



TECHNICAL PAPER

Full-screen orientation for mobile devices using **Adobe® Flash® Player 10.1**

Jani Leppanen, Rishit Shah, Brady Kroupa
Computer Scientists, Mobile
Adobe Systems, Inc.

June, 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, and Flash Player 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 Unpublished-rights reserved under the copyright laws of the United States.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

Adobe® Flash® Player 10.1 on mobile devices can display content in “full screen” mode, or it can display content that is embedded in an HTML page within the web browser. This document describes when and how Flash Player switches between these two modes on mobile devices, gives useful HTML tags and tips for optimally embedding your content for mobile devices, and discusses how to resize and adjust content presented in Flash Player when in full-screen mode.

Before reading this document, you should be familiar with [full-screen mode for earlier versions of Flash Player](#).

FULL-SCREEN IMPLEMENTATION FOR MOBILE DEVICES

When content presented in Flash Player 10.1 enters full-screen mode, it is rendered to the entire viewable region of the device’s screen. You no longer see any system toolbar, web browser address bar, or operating system chrome. Upon entering full-screen mode, Flash Player displays a temporary overlay message, instructing users how to exit full-screen mode. On Android devices, the message is “press back key to exit full-screen mode.”

When content is displayed in full-screen mode, it gets the input focus. All touch and drag events are handled by your ActionScript® code; they are not passed to the browser.

When content is embedded in a web page, on the other hand, input events are processed by the browser by default. No events are sent to Flash Player until the user sets focus to it by touching it. Once Flash Player has explicit focus, input events are sent to it before being processed by the browser.

For Flash Player 10.1 on smartphones, the HTML embed tag `allowFullScreen` no longer needs to be set to `true` for your content to be displayed in full-screen mode.

There are three ways to support the user launching full-screen mode. The preferred way is to add ActionScript code that explicitly handles user input events and switches to full-screen mode. The second way is that, when users long-press the content in Flash Player, an overlay message prompts them if they’d like to switch to full-screen mode. The third way, which makes it easy to retrofit existing content, is to use the new `fullScreenOnSelection` HTML tag.

Here’s an example in ActionScript where input events are used to switch to full-screen mode. The following code sample switches between normal mode and full-screen mode whenever the user taps the stage:

```
stage.addEventListener(MouseEvent.CLICK, toggleFullScreen);
```

```
function toggleFullScreen(event:MouseEvent):void
{
    switch(stage.displayState)
    {
        case "normal":
            stage.displayState = "fullScreen";
            break;
        case "fullScreen":
        default:
```

```
        stage.displayState = "normal";  
        break;  
    }  
}
```

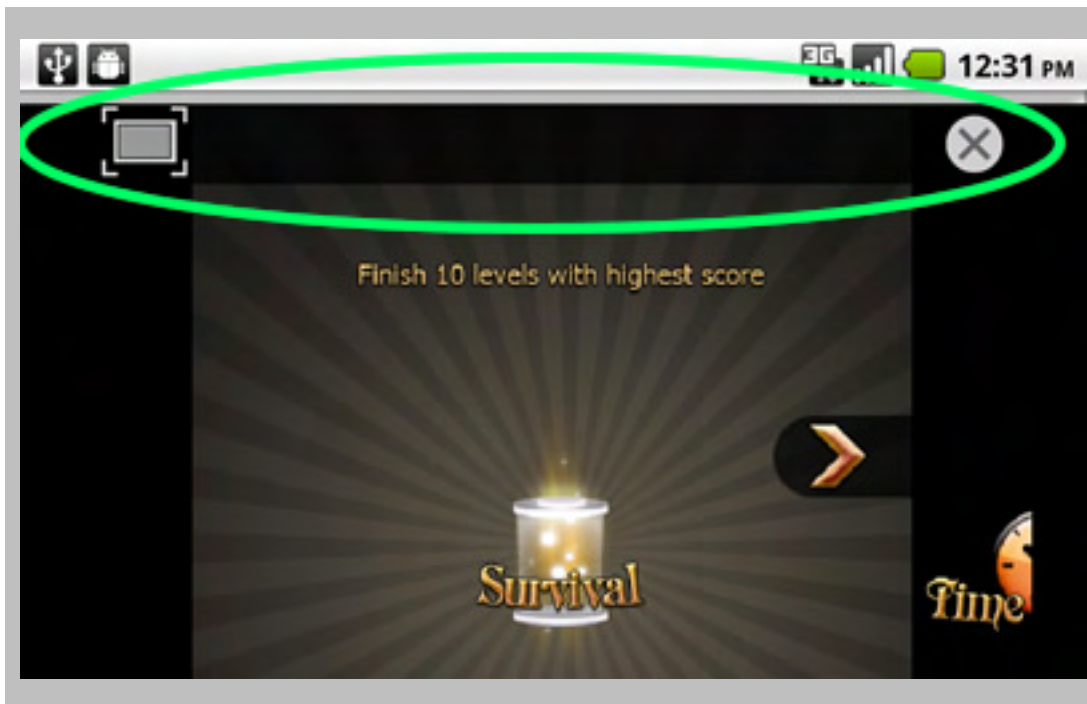
The `fullScreenOnSelection` HTML tag is convenient to use because it doesn't require any changes to your ActionScript code. If `fullScreenOnSelection` is set to `true`, the first time a user touches the content, it immediately switches to full-screen mode. For example:

```
<embed src="http://www.yoursite.com/example.swf"  
  width="480" height="380"  
  ...  
  allowFullScreen="true"  
  fullScreenOnSelection="true"  
  ...  
</embed>
```

The user may exit full-screen mode by pressing the "back" softkey or by having the content in Flash Player invoke a script to set the `stage.displayState` property to "normal." If your content requires textual input, users will need to exit full-screen mode first because text input is not allowed while in full-screen mode.

Once the user has exited full-screen mode, subsequent touch events do not cause the application to enter full-screen mode again. The touch-to-enter-full-screen behavior is restored if the page is reloaded in the browser.

If the user exits full-screen mode, where `fullScreenOnSelection` was set to `true` in HTML, an overlay appears, showing a rectangular "full-screen" button on the left and a circled "x" button on the right (see picture below). Tapping the full-screen button restores that mode, while tapping the "x" button hides the overlay.



Double-touch zooming versus full-screen mode

Entering full-screen mode is different from zooming in on content. The Android web browser supports a “double-touch” operation to fit as much content onto the screen as possible. The double-touch event does not switch to full-screen mode even if the HTML `fullScreenOnSelection` has been set. In addition, zooming in on your content does not affect the value of `stage.displayState`.

When you double-touch the content, you have not set focus to it. After you double-touch it, the content will not receive input events until it is touched once to gain focus.

If the content is loaded on a mobile-optimized web page and you don’t want the browser to zoom, you should consider using the new `viewport` HTML tag to prevent the browser from zooming in on your content:

```
<meta name="viewport" content="target-densitydpi=device-dpi,
width=device-width, user-scalable="no" />
```

Using this tag prevents any form of zooming in the browser, including the double-touch zoom.

Automatic change to landscape orientation for video in full-screen mode

Video playback generally looks better and uses more of the screen’s real estate in landscape orientation. When you switch to full-screen mode, Flash Player 10.1 now automatically detects if there’s a `Video` object contained in your content. If a video is detected, Flash Player sets and locks the orientation to landscape. This happens whether the `Video` object is on the stage or not, and happens regardless of the `netStream` object having opened a connection to the source. If you dynamically add a `Video` object at runtime while in full-screen mode, the orientation will switch to landscape.

The only exception to this rule is if `stage.fullScreenSourceRect` or the published stage size (in any scale mode other than `noScale`) has portrait dimensions.

HANDLING ORIENTATION CHANGES AND RESIZING CONTENT IN FULL-SCREEN MODE

On mobile devices, the screen orientation (and therefore the screen aspect ratio) may change while in full-screen mode. On desktop computers, the screen’s dimensions never change. Therefore, existing content may need to be updated to handle this case.

This section shows you how to define the behavior of Flash Player 10.1 when the display orientation changes while in full-screen mode. Often, the size and position of UI controls must be adjusted when the orientation change occurs.

Resizing content in full-screen mode upon orientation change

It's important for content in full-screen mode to resize and position itself properly after an orientation change. This section explains how to adapt to the new aspect ratio, either by changing the full-screen source rectangle or by laying out its content again.

First, consider a case where your content is embedded in HTML with the scale mode set to `noScale`. The content contains a square-shaped movie clip placed at the bottom right corner of the stage. When the user taps the content, the code handles that event and switches to full-screen mode. It also handles the resize event and tests the `stage.displayState` to update the square movie clip's position. By setting the scale mode to `noScale` and dynamically positioning the elements on the stage, you can maintain the content's fidelity:

```
stage.addEventListener(flash.events.Event.RESIZE, resizeListener);
stage.addEventListener(MouseEvent.CLICK, toggleFullScreen);

function toggleFullScreen(event:MouseEvent):void
{
    // same as previous toggleFullScreen example
    ...
}

// Since this SWF is published as noScale and there
// is no full screen source rectangle, when in full screen,
// stage.stageWidth and stage.stageHeight are the
// same as stage.fullScreenWidth and stage.fullScreenHeight.
//
// The following code handles both orientation changes and
// resizing to fullscreen when no scaling occurs
function resizeListener(event:Event):void
{
    // square is a movie clip that we are placing in
    // the bottom right corner of the screen.
    square.x = stage.fullScreenWidth - square.width;
    square.y = stage.fullScreenHeight - square.height;

    // note: if this code was only specific to moving
    // elements in fullscreen you could add the following test
    // if (stage.displayState == "fullscreen") {...}
}
```

If your content **does not** specify a full-screen source rectangle and uses a scale mode **other than** `noScale`, you don't need to react to the resize events. The entire stage will be visible in both orientations. Depending on the shape of your stage, you may see letterbox-like empty bars at the top and bottom or left and right sides of the screen.

If your content **does** specify a full-screen source rectangle for **all** scale modes, listen for resize events and change your `fullScreenSourceRect` accordingly. You may want to change its shape to correspond to the new shape of the display. For example, change `fullScreenSourceRect` from `300 × 400` to `400 × 300` depending on the orientation. Otherwise, if your `fullScreenSourceRect` is a tall shape and the display is rotated to landscape orientation, Flash Player shrinks the content to a very small zoom level and your screen's real estate is wasted on the left and right sides. There's also potential for "outside the stage" sprites that the developer has put just offstage to appear when this resizing happens if the stage is not updated or redrawn. You can find out the shape of the display by looking at

the values of `stage.fullScreenWidth` and `stage.fullScreenHeight` to determine the aspect ratio, and then resize the `fullScreenSourceRect` accordingly.

The advice in this section can be summarized as follows:

fullScreenSourceRect is defined	scaleMode	Tip
no	noScale	Your application should listen for the resize event. Upon orientation change, the resize handler should adjust the positions and sizes of user interface controls on the screen. See sample code above.
no	all modes except noScale	No need to handle resize events. Depending on the shape of your stage, you may see letterboxing.
yes	all modes	Your application may optionally listen for the resize event. Upon orientation change, your resize handler will typically change the width and height of the <code>fullScreenSourceRect</code> to match the aspect ratio of the screen, and it may also adjust the positions and sizes of UI controls so that they are not offscreen.

Tips concerning `fullScreenSourceRect`, scale mode, and dimensions upon orientation change

Because the orientation can change on smartphones, there are some subtle differences in how the dimensions of the content change depending on whether `fullScreenSourceRect` is set and what the scale mode is.

The following table gives some tips to help you understand how the dimensions may be updated after the orientation changes:

fullScreenSourceRect is defined	scaleMode	Tip
yes	all modes	<p>Upon orientation change, the aspect ratio <code>fullScreenSourceRect</code> stays the same, even though the display size changes. To inspect the new dimensions of the display, be sure to use <code>stage.fullScreenWidth</code> and <code>stage.fullScreenHeight</code>, <i>not</i> <code>stage.stageWidth</code> and <code>stage.stageHeight</code>.</p> <p>It is no longer true (as cited in the ActionScript 3 Language Reference) that <code>stage.fullScreenSourceRect</code> cannot be changed while in full-screen mode. For <code>noScale</code> content to redo its layout and take advantage of the entire screen when orientation changes, the content must be able to change the <code>fullScreenSourceRect</code>.</p>
yes	all modes	<p>The minimum width and height of <code>fullScreenSourceRect</code> for Flash Player on mobile is 302 and 74 pixels, respectively. If the content sets either dimension to something smaller, the dimension will be forced to the minimum value.</p>
no	<code>noScale</code>	<p>Both <code>stage.stageWidth</code> and <code>stage.stageHeight</code> match the display size and update upon orientation change.</p>
no	all modes except <code>noScale</code>	<p>Both <code>stage.stageWidth</code> and <code>stage.stageHeight</code> are equal to the published stage size. The aspect ratio of the stage dimensions remains the same upon orientation change.</p> <p>Note: The size of the SWF object in the HTML document may or may not match the stage size. You need to look inside the SWF or FLA to determine the stage size.</p>
yes or no	<code>noScale</code>	<p>While in normal mode, only <code>noScale</code> content receives resize events. Resize events are triggered when the HTML element's size changes. The HTML element's size may or may not update when orientation changes.</p>
yes or no	all modes	<p>Resize events occur for all scale modes when screen orientation changes while in full-screen mode.</p>

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

Adobe and the Adobe logo, Flash and Flash Player 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. 05/10