



TECHNICAL PAPER

Optimizing video players for mobile-ready websites

Brady Kroupa
Lead Computer Scientist, Mobile
Adobe Systems, Inc.

May, 2010

© 2010 Adobe Systems Incorporated. All rights reserved.

If this technical paper is distributed with software that includes an end user agreement, this guide, as well as the software described in it, is furnished under license and may be used or copied only in accordance with the terms of such license. Except as permitted by any such license, no part of this guide may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, recording, or otherwise, without the prior written permission of Adobe Systems Incorporated. Please note that the content in this guide is protected under copyright law even if it is not distributed with software that includes an end user license agreement.

The content of this guide is furnished for informational use only, is subject to change without notice, and should not be construed as a commitment by Adobe Systems Incorporated. Adobe Systems Incorporated assumes no responsibility or liability for any errors or inaccuracies that may appear in the informational content contained in this guide.

Any references to company names in sample templates are for demonstration purposes only and are not intended to refer to any actual organization.

Adobe and the Adobe logo, ActionScript, Flash, Flash Builder, Flash Player, and Adobe Media Encoder are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA. Notice to U.S. Government End Users. The Software and Documentation are "Commercial Items," as that term is defined at 48 C.F.R. §2.101, consisting of "Commercial Computer Software" and "Commercial Computer Software Documentation," as such terms are used in 48 C.F.R. §12.212 or 48 C.F.R. §227.7202, as applicable. Consistent with 48 C.F.R. §12.212 or 48 C.F.R. §§227.7202-1 through 227.7202-4, as applicable, the Commercial Computer Software and Commercial Computer Software Documentation are being licensed to U.S. Government end users (a) only as Commercial Items and (b) with only those rights as are granted to all other end users pursuant to the terms and conditions herein. Unpublished-rights reserved under the copyright laws of the United States. Adobe Systems Incorporated, 345 Park Avenue, San Jose, CA 95110-2704, USA. For U.S. Government End Users, Adobe agrees to comply with all applicable equal opportunity laws including, if appropriate, the provisions of Executive Order 11246, as amended, Section 402 of the Vietnam Era Veterans Readjustment Assistance Act of 1974 (38 USC 4212), and Section 503 of the Rehabilitation Act of 1973, as amended, and the regulations at 41 CFR Parts 60-1 through 60-60, 60-250, and 60-741. The affirmative action clause and regulations contained in the preceding sentence shall be incorporated by reference.

SUMMARY

This document describes ways to optimize the user experience of your video content and playback on mobile devices. It discusses how to profile your video player performance, properly encode your video, and design your video player for mobile devices.

PREREQUISITES

In order to follow along, you'll need to do a few things first. Verify that you have the following items:

- A mobile device that supports [Adobe® Flash® Player 10.1](#).
- The ability to upload content to a publicly visible website.
- Optional: Development tools used to build your video player (e.g., Adobe Flash® Professional or Adobe Flash® Builder™).
- Optional: Tools used to encode your video (e.g., Adobe Media Encoder).

2.0 OPTIMIZING YOUR CURRENT VIDEO PLAYER AND CONTENT

In this section, we will walk through the steps to optimize your video player. First, we'll measure the current frame rate of your video player. Second, we will determine whether your video content needs to be re-encoded. Third, we will determine whether your video player needs to be optimized.

2.1 Profile your video playback performance

The first thing to do is install a version of Flash Player 10.1:

1. Download and install Flash Player on your device by navigating to <http://www.adobe.com/go/getflashplayer> on your device's web browser.
2. Test if Flash Player is installed properly on your device. Open the browser on your device and navigate to this URL: <http://bit.ly/wRILS>
If you see the version information, then you're all set; otherwise restart your device and retry Step 1.

Now that Flash Player 10.1 is successfully installed on your device, you can view your online SWF content on your device:

1. Browse to the URL of your website that has your mobile player on it. You may want to create a dedicated test HTML page that contains only your embedded player. The reason to do this is to avoid interference from JavaScript Ajax calls and other Flash Player (SWF) content, such as advertising, which could degrade performance of the video player. Another tip when testing on your device is to shorten the length of your test URL by using a service like <http://bit.ly>. This makes it easier to type the URL into the mobile browser.
2. Double-press the video player so it zooms in or takes up the entire screen. Try to zoom in enough so that no other Flash Player content (such as ads) are visible along the edge of the screen.
3. Hopefully you'll see your content in the browser. Begin the video playback and let it run for approximately one minute.

How did the playback of the video and audio look and sound? Was it smooth or choppy? Did it feel slow at the start and then speed up? Assessing the subjective experience will give you an initial impression of whether your content or player could benefit from optimization.

You can also configure your video player to measure the performance in a more quantifiable manner. The following document, with sample code, shows how to make simple changes to your video player to capture the video playback frame rate. To get the most benefit out of the following sections, please update your video player according to the steps described in [Profiling performance for Flash Player 10.1 in ActionScript](#).

2.2 Analyze the video encoding

Hopefully you were able to generate the average fps playback of your video. Was it greater or less than 15 fps? If it was less than 15 fps, the most common causes are video encoding not targeted for mobile, video players with overlapping content, background processing or timed events, or content in complex HTML pages with JavaScript or other SWF animations. In this section and the ones that follow, we'll try to diagnose the exact source of the problem.

Compare your content's video encoding settings against our best practices. According to [Delivering video for Flash Player 10.1 on mobile devices](#), you should encode your video using these settings:

Preferred format	Video: H.264, Constrained Baseline profile at 24 fps ¹ Audio: AAC-LC, 44.1 kHz, stereo			
Alternate format	Video: On2 VP6, Simple profile or Sorenson Spark at 24 fps ¹ Audio: MP3, 44.1 kHz, stereo			
	WiFi – H.264	WiFi – On2/Sorenson	3G – All	2.5G – All
Bit rate (combined)	500 kbps	350 kbps	350 kbps	100 kbps
Resolution	HVGA (480 x 320)	HVGA (480 x 320)	HVGA (480 x 320)	QVGA (320 x 240)
Audio bit rate	Up to 160 kbps	Up to 128 kbps	Up to 64 kbps	Up to 32 kbps

1. For true source frame rate greater than 24, downsample by an even factor such as 2, e.g. 30 to 15, to maintain optimal performance.

As shown in the table, the preferred format is H.264 Constrained Baseline. Flash Player 10.1 can utilize hardware acceleration with this format on most devices. Thinking about your target connection speed also helps when choosing the video's encoding. For wi-fi connections, videos should be encoded less than or equal to 500 kbps; for 3G connections, less than or equal to 350 kbps.

For comparison, let's play an optimized Adobe video in your video player. The reference video was encoded in H.264 at 500 kbps with a 480 x 320 resolution. Compare the playback frame rate of the Adobe video to the frame rate of your video to help determine if your video content could use further optimization:

1. Download our [reference video content](#).
2. Unzip the downloaded file. You'll see a `deploy` folder and `source` folder.
3. Upload the `deploy` folder to a publicly accessible location on your web server. This folder includes the sample video clip that you'll use for this test, and also a video player SWF that you'll use in the next section.
4. Change your video player to play the Adobe reference video that you just uploaded. For example, the source URL would look something like the following:
`http://www.yourwebsite.com/publicFolder/deploy/video/Adobe_Test_Video_H264_Baseline32_480x320_24fps_500k.f4v`
Be sure that your server does not re-encode the reference video stream; play it as-is.
5. If you have not already done so, modify your video player to measure the video frame rate (see the last paragraph in the previous section).
6. Upload your modified player to the website.
7. On your device, browse to your video player.
8. Double-click the video player so that it zooms in or takes up the entire screen.
9. Play the video until the average frames per second appears.

Did the average frame rate improve with the reference video? If it was higher than your video's frame rate, you'll want to try to improve how your video was encoded. For helpful steps on optimizing your video content, check out more [mobile video encoding guidelines](#).

2.3 Profile the video player

Now compare the performance of your video player with our reference video player. You'll continue to use the same reference video from section 2.2. You've already uploaded the reference video player to your website, so all you need to do is profile it:

1. To enable profiling on our reference player, in the `deploy` folder of the reference video player that you downloaded in section 2.2, open `referencePlayer.html` in a text editor.
2. Search for `flashVars`.
3. In that section add the following line of code:
`profilerEnabled: "true",`
on a new line above the following line of code:
`skinURL: "swf/adobeSkin.swf"`
4. Upload the modified `referencePlayer.html` to your website and replace the copy there. For example replace this remote file with your local copy:
`http://www.yourwebsite.com/publicFolder/deploy/referencePlayer.html`
5. Open the browser on your device and clear the its cache.
6. Navigate to `referencePlayer.html` from Step 4.
7. Press the play button. The video player should switch to full-screen mode. Allow the video to play for a minute. Touch the screen and press the pause button. The average frames per second will be displayed on-screen.

Was the average fps of your video player greater or equal to our reference player? If so, good job; your video player performs as well or better than ours! You should consider trying the steps above on multiple devices due to graphics acceleration and processor differences. For

example, video playback on the Google Nexus One might be acceptable but slower on the Motorola Droid. By sampling performance across devices you'll know if you need to continue optimizing your player or content to give the best experience to the most users.

If the average fps of your player is lower than Adobe's reference player, you might want to optimize your video player. For some helpful steps at optimizing your video player for mobile check out the following documents:

- [Video player optimizations for Flash Player 10.1](#)
- [Optimizing performance for the Flash Platform](#)
- [Flash Player 10.1 hardware acceleration for video and graphics](#)

Alternatively, you could replace your video player with the reference player from Adobe. You've already downloaded and used the player earlier in this section. Adobe's reference video player is free and very easy to configure and skin the user interface. For more information, read [Reference video player for mobile devices](#).

Otherwise, there are numerous third parties that offer services to host your content and provide their own optimized players. One such company is Brightcove. To learn more about their services, visit their page on the [Brightcove Online Video Platform](#).

3.0 DESIGNING FOR MOBILE DEVICES

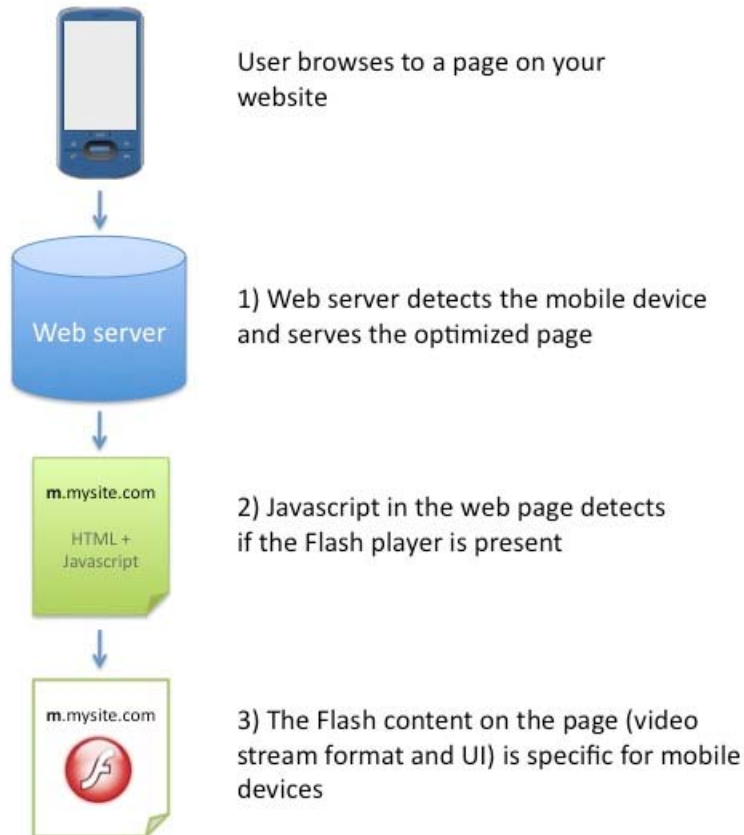
When a user visits your website, there are a number of opportunities to tailor the experience for browsing on a mobile device. By using a variety of techniques, you can improve users' experiences and intelligently deliver the right content for the right screen.

This section walks you through two strategies of configuring your website to detect what kind of device is browsing your website, check whether Flash Player is present, and determine which video stream is appropriate. We also tell you how to modify your video player so that its UI controls are appropriate for a device with a small screen.

3.1 Switch between mobile optimized and the full website

The first strategy we discuss is structuring your website to have a mobile-optimized and a full version of the site. The advantage for using this strategy is the content shown to users is specific to the device on which they're browsing. The obvious downside is a doubling of content. If this isn't an option for you, feel free to skip ahead to section 3.2 where the content itself adapts based on the device used for browsing.

The following diagram shows a high-level example of how your website and content could be configured to serve mobile and full versions of your website.



1) Web server detects the mobile device and serves the optimized page

When a web browser issues an HTTP request, it inserts some information in the header of the request. One of the inserted fields is called “user-agent.” The user-agent field describes the device on which the browser is running. For example, the Google Nexus One uses the following user-agent string:

```
Mozilla/5.0 (Linux; U; Android Froyo; en-us Build/MASTER)
AppleWebKit/530.17 (KHTML, like Gecko) Version/4.0 Mobile
Safari/530.17
```

By adding logic to your web server, you can serve mobile-specific pages when you detect a mobile browser’s user-agent string. A good way to handle redirecting a browser to a mobile-specific page on the server is by using an HTTP Redirect. Here is some more information on the [“Redirect” directive](#) in Apache’s manual and from redirection in [Microsoft IIS documentation](#). Also, there are some sample scripts that perform device detection at [John Boxall’s mobile device detection site](#).

It is important to note that the content of the user agent will vary from one device to the next, and may change if the software on a device is updated. For example, the device type is not listed in the user-agent string above. New devices and software updates appear very frequently, so you should design your server-side logic accordingly.

It is also a good idea to have an alternate way to access mobile content other than relying on the user-agent detection. For example, allowing users to directly enter a mobile URL, typically starting the URL with “m”, is helpful:
`http://m.yourwebsite.com`

2) JavaScript in the web page detects if Flash Player is present

Once your web page loads in your browser, you can easily detect in JavaScript whether or not Flash Player is installed. You can use this JavaScript class to redirect a browser to a new page or show alternate content in case Flash Player is not detected.

SWFObject is an open-source JavaScript class that implements this functionality for you. It also has a number of useful helper functions to embed and interact with SWF content.

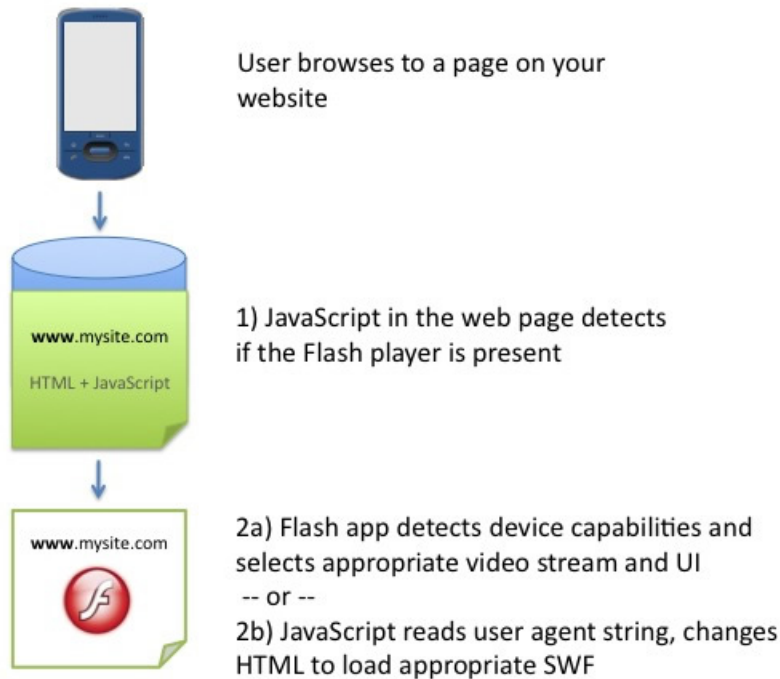
[Detecting Flash Player versions and embedding SWF files with SWFObject 2](#) explains the uses and benefits of SWFObject and how to download the source code. The Adobe reference video player uses SWFObject to detect the presence of Flash Player and show the appropriate download link if it’s missing.

3) The Flash content on the page (video stream format and UI) is specific for mobile devices

In Step (1) above, we determined whether the users’ browsing is using a desktop computer or a mobile device. If they were using a mobile device, they were redirected to a mobile-specific version of your site. The video player that is used on the mobile version of your site should be designed for a smaller display (see section 3.3, below). That player should request a mobile-optimized version of your video stream (see section 2.2, above). You may want to provide a way in the UI for users to view higher quality content if they think their devices can handle it.

3.2 Device-aware content

The second strategy we discuss is using a single version of your website for both desktop computers and mobile devices. In this strategy, you may want to adjust the content on your page depending on whether the browser is running on a desktop computer or a mobile device. For example, you may want to play a high-quality video stream on a computer and a lower-quality version on a smartphone.



1) JavaScript in the web page detects if Flash Player is present

This is the same as Step (2) in section 3.1.

2) Your Flash content or the HTML in a web page can change depending on what kind of device is browsing the page.

In this section, we'll present two approaches for detecting the type of device browsing your content. The first approach uses ActionScript®; the second approach uses JavaScript. After we describe the two approaches, we'll point out the strengths and weaknesses of each one:

a) Flash app detects device capabilities and selects appropriate video stream or UI

Flash Player 10.1 provides many useful calls in ActionScript to gather information on which platform the user is viewing your content. By combining knowledge of the device's screen dimensions, support for a mouseover, and others, you can infer whether the user is viewing your content on a mobile device. While you may not identify the specific device, you can use this information to choose the appropriate video stream according to the hardware or network capabilities. For example, if you've determined the user is viewing the content on a desktop machine, you can choose a higher quality video stream with higher bitrate and screen size, instead of one that would be targeted for mobile. You could also tailor the user interface and size of buttons and text on screen to make it easier to use on a touch-enabled device. [Delivering video for Flash Player 10.1 on mobile devices](#) describes how to introspect these capabilities and what assumptions you can make based on them.

While this strategy doesn't enable content to identify a particular device, such as a Google Nexus One versus a Motorola Droid, this strategy scales well and is robust and future-proofed.

b) JavaScript reads user-agent string, changes HTML to load appropriate SWF

Rather than having your SWF content figure out what device is browsing your content, you can embed JavaScript in the web page to do it. You can write a JavaScript function to parse the user-agent string when a web page is loaded. Depending on the user agent it detects, it can use the `document.write` method to embed a SWF tailored to the device. For example, if the user agent detected is an Android device, the embedded SWF can be designed to point to a mobile-optimized video stream.

As we mentioned above, relying on user-agent strings doesn't scale to include future agent strings and is slow because of all the string comparisons. The benefit to using this approach is the ability to detect specific devices and operating systems to help you take advantage of platform-specific capabilities. For more information about user-agent strings and how to use them, check out the information on [browser detection and cross-browser support](#) from Mozilla.org.

To get around the static nature of saving user agents in your JavaScript code, you can use WURFL—a service that maps user agents to device capabilities. This might be useful if you're trying to use JavaScript to detect specific attributes of the device that is browsing your website. For more information, please visit the [WURFL website](#).

To summarize: There are two approaches for enabling your website to work well for both desktop computers and mobile devices. The first is to use ActionScript to detect traits and device capabilities to tailor the quality of video streams and size of UI controls. While this strategy doesn't identify specific devices, it scales well and is robust and future-proofed. The other approach is to use JavaScript and parse user-agent strings to change the page's HTML, either to load a mobile-optimized SWF or not to show unnecessary content on the page. This is useful if you need to target specific devices and operating systems but is somewhat fragile and doesn't scale well as new devices and versions of operating systems are released.

3.3 Design for mobile

So far we've talked mainly about improving performance and how to introspect from the browser and ActionScript whatever content is optimized for mobile consumption. There's a broader issue of designing your content from the start, taking into account how users will interact with it on a device with a small screen and touch-interaction model. Leveraging the tools you learned from section 3.2, Step (2), you can make your content dynamically adjust your interface based on the platform it runs on. For example, you could make buttons and fonts larger, reduce reliance on `onRollOver` events, and handle orientation and full-screen changes smoothly.

The Adobe Design Experience team has put together a helpful document, [Design tips for creating mobile RIAs](#), describing some strategies and ideas to consider when designing your mobile content and video players.



Adobe Systems Incorporated
345 Park Avenue, San Jose, CA 95110-2704 USA
www.adobe.com

Adobe and the Adobe logo, ActionScript, Flash, Flash Builder, Flash Player, and Adobe Media Encoder are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States and/or other countries.

All other trademarks are the property of their respective owners.

© 2010 Adobe Systems Incorporated. All rights reserved. 04/10