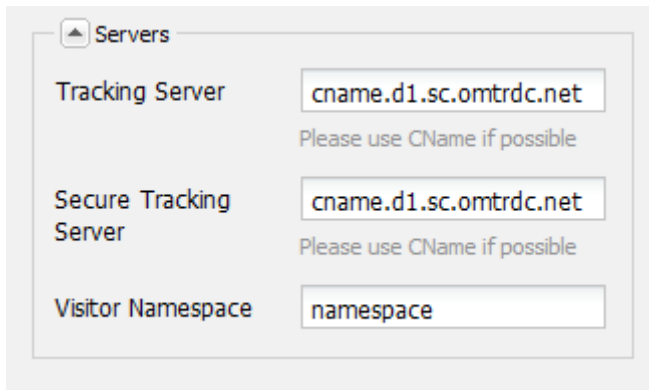
## WHAT YOU NEED

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

1. The [Android SDK](#).
2. The [Ooyala Mobile SDK for Android](#).
3. The [Ooyala Omniture Integration Sample App](#)
4. The [Omniture SDK](#). Note that Omniture was purchased by Adobe and is now marketed as The Adobe® Marketing Cloud Mobile.
5. [Eclipse IDE](#). In this guide, we use the Eclipse IDE to illustrate our 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 the 'Servers' configuration window in SiteCatalyst. The window has a title bar with a small icon and the text 'Servers'. It contains three rows of configuration fields. The first row is 'Tracking Server' with a text box containing 'cname.d1.sc.omtrdc.net' and a note below it that says 'Please use CName if possible'. The second row is 'Secure Tracking Server' with a text box containing 'cname.d1.sc.omtrdc.net' and the same note below it. The third row is 'Visitor Namespace' with a text box containing 'namespace'.

You will use this information in [Edit TrackingHelper.java](#) on page 26.

## OPEN THE ANDROID 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 Eclipse IDE. The Eclipse tool will help with your Android development effort. To get started with your development project, launch your Eclipse app.



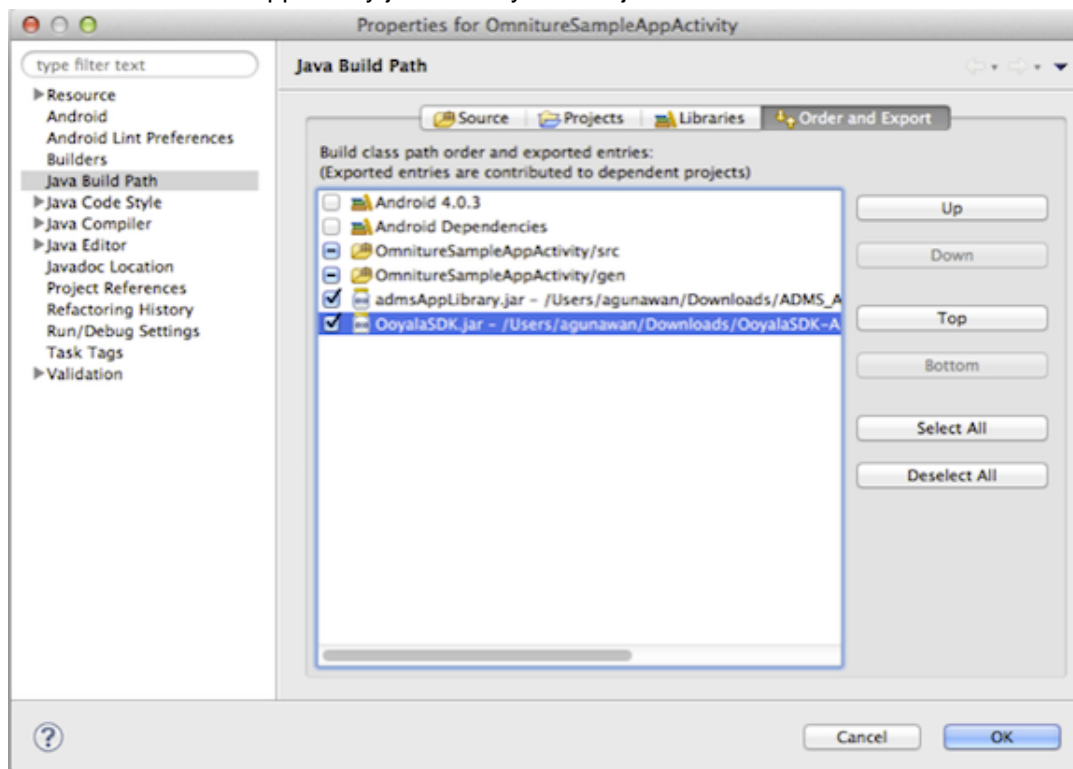
## OPEN THE SAMPLE APP PROJECT

1. Click **Open Eclipse > File > Import**
2. Go to **Android > Existing Android Code into Workspace > Next**
3. Set the “Root Directory” to the extracted OmnitureSampleAppActivity folder.
4. Click **Finish**.

## IMPORT REQUIRED LIBRARIES

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

1. Right-click the OmnitureSampleAppActivity project name.
2. Select **Build Path > Configure Build Path...**
3. Select **Java Build Path > Libraries** tab > **Add External Jars**.
4. Add the `admsAppLibrary.jar` that you downloaded and extracted from the Omniture SDK.jar.
5. Add the `OoyalaSDK.jar` that you downloaded and extracted from the Ooyala Android SDK.
6. Click the **Order and Export** tab.
7. Make sure the `admsAppLibrary.jar` and `OoyalaSDK.jar` are selected.



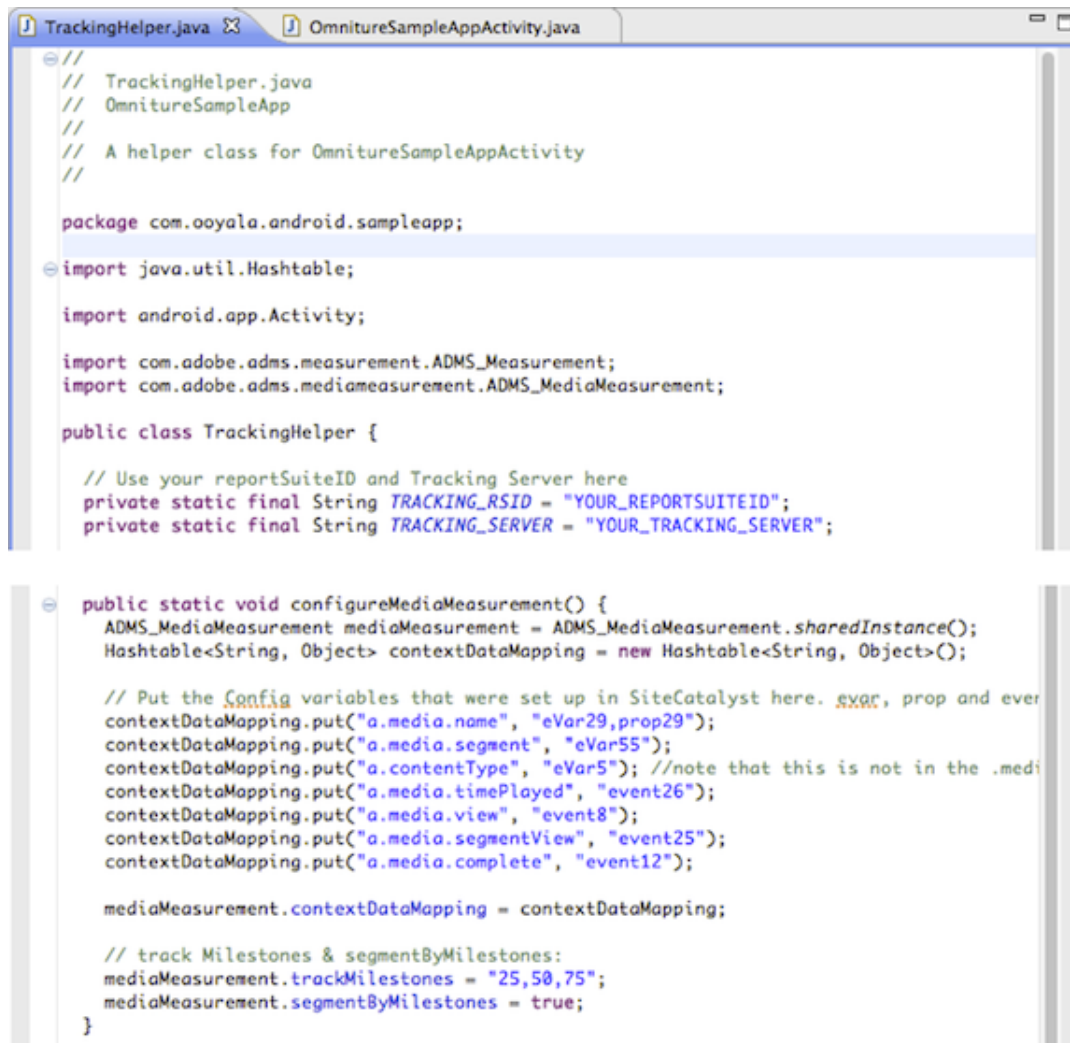
## EDIT TRACKINGHELPER.JAVA

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

1. Open the `TrackingHelper.java` file. You will make some modifications to this file with the information you saved from SiteCatalyst, as described in [What You Need](#) on page 25.
2. In your `TrackingHelper.java` file, change:



- YOUR\_REPORTSUITEID to match the one you got from SiteCatalyst.
  - YOUR\_TRACKING\_SERVER to match the one from SiteCatalyst.
3. In the TrackingHelper.java file, you also need to change the following configuration variables so that they match the equivalent variables in SiteCatalyst:
- eVars - (s.eVarN)
  - s.props - (s.propN)
  - s.events - (s.events)



```
//
// TrackingHelper.java
// OmnitureSampleApp
//
// A helper class for OmnitureSampleAppActivity
//

package com.ooyala.android.sampleapp;

import java.util.Hashtable;

import android.app.Activity;

import com.adobe.adms.measurement.ADMS_Measurement;
import com.adobe.adms.mediamasurement.ADMS_MediaMeasurement;

public class TrackingHelper {

    // Use your reportSuiteID and Tracking Server here
    private static final String TRACKING_RSID = "YOUR_REPORTSUITEID";
    private static final String TRACKING_SERVER = "YOUR_TRACKING_SERVER";

    public static void configureMediaMeasurement() {
        ADMS_MediaMeasurement mediaMeasurement = ADMS_MediaMeasurement.sharedInstance();
        Hashtable<String, Object> contextDataMapping = new Hashtable<String, Object>();

        // Put the Config variables that were set up in SiteCatalyst here. eVar, prop and event
        contextDataMapping.put("a.media.name", "eVar29,prop29");
        contextDataMapping.put("a.media.segment", "eVar55");
        contextDataMapping.put("a.contentType", "eVar5"); //note that this is not in the .medi
        contextDataMapping.put("a.media.timePlayed", "event26");
        contextDataMapping.put("a.media.view", "event8");
        contextDataMapping.put("a.media.segmentView", "event25");
        contextDataMapping.put("a.media.complete", "event12");

        mediaMeasurement.contextDataMapping = contextDataMapping;

        // track Milestones & segmentByMilestones:
        mediaMeasurement.trackMilestones = "25,50,75";
        mediaMeasurement.segmentByMilestones = true;
    }
}
```

4. If you followed these steps, your ReportSuiteID, TrackingServer and Configuration variables (eVar, prop and events) should match your SiteCatalyst configurations.
5. Although the sample contains an embed code and pcode, replace these with your own embed code and pcode before running the build.
6. Run the OmnitureSampleAppActivity.java!
7. When you run your build successfully, the Android 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 into your development environment, select Run to build your project. If successful, you will be able to see Omniture analytics display 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 relevant Adobe documentation or representative, if necessary.



# INTEGRATION WITH GOOGLE IMA ON ANDROID

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 Android SDK for Google IMA, download the following:

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

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

3. [Google Android SDK](#)
4. [Eclipse IDE](#)

## 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.

## STRUCTURE OF THE OOOYALA GOOGLE IMA SDK

The SDK has the following directories and files.

Directory/File	Description
Documentation	Reference java docs for the OoyalaIMAManager class
IMASampleApp	A sample application implementing the Ooyala SDK
OoyalaIMASDK.jar	The Ooyala SDK jar file you need to add to any new Eclipse project
VERSION	Version number of the Ooyala SDK for Google IMA



## SEE THE IMA SAMPLE APP IN ACTION ON ANDROID

---

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

1. Download the [Ooyala Google IMA SDK for Android](#).
2. Unzip the OoyalaIMASDK-Android.zip file.
3. Go to the subdirectory OoyalaIMASDK-Android/IMASampleApp.
4. In Eclipse, open the file src/ooyala/com/android/imasampleapp/IMASampleAppActivity.java.

```
package com.ooyala.android.imasampleapp;

import com.ooyala.android.imasdk.*;
import com.ooyala.android.imasampleapp.R;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.ViewGroup;

import com.ooyala.android.OoyalaPlayer;
import com.ooyala.android.OoyalaPlayerLayout;
import com.ooyala.android.OptimizedOoyalaPlayerLayoutController;

public class IMASampleAppActivity extends Activity {

    final String EMBED = "h5OWFoYTrG4YIPdrDKrIz5-VhobsuT-M"; //Embed Code,
    or Content ID
    final String PCODE = "R2d3I6s06RyB7l2DN0_2GsQS-R-Y";
    final String new PlayerDomain(DOMAIN) = "http://www.ooyala.com";
    OptimizedOoyalaPlayerLayoutController playerLayoutController;
    OoyalaIMAManager imaManager;
    /**
     * Called when the activity is first created.
     */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        OoyalaPlayerLayout playerLayout = (OoyalaPlayerLayout)
        findViewById(R.id.ooyalaPlayer);
        playerLayoutController = new
        OptimizedOoyalaPlayerLayoutController(playerLayout, PCODE, new
        PlayerDomain(DOMAIN));
        OoyalaPlayer player = playerLayoutController.getPlayer();

        //Initialize IMA classes
        imaManager = new OoyalaIMAManager(player);
        ViewGroup companionView = (ViewGroup)
        findViewById(R.id.companionFrame);

        imaManager.addCompanionSlot(companionView, 300, 50);

        if (player.setEmbedCode(EMBED)) {
            player.play();
        } else {
            Log.d(this.getClass().getName(), "Something Went Wrong!");
        }
    }
}
```



```

    }
}

@Override
protected void onStop() {
    super.onStop();
    if (playerLayoutController.getPlayer() != null) {
        playerLayoutController.getPlayer().suspend();
    }
}

@Override
protected void onRestart() {
    super.onRestart();
    if (playerLayoutController.getPlayer() != null) {
        playerLayoutController.getPlayer().resume();
    }
}
}

```

5. Focus on the lines highlighted in the graphic above and replace the values of the following constants:
  - **EMBED**: The identifier for one of your video assets that has associated IMA ads.
  - **PCODE**: Your Ooyala-supplied provider ID code, displayed in your Backlot account on the **ACCOUNTS** tab, **Developer** subtab.
  - **new PlayerDomain(DOMAIN)**: 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.
6. Save your changes.
7. From the **Run** menu, select **Run**.  
The Android Virtual Device (AVD) selector appears.
8. Select a virtual device or attach real hardware to your system.
9. Click the play button to view the video.

## A CLOSER LOOK AT THE ANDROID 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:

- Setup the Library for Your Project
- configChanges in AndroidManifest.xml File
- The Source for the Sample App
- The Imports
- The Constants
- Set Up the Controller, Initialize the OoyalaIMAManager and Classes
- Companion Ad Slot
- Append Ad Tag Parameters to your Ad Tag URL
- Play the Video



## SETUP THE LIBRARY FOR YOUR PROJECT

For a new project, be sure to put the following jar files in the lib or libs directory:

- OoyalaSDK.jar: the jar file from the baseline Ooyala Mobile SDK for Android
- OoyalaIMASDK.jar: The jar file from the Ooyala SDK for Google IMA on Android

## CONFIGCHANGES ANDROIDMANIFEST.XML FILE

In the `AndroidManifest.xml` file, add the `android:configChanges` attribute (see highlighted line below) on the applications' `<activity>` declaration.

**Note:** This is only required if you use the `OptimizedOoyalaPlayerLayoutController`, which is recommended. This is discussed further below.

```
.
.
.
    <application>
        .
        .
        .
        <activity
            android:name="com.ooyala.android.imasampleapp.IMASampleAppActivity"
            android:label="@string/app_name"
            android:configChanges="orientation|keyboardHidden" >
                .
                .
                .
            </activity>
        </application>
```

## THE SOURCE FOR THE SAMPLE APP

Open the `IMASampleApp/src/ooyala/com/android/imasampleapp/IMASampleAppActivity.java` file.

## THE IMPORTS

Examine the `import` statements at the top of the file:

- The first two import statements pull in definitions from the Ooyala SDK for Google IMA.
- The next four are standard Android imports.
- The final three pull definitions for players and view controllers from the baseline Ooyala Mobile SDK for Android.

```
.
.
.
import com.ooyala.android.imasdk.*;
import com.ooyala.android.imasampleapp.R;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.ViewGroup;
```





```
import com.ooyala.android.OoyalaPlayer;
import com.ooyala.android.OoyalaPlayerLayout;
import com.ooyala.android.OptimizedOoyalaPlayerLayoutController;

.
.
.
```

## THE CONSTANTS

The topic [Tutorial: Getting Started with the Mobile SDK](#) on page 11 introduces the constants `PCODE`, `EMBED`, and `new PlayerDomain(DOMAIN)`:

```
.
.
.
    final String EMBED = "h5OWFoYTrG4YIPdrDKrIz5-VhobsuT-M"; //Embed Code,
    or Content ID
    final String PCODE = "R2d3I6s06RyB7l2DN0_2GsQS-R-Y";
    final String new PlayerDomain(DOMAIN) = "http://www.ooyala.com";
.
.
.
```

Whereas the sample app defines these as constants, you probably want to define variables, especially for the `EMBED` constant (asset ID or content ID).

## SET UP THE CONTROLLER, INITIALIZE THE OYALAIMAMANAGER AND CLASSES

Various layout controllers combined with the `OoyalaIMAManager` are included with the SDK. [The basic views you can choose from are included in the baseline Mobile SDK for Android.](#)

**Note:** For working with Google IMA, it is highly recommended that you use the `OptimizedOoyalaPlayerLayoutController` layout controller.

In the sample app, a layout controller is declared of type `OptimizedOoyalaPlayerLayoutController`.

Finally, the `imaManager` object is initialized as type `OOIMAManager` with a player:

```
.
.
.
    OptimizedOoyalaPlayerLayoutController playerLayoutController;
    OoyalaIMAManager imaManager;
/**
 * Called when the activity is first created.
 */
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    OoyalaPlayerLayout playerLayout = (OoyalaPlayerLayout)
    findViewById(R.id.ooyalaPlayer);
    playerLayoutController = new
    OptimizedOoyalaPlayerLayoutController(playerLayout, PCODE, new
    PlayerDomain(DOMAIN));
    OoyalaPlayer player = playerLayoutController.getPlayer();
```



```
//Initialize IMA classes
imaManager = new OoyalaIMAManager(player);
.
.
.
```

For reference documentation about the `OoyalaIMAManager`, see the `OoyalaIMASDK-Android/Documentation` subdirectory.

## COMPANION AD SLOT

A companion view is declared of type `ViewGroup`, and companion ad slot with that view, with width and height in pixels, is defined for the `imaManager` object.

```
.
.
.
    ViewGroup companionView = (ViewGroup)
    findViewById(R.id.companionFrame);
    imaManager.addCompanionSlot(companionView, 300, 50);
.
.
.
```

## APPEND AD TAG PARAMETERS TO YOUR 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 a `HashMap` with two strings to hold the parameters you want to add to the ad tag URL. In the sample app, the hash is called `adTagParameters`.

```
.
.
.
    Map<String, String> adTagParameters = new HashMap<String, String>();

    adTagParameters.put("vid", EMBED);
    adTagParameters.put("url", "[referrer_url]");
    adTagParameters.put("pod", "2");
    adTagParameters.put("ppos", "2");
    adTagParameters.put("vpos", "preroll");
    adTagParameters.put("mridx", "2");
    imaManager.setAdTagParameters(adTagParameters);
.
.
.
```

2. Then for each parameter you want to add, use `put()` to place the tag name/value pair into the hash. In the sample, notice that the first ad tag name/value pair sets the `vid` tag to the value of the `EMBED` constant (the video or asset ID).
3. Finally, after you have populated the hash, pass it to the `imaManager.setadTagParameters()` method.

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 Android](#) on page 35.

## PLAY THE VIDEO

Finally, the video is played with the `setEmbedCode()` method, which comes with [the baseline Mobile SDK](#):

```
.
.
.
    if (player.setEmbedCode(EMBED)) {
        player.play();
    } else {
        Log.d(this.getClass().getName(), "Something Went Wrong!");
    }
.
.
.
```

## OVERRIDING THE AD TAG URL ON ANDROID

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()` method, 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:
imaManager.setAdUrlOverride("http://pubads.g.doubleclick.net/gampad/ads?
sz=640x480...");
.
. set the embed code
.
```



```
// Remember to 'null' it out later if need be: As long as it is non-null,  
the value will override any IMA settings from Backlot.  
imaManager.setAdUrlOverride(null);  
.  
.  
.
```



# INTEGRATION WITH FREEWHEEL ON ANDROID

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 Android SDK for Freewheel, download the following:

1. The [Google's Android SDK](#)
2. The [Ooyala Mobile SDK for Android](#)
3. The [Ooyala Freewheel SDK for Android](#) (Ooyala FreeWheel Android)
4. [FWAdManager.jar](#) from the FreeWheel website (you need Freewheel credentials to download the zip)
5. [Eclipse IDE](#)

**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.

## STRUCTURE OF THE OOOYALA FREEWHEEL SDK

The SDK has the following directories and files.

Directory/File	Description
Documentation	Reference Java docs for the OoyalaFreewheelManager class
FreewheelSampleApp	A sample application implementing the Ooyala SDK
OoyalaFreewheelSDK.jar	The Ooyala SDK jar file you need to add to any new Eclipse project
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 integration 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\_android\_ad\_server: The URL for serving ads on Android
- fw\_android\_player\_profile: The defined profile for the player on Android

```
//freewheelParameters.put("fw_android_mrm_network_id", "90750");
freewheelParameters.put("fw_android_ad_server", "http://demo.v.fwmrm.net/");
freewheelParameters.put("fw_android_player_profile",
    "90750:ooyala_android");
//freewheelParameters.put("fw_android_site_section_id",
    "ooyala_test_site_section");
//freewheelParameters.put("fw_android_video_asset_id",
    "ooyala_test_video_with_bvi_cuepoints");
//freewheelParameters.put("FRMSegment",
    "channel=TEST;subchannel=TEST;section=TEST;mode=online;player=ooyala;beta=n");
```

## SEE THE FREEWHEEL SAMPLE APP IN ACTION ON ANDROID

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

1. Download the [Ooyala Freewheel SDK for Android](#).
2. Unzip the OoyalaFreeweheelSDK-Android.zip file.
3. Go to the subdirectory OoyalaFreeweheelSDK-Android/FreewheelSampleApp.
4. In Eclipse, open the file src/ooyala/com/android/freewheelsampleapp/FreewheelSampleAppActivity.java, which is shown below.

```
package com.ooyala.android.freewheelsampleapp;

import java.util.HashMap;
import java.util.Map;

import com.ooyala.android.freewheelsdk.OoyalaFreewheelManager;
import com.ooyala.android.freewheelsampleapp.R;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

import com.ooyala.android.OoyalaPlayer;
import com.ooyala.android.OoyalaPlayerLayout;
import com.ooyala.android.OptimizedOoyalaPlayerLayoutController;

public class FreewheelSampleAppActivity extends Activity {

    final String EMBED = "RlODZyZDr93PAbk-a9fY7Phq93pA-Uwt";
    final String PCODE = "5idHc6Pt1kJ18w4u9Q5jEwAQDYCH";
    final String new PlayerDomain(DOMAIN) = "http://www.ooyala.com";

    OptimizedOoyalaPlayerLayoutController playerLayoutController;
    OoyalaFreewheelManager freewheelManager;

    /**
     * Called when the activity is first created.
     */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
```



```

        setContentView(R.layout.main);
        OoyalaPlayerLayout playerLayout = (OoyalaPlayerLayout)
        findViewById(R.id.ooyalaPlayer);
        playerLayoutController = new
        OptimizedOoyalaPlayerLayoutController(playerLayout, PCODE, new
        PlayerDomain(DOMAIN));
        OoyalaPlayer player = playerLayoutController.getPlayer();

        //Initialize Freewheel Ad Manager
        freewheelManager = new OoyalaFreewheelManager(this,
        playerLayoutController);

        //Set Freewheel parameters. Note that these are optional, and
        override configurations set in Backlot or Ooyala internals
        Map<String, String> freewheelParameters = new HashMap<String,
        String>();
        //freewheelParameters.put("fw_android_mrm_network_id", "90750");
        freewheelParameters.put("fw_android_ad_server", "http://
        demo.v.fwmrm.net/");
        freewheelParameters.put("fw_android_player_profile",
        "90750:ooyala_android");
        //freewheelParameters.put("fw_android_site_section_id",
        "ooyala_test_site_section");
        //freewheelParameters.put("fw_android_video_asset_id",
        "ooyala_test_video_with_bvi_cuepoints");

        freewheelManager.overrideFreewheelParameters(freewheelParameters);

        if (player.setEmbedCode(EMBED)) {
            player.play();
        } else {
            Log.d(this.getClass().getName(), "Something Went Wrong!");
        }
    }

    @Override
    protected void onStop() {
        super.onStop();
        if (playerLayoutController.getPlayer() != null) {
            playerLayoutController.getPlayer().suspend();
        }
    }

    @Override
    protected void onRestart() {
        super.onRestart();
        if (playerLayoutController.getPlayer() != null) {
            playerLayoutController.getPlayer().resume();
        }
    }
}

```

5. Focus on the lines highlighted in the graphic above and replace the values of the following constants:

- **EMBED**: 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.
- **new PlayerDomain(DOMAIN)**: 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.

6. Save your changes.



7. From the **Run** menu, select **Run**.

The Android Virtual Device (AVD) selector appears.

8. Select a virtual device or attach real hardware to your system.
9. Click the play button to view the video.

## A CLOSER LOOK AT THE ANDROID 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:

- Setup the Library for Your Project
- `configChanges` in `AndroidManifest.xml` File
- The Source for the Sample App
- The Imports
- The Constants
- Set Up the Controller, Initialize the `OoyalaFreewheelManager` and Classes
- Freewheel Parameters
- Play the Video

### SETUP THE LIBRARY FOR YOUR PROJECT

For a new project, be sure to put the following jar files in the `lib` or `libs` directory:

- `OoyalaSDK.jar`: the jar file from the baseline Ooyala Mobile SDK for Android
- `OoyalaFreewheelSDK.jar`: The jar file from the Ooyala SDK for Google IMA on Android

### CONFIGCHANGES ANDROIDMANIFEST.XML FILE

In the `AndroidManifest.xml` file, add the `android:configChanges` attribute (see highlighted line below) on the applications' `<activity>` declaration.

**Note:** This is only required if you use the `OptimizedOoyalaPlayerLayoutController`, which is recommended. This is discussed further below.

```
.  
.   
.   
    <application>  
        .  
        .  
        .  
        <activity  
            android:name="com.ooyala.android.imasampleapp.IMASampleAppActivity"  
            android:label="@string/app_name"  
            android:configChanges="orientation|keyboardHidden" >  
                .  
                .  
                .  
            </activity>  
        </application>
```





## THE SOURCE FOR THE SAMPLE APP

Open the `FreewheelSampleApp/src/ooyala/com/android/freewheelsampleapp/FreewheelSampleAppActivity.java` file.

## THE IMPORTS

Examine the `import` statements at the top of the file:

- The first two import statements pull in standard Java definitions for `Map` and `HashMap`.
- The next two pull in definitions for the `OoyalaFreewheelManager` and the sample application.
- The next three are standard Android imports.
- The final three pull definitions for player layouts and optimized layout controllers from the baseline Ooyala Mobile SDK for Android.

```
.
.
.
import java.util.HashMap;
import java.util.Map;

import com.ooyala.android.freewheelsdk.OoyalaFreewheelManager;
import com.ooyala.android.freewheelsampleapp.R;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;

import com.ooyala.android.OoyalaPlayer;
import com.ooyala.android.OoyalaPlayerLayout;
import com.ooyala.android.OptimizedOoyalaPlayerLayoutController;
.
.
.
```

## THE CONSTANTS

The topic [Tutorial: Getting Started with the Mobile SDK](#) on page 11 introduces the constants `PCODE`, `EMBED`, and new `PlayerDomain(DOMAIN)`, which are further discussed in [See the Freewheel Sample App in Action on Android](#) on page 38.

```
.
.
.
final String EMBED = "RlODZyZDr93PAbk-a9fY7Phq93pA-Uwt ";
final String PCODE = "5idHc6Pt1kJ18w4u9Q5jEwAQDYCH";
final String new PlayerDomain(DOMAIN) = "http://www.ooyala.com";
.
.
.
```

Whereas the sample app defines these as constants, you probably want to define variables, especially for the `EMBED` constant (asset ID or content ID).



## SET UP THE CONTROLLER, INITIALIZE THE OYOALAFREEWHEELMANAGER AND CLASSES

Various layout controllers combined with the OoyalaFreewheelManager are included with the SDK. *The basic views you can choose from are included in the baseline Mobile SDK for Android.*

**Note:** For working with Freewheel, it is highly recommended that you use the OptimizedOoyalaPlayerLayoutController layout controller.

In the sample app, a layout controller is declared of type OptimizedOoyalaPlayerLayoutController.

Finally, the freewheelManager object is initialized as type OoyalaFreewheelManager with this activity and the layout controller:

```
.
.
.
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        OoyalaPlayerLayout playerLayout = (OoyalaPlayerLayout)
        findViewById(R.id.ooyalaPlayer);
        playerLayoutController = new
        OptimizedOoyalaPlayerLayoutController(playerLayout, PCODE, new
        PlayerDomain(DOMAIN));
        OoyalaPlayer player = playerLayoutController.getPlayer();

        //Initialize Freewheel Ad Manager
        freewheelManager = new OoyalaFreewheelManager(this,
        playerLayoutController);
.
.
.
```

For reference documentation about the OoyalaFreewheelManager, see the OoyalaFreewheelSDK-Android/Documentation subdirectory.

## FREEWHEEL PARAMETERS

For background, see the discussion in *Essential Parameters and FreeWheel OPF Module Ad Set* on page 37. The two parameters highlighted below must be set in your app itself.

```
.
.
.
//Set Freewheel parameters. Note that these are optional, and override
configurations set in Backlot or Ooyala internals
Map<String, String> freewheelParameters = new HashMap<String, String>();
//freewheelParameters.put("fw_android_mrm_network_id", "90750");
freewheelParameters.put("fw_android_ad_server", "http://demo.v.fwmrm.net/");
freewheelParameters.put("fw_android_player_profile",
"90750:ooyala_android");
//freewheelParameters.put("fw_android_site_section_id",
"ooyala_test_site_section");
//freewheelParameters.put("fw_android_video_asset_id",
"ooyala_test_video_with_bvi_cuepoints");
//freewheelParameters.put("FRMSegment",
"channel=TEST;subchannel=TEST;section=TEST;mode=online;player=ooyala;beta=n");
```



```
freewheelManager.overrideFreewheelParameters(freewheelParameters);  
.  
.  
.
```

## PLAY THE VIDEO

Finally, the video is played with the `setEmbedCode()` method, which comes with [the baseline Mobile SDK](#):

```
.  
.  
.  
    if (player.setEmbedCode(EMBED)) {  
        player.play();  
    } else {  
        Log.d(this.getClass().getName(), "Something Went Wrong!");  
    }  
.  
.  
.
```

