

**Google AdSense for Video**  
**Flash Integration Guide (AS2 & AS3)**

Revised: 10/14/2008

# CONTENTS

<b>Chapter 1</b>	<b>3 Overview</b>
	3 Introduction
	3 Product Overview
	3 User Experience
<b>Chapter 2</b>	<b>6 Policies</b>
	6 AdSense for Video Program Policies
	7 Policy Links
<b>Chapter 3</b>	<b>8 Targeting</b>
	8 Video Ads, InVideo Graphical Overlay Ads, and Fullscreen Image Endcap Ads
	8 Text Overlay Ads and Fullscreen Text Endcap Ads
<b>Chapter 4</b>	<b>9 Flash Integration</b>
	9 Important Note on Testing Procedure
	9 Testing in the Flash IDE
	9 Pre-Ad Request Set Up
	10 Ad Request Parameters
	12 Ad Request Callback Object
	12 Setting up the Post-Ad Request Callback Method
	13 Executing the AFV Ad Request
<b>Chapter 5</b>	<b>14 HTML</b>
	14 Standard HTML Set Up (Firefox and IE Support)
	14 SWFObject Set Up (Firefox and IE Support)
	14 Embed Code Provided to Users
<b>Chapter 6</b>	<b>15 Testing</b>
	15 Video Ads
	15 Fullscreen Endcap Ads
	16 Overlay Ads
<b>Appendix A</b>	<b>17 Troubleshooting</b>
	17 Troubleshooting Tools
	17 Ad Rendering Issues
	19 Ad Targeting Issues
	20 Pause/ResumeContentVideo Issues
	20 Channels Issues
<b>Appendix B</b>	<b>21 Using onStateChange()</b>
<b>Appendix C</b>	<b>22 Channels</b>
	22 Navigating to the Channel Creation Page
	22 Creating a Reporting Channel
	23 Creating a Targetable Channel
	24 Passing in the Channel ID in Your Ad Request
<b>Appendix D</b>	<b>25 Daisy Chaining</b>
	25 Channels and Minimum CPM
	26 General Procedure
	27 Reporting by Ad Format

# CHAPTER 1: OVERVIEW

## Introduction

Thank you for your interest in the AdSense for Video program! This document is designed to provide you with the complete technical specifications for integrating AFV pre-, mid-, and post-roll video ads, AFV fullscreen endcap ads (text and image), and AFV overlay ads (text and graphical) into your custom Flash video player using AS2 or AS3. If you have any questions about the implementation process, please contact your AFV Technical Account Manager (TAM).

## Product Overview

There are several AFV ad formats available. Video ads can be shown as pre-rolls, mid-rolls, or post-rolls. Fullscreen text and image endcap ads can be shown at the end of a video. Text overlay and InVideo graphical overlay advertisements are served during video stream playback, appearing at the bottom of the display. Publishers can select which videos they would like to monetize and will be able to view reporting for their videos through the Google AdSense website. All AFV ads include the “Ads by Google” attribution, with the exception of InVideo graphical overlay ads.

## User Experience

### Video Advertisements:

To receive a video ad, a single ad request for the ad type “video” should be made. Video ads can be used in pre-roll, mid-roll, or post-roll ad slots. Video ads can be daisy chained with other ads (see Appendix D for more info).

The screenshot below shows the user experience for a **video ad**:



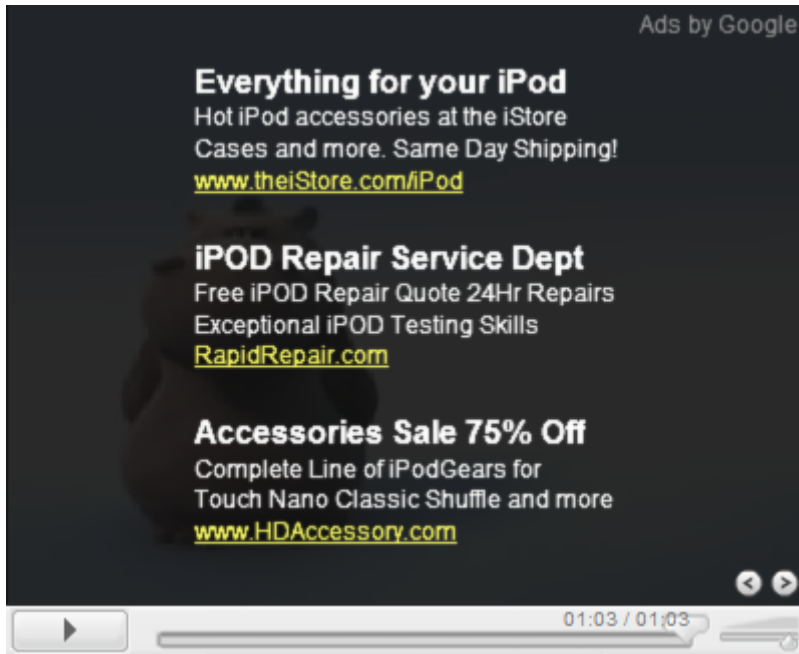
### FEATURES:

- Ad unit occupies the full width and height of the video player display area
- Ad unit supports pre-, mid-, and post-roll video ads up to 30-seconds in duration
- Yellow bar and countdown timer at top of player indicate progress through ad
- If the user clicks on the ad, the advertiser’s website opens in a new window

## Fullscreen Endcap Advertisements:

To receive fullscreen endcap ads, a single ad request for the general ad type “fullscreen” should be made. Fullscreen text and image endcap ads compete directly against each other in the ad auction, and the AFV system will automatically deliver the ad that will monetize the best. Fullscreen endcap ads can be daisy chained with other ads (see Appendix D for more info).

This screenshot shows the user experience for a **fullscreen text endcap ad**:



### FEATURES:

- Ad unit occupies the full width and height of the video player display area
- If the user clicks on an ad, the advertiser's website opens in a new window
- User may scroll forward or backward through several pages of ads

This screenshot shows the user experience for a **fullscreen image endcap ad**:



### FEATURES:

- Ad unit occupies the full width and height of the video player display area
- Ad unit supports static image ads, animated Flash ads, and click-to-play video ads
- If the user clicks on an ad, the advertiser's website opens in a new window

## Overlay Advertisements:

To receive text overlay and InVideo graphical overlay ads, a single ad request for the general ad type “overlay” should be made. Text overlay ads and InVideo graphical overlay ads compete directly against each other in the ad auction, and the AFV system will automatically deliver the ad that will monetize the best. Overlay ads can be daisy chained with other ads (see Appendix D for more info).

Text overlay and InVideo overlay ad demos: <http://www.google.com/ads/videoadsolutions/demos.html>

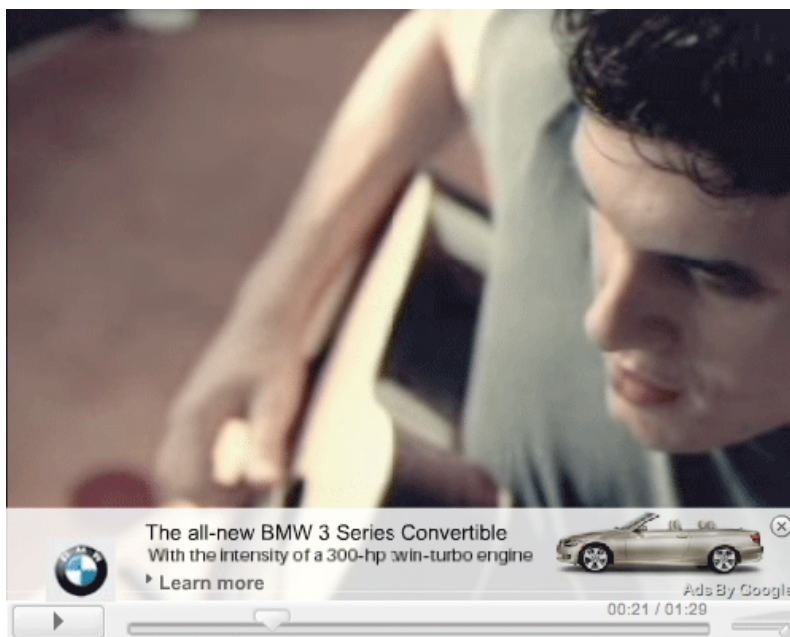
The screenshot below shows the user experience for a **text overlay ad**:



### TEXT OVERLAY AD FEATURES:

- Ad unit occupies the bottom 20% of the display area
- Text ads rotate after 20 seconds of no user interaction
- Ad unit minimizes to a small headline after 60 seconds of no user interaction
- User may close the ad unit at any time and a small “Ad” pill will be displayed in the corner
- If the user clicks on an ad, the advertiser’s website opens in a new window
- User may scroll forward or backward through the ads

This screenshot shows the user experience for an **InVideo graphical overlay ad**:



### FEATURES:

- Ad unit occupies the bottom 20% of the display area
- Animated Flash banner can be accompanied by an optional video creative
- Ad unit minimizes after 15 seconds of no user interaction and a small “Ad” pill is displayed in the corner
- User may close the ad unit at any time and a small “Ad” pill will be displayed in the corner
- If the user clicks on the graphical overlay, an optional video creative is displayed (or the advertiser’s website opens in a new window)
- If the user clicks on the optional video creative, the advertiser’s website opens in a new window

## CHAPTER 2: POLICIES

Publishers must adhere to the [AFV program policies](#), the [AFV terms and conditions](#), and the [standard AdSense program policies](#). For your convenience, the AFV program policies are reproduced in this chapter. Links to all of these documents may be found at the end of this chapter.

### AdSense for Video Program Policies

At this time, AdSense for video may only be implemented for pages and videos that are in English and that receive primarily U.S. traffic.

#### PLAYER REQUIREMENTS

To request and display Google ads, video players must fulfill the following basic requirements:

- Players must be Flash version 7 or above.
- Players must always maintain an aspect ratio of 4:3 or 16:9.
- Players must have a playable area no smaller than 320 x 240 pixels in size.

#### VIDEO REQUIREMENTS

- **Content Length:** Content clips must adhere to the following minimum length requirements:
  - Overlay ads: The content clip must be 30 seconds in length or longer to show overlay ads.
  - Fullscreen endcap ads: The content clip must be 15 seconds in length or longer to show fullscreen endcap ads.
  - Video ads: The content clip must be 1 minute in length or longer to show a single video ad. To show more than 1 video ad in a content clip, there must be 2 additional minutes of content for each additional video ad.
- **Video Content:** Publishers using AdSense for video may only place Google ads in videos that adhere to our content guidelines. Some of these guidelines are highlighted below. Please see the AdSense program policies for the complete list of content guidelines. Please note that all pages with video players using AdSense for video must also comply with these content guidelines. If your video player is embeddable, you must make every effort possible to ensure that pages where the video will display are also in compliance with the program policies. Videos displaying Google ads may not include:
  - Pornography, adult, or mature content
  - Any other content that is illegal, promotes illegal activity, or infringes on the legal rights of others
  - Violent content, racial intolerance, or advocacy against any individual, group, or organization
  - Content protected by copyright law, unless the content provider has the necessary legal rights to display that content. Please see our DMCA policy for more information.
- **Video Descriptions and Metadata:**
  - High quality and accurate metadata, via an RSS feed, or a video description URL must be provided and maintained for all videos monetized through AdSense for video.
  - Each video file (FLV) must have its own unique and static URL.
  - All parameters passed to Google at the time of the ad request must be accurate.

#### AD AND VIDEO PLAYER BEHAVIOR

Code provided to you through the AdSense program may not be altered, nor may the standard ad behavior, targeting, or delivery of ads be manipulated, in any way that is not explicitly permitted by Google.

- If an ad request is made and returned, the ad must always be displayed, unless a user abandons the video before the ad is displayed.
- Ads may not be hard-coded into your video player or video stream.
- Ad banners and images may not be used as the template background for your video player.
- Your video player must accommodate all appropriate types of user interaction with Google ads. For instance, all clickable links within an ad must remain clickable.

- Publishers may not obscure, hide, or remove any elements of the ad unit under any circumstances. Conversely, ads must not be placed in close proximity to or obstruct in any way your video player controls (play, pause, volume, etc.).
- If your video player automatically plays when the page is loaded, the player needs to be placed above the fold of the page (must be within 800x600 pixels from the top left corner of the page).

## ADS SERVED

AdSense for video offers several ad formats and we encourage publishers to experiment with a variety of placements in your video player, provided that you respect the policies listed below. Please note that these policies also apply to any third party ads you choose to display in players with Google ads.

### Ad Mixing Guidelines:

Publishers may mix AFV ads and ads from another network in the same stream, with the following restrictions:

- Only one ad may be displayed within the video player at any given time.
- Once an AFV ad has been requested, it must be displayed until:
  - The ad itself finishes playing
  - The content video ends
  - The user skips or closes the ad
  - The user navigates away from the video

### Ad Format-Specific Guidelines:

- Overlay ads:
  - Overlay ads may start at any time in the video stream, though beginning at ten seconds is recommended. The ad must have at least 20 seconds of display time.
- Fullscreen endcap ads:
  - Fullscreen endcap ads must be the last item shown at the end of a video and must persist in the player until the user navigates away from the video.
- Video ads:
  - Pre-roll and mid-roll video ads must persist in the player until the ad finishes playing or the user clicks the skip button.
  - Post-roll video ads must be the last item shown at the end of a video and must persist in the player until the ad finishes playing, the user clicks the skip button, or the user navigates away from the video.

## Policy Links:

**AFV PROGRAM POLICIES:** <https://www.google.com/adsense/support/bin/answer.py?answer=73987>

**AFV TERMS AND CONDITIONS:** [https://www.google.com/adsense/static/en\\_US/AvTerms.html](https://www.google.com/adsense/static/en_US/AvTerms.html)

**ADSENSE PROGRAM POLICIES:** <https://www.google.com/adsense/support/bin/answer.py?answer=48182>

## CHAPTER 3: TARGETING

### Video Ads, InVideo Graphical Overlay Ads, and Fullscreen Image Endcap Ads

Video ads, InVideo graphical overlay ads, and fullscreen image ads are specifically targeted by advertisers to a particular site, or to particular segments of a site's video content called channels. Note: channels are NOT used for contextually targeting text ads.

In order for advertisers to target segments of your content, you will need to create targetable channels in your AFV account. Please refer to Appendix C for step-by-step channel creation instructions.

### Text Overlay Ads and Fullscreen Text Endcap Ads

The AFV system can contextually target text ads using three methods: videoDescriptionUrl, page content, and media RSS feeds. The three targeting signals are complimentary, and will be used in combination to target relevant ads, with emphasis given to the best-performing targeting method. Note: we must receive at least one of the three targeting signals in order to deliver AFV ads.

#### METHOD 1 – VideoDescriptionUrl:

- The primary method for contextually targeting text ads is via a videoDescriptionUrl that is passed to the AFV system in the ad request.
- The videoDescriptionUrl should be the URL of any HTML page that contains descriptive information (e.g., title, description, categories) for a single video.
- When we receive an ad request, we will crawl the videoDescriptionUrl page, store information about the content, and use that data as a targeting signal for subsequent ad requests.

#### METHOD 2 – Page content:

- Text ads can also be contextually targeted using the content of the page making the ad request.
- The AFV system will automatically detect the URL of the page making the ad request as long as `allowScriptAccess` is set to "always." This setting is mandatory.
- When we receive an ad request, we will crawl the page making the ad request, store information about the content, and use that data as a targeting signal for subsequent ad requests.

#### METHOD 3 – Media RSS feeds:

- Lastly, text ads can be contextually targeted using metadata passed in via a media RSS feed.
- This option is available if you have a significant amount of metadata available over and above what is already included on the videoDescriptionUrl page (e.g., video transcripts).
- To use this targeting method, you must generate a media RSS feed that conforms to the Google AdWords Video XML metadata specification. We will manually import the feed into our Sitemaps system, crawl it, and store metadata about each video.
- When we receive an ad request, we will match the publisher ID and video ID in the ad request with the publisher ID and video ID we stored from the media RSS crawl.
- If you are interested in using this feature, please contact your AFV Technical Account Manager.

## CHAPTER 4: FLASH INTEGRATION

### Important Note on Testing Procedure

Prior to launching AFV on your site, you **MUST** set up a stand-alone test page for review by Google. The test page should not be generally accessible to your users. Your TAM will review the functionality of your overlay ad, fullscreen endcap ad, or video ad implementation to ensure that the ads are being rendered and targeted properly. Once the review process has been completed, you will then be able to launch AFV on your site.

### Testing in the Flash IDE

#### Security sandbox violation error:

When testing in the Flash IDE, you may see a “security sandbox violation” error in the output console. This error is expected, and occurs because Flash movies are not permitted to access data (e.g., from the AFV servers) that resides outside the exact web domain from which the SWF originated. In most cases, this error should not prevent you from testing your AFV implementation inside the Flash IDE. However, if you have difficulty getting the ads to display, please try using the “Publish to HTML” option in the Flash IDE. If you choose the “Publish to HTML” option, please note you may need to modify the `allowScriptAccess` setting to “always” (the default value of “sameDomain” may cause a security popup box to appear).

### Pre-Ad Request Set Up

#### Calling `allowDomain`:

When loaded into another Flash application, the Flash ad library SWF must be set to allow your video player to access the ads being served by Google. To do this, you must include the `allowDomain` call. See *Example 4.1a for AS2* and *Example 4.1b for AS3*.

#### Loading the AFV SWF library:

The AFV SWF library provides a mechanism for fetching and playing in-stream ads. In order to access the AFV-related methods, you will need to load either the AS2 or AS3 AFV SWF library into your Flash player. See *Example 4.1a for AS2* and *Example 4.1b for AS3*.

#### *Example 4.1a – AS2 setup:*

```
public function go(target:MovieClip):Void {
    System.security.allowDomain("http://pagead2.googlesyndication.com");
    var swfClip:MovieClip = target.createEmptyMovieClip(
        "swfClip", target.getNextHighestDepth());

    var mclListener:Object = new Object();
    mclListener.onLoadInit = delegate(this, sendAdRequest);

    var mcl:MovieClipLoader = new MovieClipLoader();
    mcl.addListener(mclListener);

    mcl.loadClip(
        "http://pagead2.googlesyndication.com/pagead/googlevideoadslibrary.swf",
        swfClip);
}

// Helper function
public static function delegate(scope:Object, handler:Function):Function {
    var fn:Function = function() {
        return handler.apply(scope, arguments);
    }
    return fn;
}
```

**Example 4.1b – AS3 setup:**

```

public function go(target:MovieClip):void {
    Security.allowDomain("pagead2.googlesyndication.com");

    // Prepare to load Google SWF
    var request:URLRequest = new URLRequest("http://pagead2.googlesyndication.com/" +
        "pagead/scache/googlevideoadslibraryas3.swf");
    var loader:Loader = new Loader();
    loader.contentLoaderInfo.addEventListener(Event.COMPLETE, sendAdRequest);

    // load Google SWF
    loader.load(request);
    addChild(loader);
}

// Helper function
public static function delegate(scope:Object, handler:Function):Function {
    var fn:Function = function() {
        return handler.apply(scope, arguments);
    }
    return fn;
}

```

**Ad Request Parameters****Setting up the ad request parameters object:**

Prior to making an ad request, you will need to set up an object containing several required parameters, including your publisher ID, videoDescriptionUrl, and the type of ad you are requesting. See *Examples 4.2a for AS2 and 4.2b for AS3*.

**Required ad request parameters for ALL ad types:**

Parameter name	Description
videoid	The publisher's unique identifier for each video.
videoPublisherId	The publisher's AFV client ID (provided by AdSense).
videoFlvUrl	The full URL where the video's .flv file is located.
videoDescriptionUrl	The full URL of an HTML page that describes the video.
channels	An array of strings representing AdSense targeting/reporting channel IDs. Up to 5 channels may be passed in per ad request. See Appendix C for more information on creating channels.
adType	The type of ad being requested. The following ad types may be requested: video (pre-/mid-/post-roll video ads) fullscreen (text and image fullscreen endcap ads) overlay (both text overlays and InVideo graphical overlays) text_overlay (standard text overlays only) graphical_overlay (standard InVideo graphical overlays only)
pubWidth	Specifies the width of the playable area in the publisher's Flash video player.
pubHeight	Specifies the height of the playable area in the publisher's Flash video player.

**Additional required ad request parameters for video ads:**

Parameter name	Description
adTimePosition	Specifies when in the video stream the video ad will be displayed. Set to "0" for pre-roll ads, "1" for mid-roll video ads, and "-1" for post-roll video ads. Defaults to "-1" (post-roll) if not set. The following behaviors will occur for each setting: 0 (pre-roll):     Calls resumeContentVideo() upon ad completion

- 1 (mid-roll): Calls `pauseContentVideo()` when ad starts playing  
Calls `resumeContentVideo()` upon ad completion
- 1 (post-roll)\*: Calls neither `pauseContentVideo()` nor `resumeContentVideo()`

\*Note: If you use video ads between videos in a playlist environment, you MUST use the "-1" (post-roll) setting.

<code>maxTotalAdDuration</code>	Specifies the maximum video ad length that should be returned. Must be specified in <u>milliseconds</u> . Supported values are "15000" (15 seconds) or "30000" (30 seconds).
---------------------------------	--

### Optional ad request parameters for **ALL** ad types:

Parameter name	Description
<code>adtest</code>	Specifies whether this is a test implementation or not. Possible values are "on" or "off". If set to "on," impressions and clicks will not be recorded in your account. This parameter MUST be used during testing. Defaults to "off" if not set.
<code>adsafe</code>	Specifies the setting that should be used to filter ads that display adult or pornographic content. Possible values are "high" (all pornographic and adult ads are filtered) or "medium" (pornographic ads are still filtered, but other adult ads, such as ads for gambling or prescription drugs, are returned). Defaults to "high" if not set.

### Ad request execution set up:

In order to execute an AFV ad request, you must pass in the object containing the required ad request parameters, as well as specify a callback method (`onAdsRequestResult`) that will be executed upon completion of the ad request. See *Example 4.2a* for AS2 and *Example 4.2b* for AS3.

*Example 4.2a – AS2 ad request (parameters highlighted in yellow must be customized):*

```
private function sendAdRequest(clip:MovieClip) {
    // Create request params object
    var request:Object = new Object();
    request.videoId = "78262";
    request.videoPublisherId = "ca-video-pub-0000000000000000";
    request.videoFlvUrl = "http://media2.example.com/videos/78262.flv";
    request.videoDescriptionUrl = "http://media2.example.com/videos/78262.html";
    request.channels = ["1234567890", "9876543210"]; // Must be an array of strings
    request.pubWidth = "320";
    request.pubHeight = "240";
    request.adType = "fullscreen";

    // Fetch an ad, specify callback method
    clip.requestAds(request, delegate(this, onAdsRequestResult));
}
```

*Example 4.2b – AS3 ad request (parameters highlighted in yellow must be customized):*

```
private function sendAdRequest(event:Event) {
    var googleAds:Object = event.target.content;

    // Create request params object
    var request:Object = new Object();
    request.videoId = "78262";
    request.videoPublisherId = "ca-video-pub-0000000000000000";
    request.videoFlvUrl = "http://media2.example.com/videos/78262.flv";
    request.videoDescriptionUrl = "http://media2.example.com/videos/78262.html";
    request.channels = ["1234567890", "9876543210"]; // Must be an array of strings
    request.pubWidth = "320";
    request.pubHeight = "240";
    request.adType = "fullscreen";

    // Fetch an ad, specify callback method
    googleAds.requestAds(request, onAdsRequestResult);
}
```

## Ad Request Callback Object

Once you have executed the ad request, the system will return a callback object. If the ad request was successful, you will be able to pull a MovieClip from the callback object that contains the ads. If the ad request failed, the object's `errorMsg` parameter will indicate possible errors. See *Example 4.3*.

The callback object contains the following parameters:

Parameter name	Description
success	A Boolean value indicating whether ads were successfully returned. "True" indicates ads were successfully returned. "False" indicates that an error occurred.
errorMsg	A string indicating possible errors in the ad request.

The callback object also contains the following methods:

Method name	Description
getAdPlayerMovieClip	Creates a video player MovieClip to play the in-stream ad. The MovieClip plays within the partner player.
getType	Returns a string that describes the type of ad returned. The following values will be returned: <u>Video ads:</u> video_fullscreen (pre-/mid-/post-roll video ads)  <u>Fullscreen ads:</u> text_fullscreen (standard fullscreen text ads) graphical_fullscreen (fullscreen static image or Flash ads) graphical_fullscreen_ctp (fullscreen click-to-play video ads)  <u>Overlay ads:</u> text_overlay (standard text overlay ads) graphical_overlay (graphical overlay ads WITHOUT a video creative) graphical_overlay_ctp (graphical overlay ads WITH a video creative)

## Setting up the Post-Ad Request Callback Method

You will also need to set up the `onAdsRequestResult` callback method that will be executed after the ad request is completed. The callback method should accept the callback object as an input. The callback method should retrieve the ad MovieClip, set its size, position it, load it, and then play it. Within `onAdsRequestResult`, you must also implement the `pauseContentVideo` callback method (to automatically pause your content video at the appropriate times) and the `resumeContentVideo` callback method (to automatically begin or resume playing your content video at the appropriate times).

**Required methods for ALL ad types:**

Method name	Description
setX	Specifies the X coordinate for positioning the ad overlay.
setY	Specifies the Y coordinate for positioning the ad overlay.
load	Loads the ad MovieClip, begins buffering it, and leaves it in a paused state. Playback will not start until <code>playAds()</code> has been called.
playAds	Begins playback of the ad MovieClip. Calling <code>playAds()</code> after playback has completed will restart playback from the beginning of the MovieClip.
pauseContentVideo	Called when playback of the content video should pause. This method is executed when a mid-roll video ad is shown, or when a text overlay or InVideo graphical overlay ad is clicked, so that the user does not miss any of the content video while viewing the advertisement.

resumeContentVideo	Called when playback of the content video should begin or resume. This method is executed when a pre- or mid-roll video ad ends, or after the inset video ad accompanying an InVideo graphical overlay ad ends or is closed by the user.
--------------------	--

### Optional methods for **ALL** ad types:

Method name	Description
setSize(X, Y)	Sizes the AFV ad unit according to the specified width (X) and height (Y) dimensions. The minimum size is 320x240. This method also returns a Boolean value. It will return "false" if an invalid size is specified and calling playAds() will have no effect. In all other cases, the Boolean value returned will be "true."
setReopenPill(position)	Positions the small "Ad" pill at either the bottom right-hand side or bottom left-hand side of the player. Possible values for the position parameter are: "right", "r", "left", or "l". Defaults to "right".
pause	Pauses playback of a video ad. Closes a fullscreen ad and causes the "Ad" pill to be displayed. Not applicable to overlay ads.
destroy	Destroys the ad MovieClip.
onDestroy	Called when the destroy() method (if used) has finished executing.
onStateChange	Called whenever a state change occurs in the player. Valid state strings for "oldState" and "newState" are "paused", "buffering", "playing", and "completed". See Appendix B for more information.
onAdSkip	Called when the user clicks the "X" or skip button to close or skip an ad.
onAdClick	Called when the user executes a click that opens the advertiser's website (e.g., a click on a pre-/mid-/post-roll video ad, a click on a fullscreen text or image endcap ad, a click on a graphical overlay's inset video ad, a click on a graphical overlay itself if it is not accompanied by an inset video ad, or a click on a text overlay ad).
onError	Called whenever an unrecoverable error occurs when loading or playing a video creative (e.g., a pre-/mid-/post-roll video ad, a click-to-play video ad shown in a fullscreen endcap ad slot, or a graphical overlay's inset video creative).

### Example 4.3 – AS2 and AS3 callback method (parameters highlighted in yellow must be customized):

```
private function onAdsRequestResult(callbackObj:Object):Void {
    trace("onAdsRequestResult: callbackObj.success = " + callbackObj.success);
    if (callbackObj.success) {
        var player:MovieClip = callbackObj.ads[0].getAdPlayerMovieClip();
        player.setSize(320, 240);
        player.setX(0);
        player.setY(0);
        player.load();
        player.playAds();
        player.pauseContentVideo = delegate(this, pauseStream);
        player.resumeContentVideo = delegate(this, resumeStream);
    }
    else {
        trace("Error: " + callbackObj.errorMessage);
    }
}
```

## Executing the AFV Ad Request

Lastly, you will need to call the appropriate method to execute the ad request. If you used the examples in this document, you will need to call the `go(this);` method (see *Example 4.1a and 4.1b*) at the point in your Flash code where you want the ads to display, such as when the video begins playing (overlays) or at the end of the video (fullscreen endcaps). If you wish to have an overlay ad display after a short delay (e.g., 10 seconds into the video), you will need to build that logic into your player yourself as well.

## CHAPTER FIVE: HTML

In order for AFV ads to be served on videos, allowScriptAccess **MUST** be set to “always.” This allows the AFV system to detect the URL of the page making the ad request. Ads will not be served on videos where we cannot detect the URL. We need to be able to detect the page URL so that we can report information to advertisers on where their ads are being displayed. The page URL is also used for policy review purposes and to improve contextual ad targeting.

### Standard HTML Setup (Firefox and IE Support)

If you are using standard HTML tags to embed your video player, you must include both the <object> and <embed> tags in order to properly set allowScriptAccess in both IE (using the <object> tag) and Firefox (using the <embed> tag).

Note: For the <object> tag, the “classid” and “id” attributes, and the “movie” param, are required. Additionally, allowScriptAccess must also be added to any code made available to users for embedding videos in blogs, etc. See Example 5.1.

Example 5.1 (parameters highlighted in yellow must be customized):

```
<object width="320" height="240" classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
  id="NeedThisParam">
  <param name="movie" value="http://www.example.com/v/?i=11149286"
    type="application/x-shockwave-flash"></param>
  <param name="allowScriptAccess" value="always"></param>
  <param name="wmode" value="transparent"></param>
  <embed src="http://www.example.com/v/?i=11149286"
    type="application/x-shockwave-flash"
    allowScriptAccess="always"
    width="320"
    height="240">
  </embed>
</object>
```

### SWFObject Setup (Firefox and IE Support)

If you are using SWFObject.js to embed your video player, you must use the “addParam” method to set allowScriptAccess to “always.” See Example 5.2. Please refer to <http://blog.deconcept.com/swfobject/> for documentation on using SWFObject.

Example 5.2 (parameters highlighted in yellow must be customized):

```
<script type="text/javascript" src="swfobject.js"></script>
<div id="flashcontent">
  This text is replaced by the Flash movie.
</div>
<script type="text/javascript">
  var so = new SWFObject("player.swf", "mymovie", "320", "240", "8", "#336699");
  so.addParam("allowScriptAccess", "always");
  so.write("flashcontent");
</script>
```

### Embed Code Provided to Users

If you allow users to embed your video player in third party blogs or websites, and you wish to display AFV ads on those videos, allowScriptAccess support (for both Firefox and IE, as described above) **MUST** be added to the embed code you provide to users.

## CHAPTER 6: TESTING

Congratulations! You have completed the code portion of the AFV integration. The last step is to verify the functionality of your ads. Be sure to send your TAM a link to your stand-alone test page for review, prior to launching AFV on your site.

### Video Ads

To test pre-, mid-, or post-roll video ads, you will need to pass in the “ca-video-afvtest” test publisher ID in the ad request. A sample video ad has been targeted to that specific publisher ID. Please make sure the test publisher ID is passed in ONLY on the test page and not on your production site. It is critical that the sample video ad never be shown to live users. See *Example 6.1*.

*Example 6.1:*

```
request.videoPublisherId = "ca-video-afvtest";
```

#### Expected behavior for pre-roll video ads:

- Video ad is returned and rendered correctly
- When the video ad has finished playing, the content video automatically begins playback
- If the video ad is clicked, the advertiser’s website opens in a new browser window or tab, and the video ad is no longer displayed in the player

#### Expected behavior for mid-roll video ads:

- Video ad is returned and rendered correctly
- When the video ad is displayed, the content video is automatically paused
- When the video ad has finished playing, the content video automatically resumes playback
- If the video ad is clicked, the advertiser’s website opens in a new browser window or tab, and the video ad is no longer displayed in the player

#### Expected behavior for post-roll video ads:

- Video ad is returned and rendered correctly
- When the video ad has finished playing, the ad is destroyed
- If the video ad is clicked, the advertiser’s website opens in a new browser window or tab, and the video ad is no longer displayed in the player

### Fullscreen Endcap Ads

If you have successfully completed the fullscreen endcap ad integration process, fullscreen endcap ads should now be appearing in your Flash video player.

#### Expected behavior for fullscreen endcap ads:

- Text or image ad is returned and rendered correctly
- Text or image ad persists on the screen and is not closed
- If the ad is clicked, the advertiser’s website opens in a new browser window or tab

## Overlay Ads

If you have successfully completed the overlay ad integration process, text ads should now be appearing in your Flash video player.

### Expected behavior for text overlay ads:

- Text ads are returned and rendered correctly
- Text ads rotate after 20 seconds of no user interaction
- Text ads minimize down to a small headline after 60 seconds of no user interaction
- Text ads can be manually minimized with the "X" button and the "Ad" pill is displayed
- Minimized text ads can be resurrected by clicking the "Ad" pill
- If a text ad is clicked, the content video is automatically paused
- If a text ad is clicked, the advertiser's website opens in a new browser window or tab

To test InVideo graphical overlay ads, you will need to pass in the "ca-video-afvtest" test publisher ID in the ad request. A sample InVideo graphical overlay ad has been targeted to that specific publisher ID. Please make sure the test publisher ID is passed in ONLY on the test page and not on your production site. It is critical that the sample InVideo graphical overlay ad never be shown to live users. See *Example 6.2*.

*Example 6.2:*

```
request.videoPublisherId = "ca-video-afvtest";
```

### Expected behavior for graphical overlay ads:

- Graphical overlay ad animation plays correctly
- Graphical overlay ad minimizes after 15 seconds of no user interaction and the "Ad" pill is displayed
- Graphical overlay ad can be manually minimized with the "X" button and the "Ad" pill is displayed
- Minimized graphical overlay ad can be resurrected by clicking the "Ad" pill
- If the graphical overlay ad is clicked, the content video is automatically paused
- If the graphical overlay ad is clicked, an inset video ad plays
- If the inset video ad is clicked, the advertiser's website opens in a new browser window or tab
- When the inset video ad finishes playing or is closed using the "X" button, the content video automatically resumes playing and the "Ad" pill is displayed

# APPENDIX A: TROUBLESHOOTING

## Troubleshooting Tools

Taking a look at the AFV ad request can help you quickly troubleshoot many common integration problems. In order to view the ad request, you will need a tool that allows you to see the HTTP header communication taking place between your browser and the server.

### Suggested Tools:

- Firefox – LiveHTTPHeaders (<http://livehttpheaders.mozdev.org/>)
- IE – Fiddler (<http://www.fiddlertool.com/fiddler/>)

### Viewing the AFV Ad Request:

Once you have installed an HTTP header tool, you will need to check the HTTP headers for a request to Google's "pagead2.googleadsyndication.com" server.

The most important parameters you will see in the AFV ad request URL are:

Ad request URL parameter:	Corresponding Flash ad request parameter:
video_url_to_fetch	videoDescriptionUrl
url	[automatically detected]
client	videoPublisherId
ad_type	adType
channel	channels
video_doc_id	videoId
max_ad_duration	maxTotalAdDuration
videoad_start_delay	adTimePosition

## Ad Rendering Issues

### ADS ARE NOT APPEARING AT ALL

#### Check to see if ad request is being executed:

AFV ads will only be displayed if the ad request is properly executed.

- Check the HTTP headers to make sure that an AFV ad request is actually being made.

#### Check for missing targeting signals:

The AFV system must receive at least one of the three targeting signals (videoDescriptionUrl, page URL, media RSS feed) in order to deliver ads.

- Check the ad request to make sure that the "video\_url\_to\_fetch" (videoDescriptionUrl) and "url" (page URL) parameters are not null.
  - If "video\_url\_to\_fetch" is null, check your Actionscript code to make sure the videoDescriptionUrl parameter is being passed in correctly. Note: all ad request parameters are case-sensitive.
  - If "url" is null, check your HTML embed code to make sure `allowScriptAccess` is set to "always" in manner supported by both Firefox and IE (see Chapter 5: HTML for more information).
- If you are using a media RSS feed, check to make sure that the "gm:adclient" (publisher ID) tag and the "guid" (video ID) tag in the feed match the "videoPublisherId" and "videoId" you are passing in via the ad request.

**Check for missing ad request parameters:**

The AFV system must receive all required ad request parameters in order to deliver ads.

- Check your Actionscript code to make sure all of the following are being passed in:
  - `videoid`
  - `videoPublisherId`
  - `videoHostingSite`
  - `videoFlvUrl`
  - `videoDescriptionUrl`
  - `adType`

**Check for missing calls to `load()` and `playAds()`:**

In order for an AFV ad to be displayed, you must call `load()` and `playAds()` on the ad MovieClip.

- Check your Actionscript code to make sure that the `load()` and `playAds()` methods are being called.

**Check for missing call to `setSize()` method:**

If the `setSize()` method is not called, ads will not be displayed.

- Check your Actionscript code to make sure that the `setSize()` method is being called with the correct dimensions of your player.

**Check for adult or negative/sensitive content:**

If the AFV system flags a video as containing adult or negative/sensitive content, ads will not be delivered.

- Check the video in question to make sure it is not adult or negative/sensitive in nature. If it is not, contact your TAM for further investigation.

**ADS ARE APPEARING IN FIREFOX BUT NOT IE****Check `allowScriptAccess` settings:**

In order for AFV ads to be fetched and displayed, `allowScriptAccess` must be set to "always".

- Check your HTML embed code to make sure that `allowScriptAccess` is set to "always" in both the `<embed>` and `<object>` tags. See Chapter 5: HTML for more information.

**SAMPLE GRAPHICAL OVERLAY AD IS NOT APPEARING****Check that correct publisher ID is being passed in:**

The sample graphical overlay ad is targeted to the test publisher ID "ca-video-afvtest". If this parameter is not being passed in properly, the graphical overlay ad will not be displayed.

- Check the ad request to be sure that you are passing in the correct publisher ID.

**ADS ARE NOT APPEARING IN PLAYER FULLSCREEN MODE****Check for missing call to `setSize()` method:**

If your player window is resized after you initially call `setSize()` on the ad MovieClip, the ads will not be scaled to the new player window size and thus may not be visible.

- Check your Actionscript code to make sure that the `setSize()` method is being called again with the new dimensions of your player after the user resizes the player window.

## ADS ARE REMAINING VISIBLE FOR MORE/LESS THAN THE EXPECTED DURATION

### Ad behavior is based on periods of no user interaction:

Ad behavior timers, which control events such as when text overlay ads rotate/minimize/close, are based on periods of no user interaction. Any interaction between the user and the AFV ad unit (e.g., mousing over the ad) will restart the ad behavior timer.

### Flash creative framerate requirement:

Advertisers are required to design their InVideo graphical overlays to run at 25 FPS. If a publisher's video runs at a different framerate, the ad may be displayed for a longer or shorter than expected duration.

### User interface experiments:

In order to optimize user experience, advertiser value, and publisher revenue, Google may at times test minor modifications to the appearance or behavior of a small percentage of AFV ads.

## Ad Targeting Issues

## ADS ARE NOT RELEVANT

### Check for missing targeting signals:

The AFV system must receive at least one of the three targeting signals (videoDescriptionUrl, page URL, media RSS feed) in order to deliver ads.

- Check the ad request to make sure that the "video\_url\_to\_fetch" (videoDescriptionUrl) and "url" (page URL) parameters are not null.
  - If "video\_url\_to\_fetch" is null, check your Actionscript code to make sure the videoDescriptionUrl parameter is being passed in correctly. Note: all ad request parameters are case-sensitive.
  - If "url" is null, check your HTML embed code to make sure `allowScriptAccess` is set to "always" in manner supported by both Firefox and IE (see Chapter 5: HTML for more information).
- If you are using a media RSS feed, check to make sure that the "gm:adclient" (publisher ID) tag and the "guid" (video ID) tag in the feed match the "videoPublisherId" and "videoID" you are passing in via the ad request.

### Check for adequate metadata:

In order for the AFV system to deliver the most contextually relevant ads, there must be adequate text-based metadata available via one or more of the three targeting signals described above.

- Check your videoDescriptionUrl page, the page making the ad request, and your media RSS feed (if applicable) to ensure that as much descriptive text as possible is available. The less metadata there is, the less information the AFV system has for contextually targeting ads.
- Add additional descriptive text to the page or media RSS feed (e.g., transcripts, categories, extended descriptive text, etc.) if possible. The descriptive text must be visible on the page, and cannot be hidden within HTML comments or metatags, or the crawler will not detect it.

### Check for CPM text ads:

In addition to contextually targeted CPC text ads, you may also receive site targeted CPM text ads. Since CPM text ads are targeted by advertisers to your specific site, they may not appear to be contextually relevant. Note: CPM text ads compete directly against CPC text ads in the ad auction, and the system will deliver whichever type it calculates will pay you more.

- Use the LiveHTTPHeader Firefox extension and locate the AFV ad request.
- Copy and paste the entire ad request URL into a browser. An XML version of the ads will be displayed in your browser.
- Check whether the targeting type of the ad is "CPM" or "CPC".

## Pause/ResumeContentVideo Issues

### PAUSECONTENTVIDEO / RESUMECONTENTVIDEO ARE NOT BEING CALLED

#### Check to make sure the callback methods are set at the proper context:

In order for the AFV SWF to access your `pauseContentVideo` and `resumeContentVideo` methods, it may be necessary to use a delegate method to run them within the proper scope.

- Try using the delegate helper function in Example 4.1 to execute `pauseContentVideo` and `resumeContentVideo` within the “this” scope. See Example 4.3 for additional info.

## Channels Issues

### VIDEO PLAYBACKS ARE NOT BEING RECORDED IN THE ADSENSE FRONT END

#### Check that correct publisher ID and channel ID are being passed in:

In order for AFV stats to be properly recorded in your account, you must pass in the correct publisher ID in the ad request. Additionally, in order for stats to be recorded under a particular channel, the correct channel ID must also be passed in via the ad request.

- Check the ad request to be sure that you are passing in the correct publisher ID.
  - Your publisher ID must be in the format “ca-video-pub-0000000000000000”.
- Check the ad request to be sure the “channel” parameter is not blank or undefined.
  - If the “channel” parameter is undefined, check your Actionscript code to make sure the channels are being passed in as an array, not as a string, even if it is only a single value.
    - Example: `request.channels = ["1234567890"];`
    - Example: `request.channels = ["1234567890", "5678901234"];`
  - If the “channel” parameter is blank, check your Actionscript code to make sure that the “channel” parameter is included among your ad request parameters.

## APPENDIX B: USING ONSTATECHANGE()

The `onStateChange()` method is called whenever a state change occurs in the player, whether requested by the client or not. One common usage is the situation where a publisher wishes to disable their video player controls when the video creative that accompanies an InVideo graphical overlay is playing, to help avoid user confusion. See *Example B.1*.

State:	State occurs when:
buffering	<ul style="list-style-type: none"> <li>- Video creative is in the process of being buffered.</li> <li>- Applicable to: Video ads and InVideo graphical overlays.</li> </ul>
paused	<ul style="list-style-type: none"> <li>- Video creative has been loaded and buffered, but video creative has not begun playback yet.</li> <li>- Applicable to: Video ads and InVideo graphical overlays.</li> </ul>
playing	<ul style="list-style-type: none"> <li>- Video ad begins playback.</li> <li>- Video creative that accompanies a graphical overlay begins playback.</li> <li>- Fullscreen endcap begins playback.</li> <li>- Text overlay begins playback.</li> <li>- Applicable to: Video ads, text overlays, InVideo graphical overlays, fullscreen endcaps.</li> </ul>
completed	<ul style="list-style-type: none"> <li>- Video creative finishes playing on its own.</li> <li>- Video creative is manually closed or skipped via the "X" button.</li> <li>- Video creative is automatically closed when user clicks on the creative, opening the advertiser's website in a new window.</li> <li>- Applicable to: Video ads and InVideo graphical overlays.</li> </ul>

### Example B.1:

```
// Determine the type of ad that was returned
var adType:String = callbackObj.ads[0].getType();

// Disable video player controls when a video creative is playing
// and re-enable once the creative has completed playing
player.onStateChange = function(oldState:String, newState:String) {
    if (adType == "video_fullscreen" && newState == "playing") {
        // Insert code for disabling video player controls here.
    }
    else if (adType == "video_fullscreen" && newState == "completed") {
        // Insert code for re-enabling video player controls here.
    }
}
```

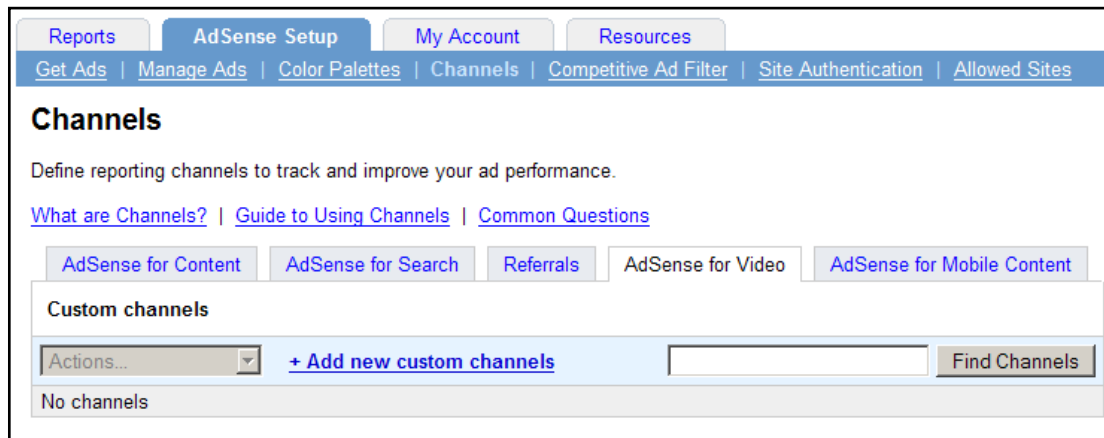
## APPENDIX C: CHANNELS

AdSense for Video provides two different types of channels: reporting channels (used simply for tracking performance) and targetable channels (which allow advertisers to target specific segments of your content).

### Navigating to the Channel Creation Page

To begin creating channels in your AFV account, please complete the following steps:

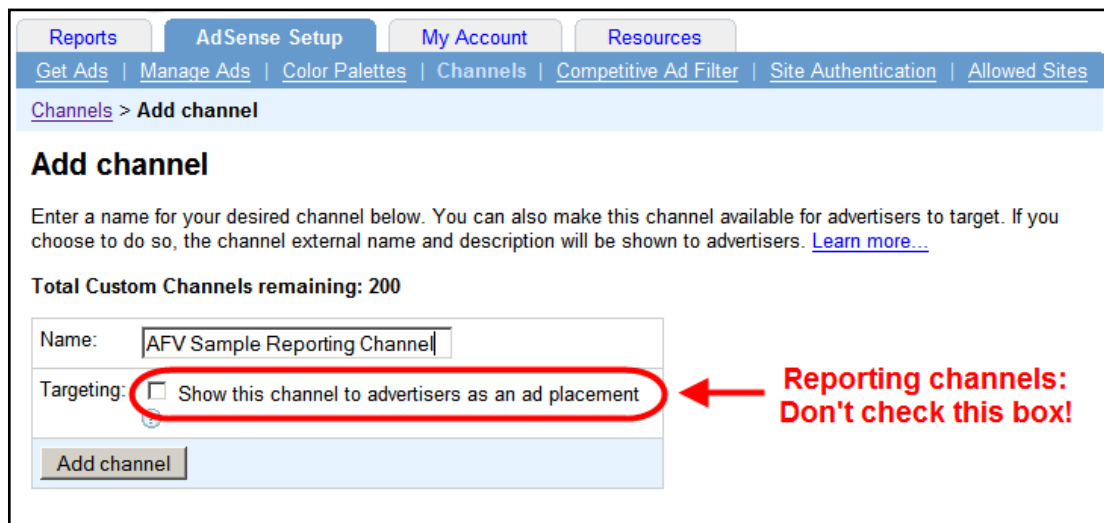
- Log into your AFV account at <http://www.google.com/adsense>
- Click on the “AdSense Setup” tab and then the “Channels” sub-tab.
- Click on the “AdSense for Video” product tab.
- Click on the “Add new custom channels” link.



### Creating a Reporting Channel

To create a reporting channel, please complete the following steps:

- Enter a name for the channel.
- DO NOT check the “Show this channel to advertisers as an ad placement” box!
- Click the “Add channel” button.



## Creating a Targetable Channel

To create a targetable channel, please complete the following steps:

- Enter a name for the channel.
- Check the “Show this channel to advertisers as an ad placement” box. Two additional fields will be displayed below.
- Enter a name in the “External name” box. This name will be shown to advertisers. We recommend including your site name and a brief description of the content, such as “MySite – Entertainment”.
- Enter a description in the “Description” box. This description will be shown to advertisers. We recommend including a brief summary of the type of content that will be tagged with this channel. Demographic information about your viewers and an example URL can also be included here.
- Click the “Add channel” button.

The screenshot shows the AdSense 'Add channel' interface. At the top, there are navigation tabs: Reports, AdSense Setup (selected), My Account, and Resources. Below these are links: Get Ads, Manage Ads, Color Palettes, Channels (selected), Competitive Ad Filter, Site Authentication, and Allowed Sites. The main heading is 'Channels > Add channel'.

**Add channel**

Enter a name for your desired channel below. You can also make this channel available for advertisers to target. If you choose to do so, the channel external name and description will be shown to advertisers. [Learn more...](#)

**Total Custom Channels remaining: 200**

Name:

Targeting:  Show this channel to advertisers as an ad placement ?

These fields will be shown to advertisers

External name:

Description:

A red circle highlights the checkbox, and a red arrow points to it from the text: **Targetable channels: DO check this box!**

## Passing in the Channel ID in Your Ad Request

- On the “Channels” screen, note the 10-digit ID number of the channels you just created.

The screenshot shows the AdSense 'Channels' management page. At the top, there are navigation tabs for 'Reports', 'AdSense Setup', 'My Account', and 'Resources'. Below these are links for 'Get Ads', 'Manage Ads', 'Color Palettes', 'Channels', 'Competitive Ad Filter', 'Site Authentication', and 'Allowed Sites'. The main heading is 'Channels', with a sub-heading 'Define reporting channels to track and improve your ad performance.' There are also links for 'What are Channels?', 'Guide to Using Channels', and 'Common Questions'. Below this, there are tabs for 'AdSense for Content', 'AdSense for Search', 'Referrals', and 'AdSense for Video'. The 'Custom channels' section contains a table with columns: Name, ID, Description, and Targetable. The 'ID' column for the 'MySite - Entertainment' channel is circled in red.

Name	ID	Description	Targetable
AFV Sample Reporting Channel	2335156104		
MySite - Entertainment	0168629492	Entertainment content on MySite.com	Yes

- In your ad request parameters object, pass in the 10-digit ID number of each channel you wish to use. See *Example C.1*. You may pass in up to 5 channels per ad request. Note: you must use an array for the channels parameter, even if you are only passing in a single channel ID.

*Example C.1 (parameters highlighted in yellow must be customized):*

```
request.channels = ["2335156104", "0168629492"];
```

## APPENDIX D: DAISY CHAINING

This appendix contains instructions for inserting AFV ad requests into your daisy chain of ad networks.

We recommend that you set up your daisy chain of ad networks in the following order:

Pre-roll video ads:	Post-roll video ads:	Overlay ads:	Fullscreen endcap ads:
1. Directly sold inventory	1. Directly sold inventory	1. Directly sold inventory	1. AFV endcaps
2. AFV pre-roll video ads	2. AFV post-roll video ads	2. AFV graphical overlays	2. Non-Google endcaps
3. Non-Google video ads	3. AFV endcaps	3. Non-Google overlays	
	4. Non-Google post-roll video ads or endcaps	4. AFV text overlays	

Using AFV daisy chaining will allow you to:

- Maximize your revenue by controlling the placement of AFV in your sequence of ad calls
- Monetize all unsold inventory by falling back to AFV ads
- Optionally specify the minimum CPM you are willing to accept for each ad format
- Gain more granularity in your reports by breaking out performance by ad format

### Channels and Minimum CPM

In order to use AFV daisy chaining, you will need to pass in an additional channel ID indicating what type of ad you are requesting. This will allow you to break out the performance of video ads, graphical overlays, text overlays, and fullscreen endcaps in your reporting, and it will also allow you to optionally set a separate minimum CPM for each of those formats. Note: The minimum CPM does NOT apply to CPC text ads.

Please follow these steps:

- The default minimum CPMs set for each ad format are shown below. If you would like to modify any of the minimum CPMs, please notify your TAM, who will make the changes for you in the AFV back end system.
  - 1) Video ads: \$3.00
  - 2) Graphical overlays: \$3.00
  - 3) Text overlays: \$0.00
  - 4) Endcaps: \$0.00
- When you request video ads, pass in the channel ID "VideoAds".
- When you request graphical overlay ads, pass in the channel ID "GraphicalOverlayAds".
- When you request text overlay ads, pass in the channel ID "TextOverlayAds".
- When you request endcap ads, pass in the channel ID "EndcapAds".

*Example ad request parameters (parameters highlighted in yellow must be customized):*

```
// Video ads
request.adType = "video";
request.channels = ["VideoAds", "1234567890"];
```

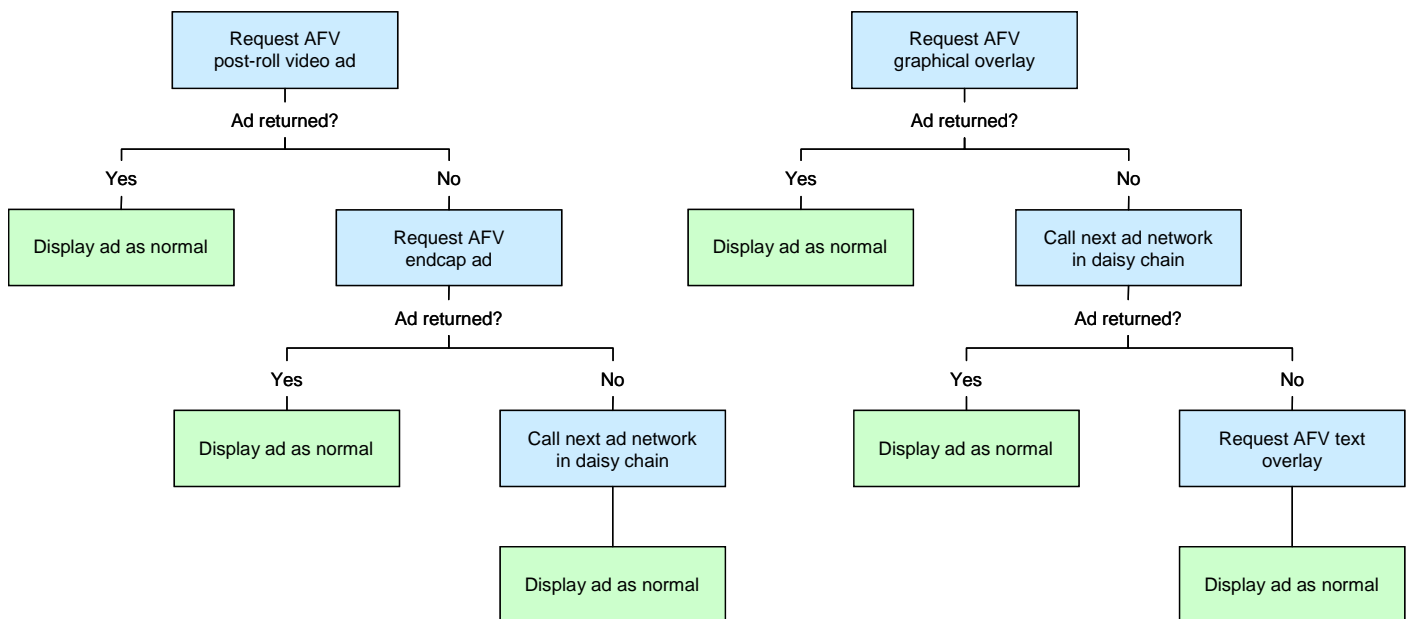
```
// Graphical overlay ads
request.adType = "graphical_overlay";
request.channels = ["GraphicalOverlayAds", "1234567890"];
```

```
// Text overlay ads
request.adType = "text_overlay";
request.channels = ["TextOverlayAds", "1234567890"];
```

```
// Fullscreen ads
request.adType = "fullscreen";
request.channels = ["FullscreenAds", "1234567890"];
```

## General Procedure

In order to add AFV to your ad network daisy chain, you will need to first request a particular ad format and determine if an ad was returned. To do this, your Actionscript code should check the `callbackObj.success` parameter of the ad request callback object. This is a Boolean value indicating whether ads were successfully returned. If the value returned is `true`, an AFV ad was returned and you should proceed with displaying the AFV ads as normal. If the value returned is `false`, an AFV ad was NOT returned and you should make a call to the next ad network in your daisy chain. See example code at bottom of page.



*Example AS2 or AS3 post-ad request callback method (parameters highlighted in yellow must be customized):*

```
private function onAdsRequestResult(callbackObj:Object):Void {
    trace("onAdsRequestResult: callbackObj.success = " + callbackObj.success);
    if (callbackObj.success) {
        var player:MovieClip = callbackObj.ads[0].getAdPlayerMovieClip();
        player.setSize(320, 240);
        player.setX(0);
        player.setY(0);
        player.load();
        player.playAds();
        player.pauseContentVideo = delegate(this, pauseStream);
        player.resumeContentVideo = delegate(this, resumeStream);
    }
    else {
        // Insert code here that calls the next ad network in your daisy chain
    }
}
```

## Reporting by Ad Format

One of the advantages that AFV daisy chaining provides is the ability to pull reports by ad format, using the daisy chaining channels. This will allow you to directly compare the performance of each ad type (video, graphical overlay, text overlay, fullscreen endcap).

In your AFV account, under the “Advanced Reports” section, the “Video eCPM” column for both aggregate and channel reports is calculated as follows:

**Existing calculation:** Video eCPM = (Revenue / Video Playbacks) \* 1000

The caveat is that when you select the aggregate report option, the “Video Playbacks” column reports streams. However, if you select the channel report option, the “video playbacks” column actually reports ad requests, regardless of whether we delivered an ad or not. This means that for ad formats where we do not have 100% coverage (such as graphical overlay ads or video ads), your “Video eCPM” stats are being diluted.

In order to more accurately calculate your “Video eCPM” for each ad format channel, we recommend that you manually perform the following calculation instead when using the daisy chaining channel reports:

**Revised calculation:** Video eCPM = (Revenue / Ad Impressions) \* 1000