



# **Flash Sizing Zen:** Android Devices

Getting Android to Respect Your Intent

[v6] updated on June 7, 2010

by Allen Ellison (aellison@adobe.com)

Achieving Zen .....	1
<i>Introduction</i> .....	1
<i>Full-Screen Mode</i> .....	1
<i>Embedded Mode</i> .....	3
HTML Guidelines for Mobile-Optimized Flash Content.....	4
<i>Tell browser your content is mobile-tailored</i> .....	4
<i>Lock the Width on Android</i> .....	5
<i>Don't cache</i> .....	5
<i>Use SWFObject 2</i> .....	5
<i>Set the Width and Height</i> .....	6
<i>Get Rid of the Address Bar</i> .....	6
<i>Go Full Screen on Selection</i> .....	6
<i>Resizing Example</i> .....	8
Scaling and Aligning Content .....	9
<i>Choose a Scaling Mode Appropriate for Your Content</i> .....	9
<i>Align Your Content Relative to the Browser/Screen</i> .....	10
Device Orientation .....	11
<i>Locking Your Content into Landscape Mode</i> .....	11
<i>Locking Your Content into Portrait Mode</i> .....	12
<i>Detecting a change in orientation or full screen mode</i> .....	13
Sizing Your Content Appropriately .....	14
Additional Notes .....	15
Appendix: Avoid Transcoding.....	16

## ACHIEVING ZEN

### INTRODUCTION

When you bring Flash games and content to the multi-screen web using Flash Player 10.1, there are a wide variety of requirements and challenges that you have to meet. Not the least of which is ensuring that your content is sized correctly.

There are two primary display modes that Flash Player 10.1 supports on mobile devices: embedded and full-screen. Adobe AIR for mobile, which lives outside the browser, has greater display flexibility but is outside the scope of this document.

### FULL-SCREEN MODE

When navigating to a new URL that contains Flash content, mobile devices will never automatically enter Full Screen mode. Any of the following user interactions will invoke full-screen mode:

- If the content would ordinarily invoke full-screen mode on the desktop (e.g. HTML object/embed tag contains the parameter `allowFullScreen="true"` and a button or key-initiated event set the `displayMode` to true), then full-screen mode will also be initiated on mobile. All of the full screen security restrictions/features apply.
- If the object tag also contains the parameter `'fullScreenOnSelection'`, then the content will go full-screen when the user selects the content (and they do this by tapping on the content area). The user will receive a hint that tapping the content will make it go full-screen and once selected, all of the standard full screen security features apply.
- If the user does a long-tap over the content area (when not in full screen mode already), they will be presented with an option to make the content go full-screen, regardless of whether or not the `allowFullScreen` option is set to true or not. Long-tap in full screen mode does nothing special. This is no way to disable this functionality, no way to opt-out nor is there a way to opt-in. Once the user selects the full-screen option, the content goes full screen and all of the standard full screen security features apply.

The browser will leave full-screen mode when any of the following happen:

- The user hits the back button (on Android devices) or initiates an equivalent gesture on non-Android devices or devices without a back button.
- The content that is currently full screen sets the display mode to normal, regardless of whether or not this was user-initiated.
- The content's ActionScript (and presumably the wrapper's JavaScript) attempts to navigate to a new URL – incidentally I recommend against attempting to launch the URL in a new window – this can result in some undesirable behavior (*content appears to remain in full screen mode but no longer responds to interaction*).

Once the user has left full-screen mode (thereby returning to embedded mode), re-selecting the content (even if `fullScreenOnSelection="true"`) will not have any effect (unless you refresh the page, or navigate away and back). Because of this, I recommend that gaming content pause the game when it detects exiting full screen mode.

Full Screen mode offers the following benefits:

- Maximum utilization of device's screen real estate,
- Since user interaction is required to invoke full-screen mode, you can remain assured that the content is always selected,
- (which means that you can rest assured that your content is receiving DPAD and trackball events)
- You can lock the screen into landscape mode (although not in portrait),
- You can ensure that your content is sized appropriately, regardless of orientation
- You don't have to be concerned with gesture ambiguity – which can result in your content missing some touch-related events and in many touch-related events being slow to respond.

Full Screen mode has the following drawbacks:

- You cannot access the device's physical or virtual keyboard while in full screen mode.
- HTML content will not display in full-screen mode (although JavaScript will continue to work) – this may affect some ad management frameworks.

## **EMBEDDED MODE**

By default, when the user navigates to a new URL that contains one more pieces of Flash content, the Flash content is, by default, in embedded mode. There is a variation on embedded mode that will sometimes be referred to as 'selected' or 'focused' mode. All that this means is that the user has selected the content by tapping on its content area. When this happens, the content area will briefly flash to indicate that it has received focus.

Selected/focused content has the following advantages over regular embedded content:

- Higher precedence than other Flash content – if the device runs short on memory for example, other Flash content will be jettisoned first
- Will receive keyboard and DPAD/trackball events
- Can receive a greater variety of gesture interactions

Although it appears to not work today, you may eventually be able automate the selection of content, although this will not automatically result in the content going full screen, even if `fullScreenOnSelection="true"`.

When supported, it should look equivalent to the following:

```
document.getElementById("YourFlashMovie").focus();
```

## HTML GUIDELINES FOR MOBILE-OPTIMIZED FLASH CONTENT

### TELL BROWSER YOUR CONTENT IS MOBILE-TAILORED

Once you go into full screen mode, getting your content the right size is a lot easier. But until your content is in full screen mode, getting it sized correctly is a bit tricky. The Flash Player team recommends:

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic
1.0//EN" "http://www.w3.org/TR/xhtml-basic/xhtml-
basic10.dtd">
<html xmlns="http://www.w3.org/1999/xhtml"
lang="en" xml:lang="en">

<head>
    <meta http-equiv="Content-Type"
    content="text/html; charset=utf-8" />
    ...
</head>
</html>
```

for the following reasons:

- it avoids Verizon Transcoding (see Appendix for more details)
- It avoids mislabeling the content as WAP
- uses one of the W3C standard XHTML subsets
- note that this basic sub-type **intentionally** excludes support for frames, but this is probably a non-issue for gaming content.

## LOCK THE WIDTH ON ANDROID

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

You can also specify the width in pixels, such as:

```
<meta name="viewport" content="target-  
densitydpi=device-dpi, width=480, user-scalable=no"/>
```

It's worth noting that there are other viewport attributes which Android appears to ignore (or that there are special requirements like they all must be present): **initial-scale**, **minimum-scale** and **maximum-scale**.

## DON'T CACHE

Don't cache (while in development) - unless you like manually clearing your cache every time you look at your content, add the following meta tag to your head section:

```
<meta http-equiv="CACHE-CONTROL" content="NO-CACHE">
```

I'd recommend removing that tag once you're in production.

## USE SWFOBJECT 2

It can simplify the process of embedding Flash content, allow you to reach more customers gracefully and easily detect whether or not the user has the right Flash Player version installed. You can download the source and documentation from the official swfobject page:

<http://code.google.com/p/swfobject/>

```
swfobject.embedSWF("/swf-path/yourContent.swf", "theGame",  
"480", "678", "10.1.61", "/swf/expressInstall.swf",  
flashvars, params, attributes);
```

## SET THE WIDTH AND HEIGHT

You can use 480 x 678. On the Nexus One, this is the size of the screen after subtracting the browser chrome. If you're using the resizing code, this isn't critical, because it'll get reset soon enough by the `resize` Javascript method.

Alternatively, you can set the width and height both to 100% - this works fine for certain types of content – in that case, the `resize` method may prove to be un-necessary. You can also set to `window.innerWidth` & `window.innerHeight`.

## GET RID OF THE ADDRESS BAR

```
window.scrollTo(0,1);
```

I don't recommend doing this if your Flash content occupies the entire window and listens for Mouse move events. This might result in the user being unable to scroll the address bar back into view, which would be frustrating and while the user can hit the back button to get to the previous page which did not block using the address bar, it can result in a bad impression.

## GO FULL SCREEN ON SELECTION

Allow your content to go full screen the first time the user taps on it by setting the follow parameters when embedding the SWF:

```
fullScreenOnSelection="true"
```

Note that while `allowFullScreen` works on desktop and mobile, `fullScreenOnSelection` is specific to mobile instances of the Flash Player. Also, it will work regardless of whether or not `allowFullScreen` is true.

*Note: full screen mode is subject to the same security restrictions as on the desktop, namely you don't have access to the keyboard (but can still detect DPAD/trackball events which act like up, down, right, left arrow keys).*

- The `fullScreenOnSelection` parameter only applies the first time the user selects your content while in embedded mode. If for some reason, the user backs out of full screen mode and then selects your content, the content will not go full screen.

- Therefore, I recommend that you detect the case where the user backs out of full screen mode (by watching the Stage resize events) – pausing the game if they exit full screen, and then when the user hits the resume button, you just need to re-invoke the full screen mode again explicitly using ActionScript (associated with the click event).
- Ensure that the user has the option to install the Flash Player.

If you forego SWFObject, be sure that the user has a way of being re-directed to Adobe.com to get the Flash Player:

```
<noscript>
<object classid="clsid:D27CDB6E-AE6D-11cf-96B8-444553540000"
width="480" height="678" id="testStage">
<param name="movie" value="testStage.swf" />
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<param name="allowScriptAccess" value="sameDomain" />
<param name="allowFullScreen" value="true" />
<!--[if !IE]>-->
<object type="application/x-shockwave-flash" data="testStage.swf"
width="480" height="678">
<param name="quality" value="high" />
<param name="bgcolor" value="#ffffff" />
<param name="allowScriptAccess" value="sameDomain" />
<param name="allowFullScreen" value="true" />
<p> Either scripts and active content are not permitted to run or
Adobe Flash Player version 10.1.0 or greater is not
installed.</p>
<a href="http://www.adobe.com/go/getflashplayer">

</a>
</object>
</object>
</noscript>
```

## RESIZING EXAMPLE

This example handles sizing issues in both portrait and landscape (thanks to Dan Skilken from SonicSwap):

```
<script type="text/javascript">

    var iw = window.innerWidth;
    var ih = window.innerHeight;
    function resizeHandler()
    {   var scrolling = false;
        iw = window.innerWidth;
        if(iw > 480) {

            window.scrollTo(0, 1);
            setTimeout("resizeSWF();", 800);

        } else {resizeSWF();}
    }
    function resizeSWF()
    {
        document.getElementById("theGame").width =
window.innerWidth;

        document.getElementById("theGame").height =
window.innerHeight;
    }
    window.onresize = resizeHandler;
    window.onload = resizeHandler;

</script>
```

This code will not scroll out the address bar in the portrait mode, but will in the landscape mode. It can certainly be modified to remove the scroll bar in both directions, but right now, there are problems in FroYo with removing the scroll bar in the portrait mode (although this may have been recently fixed).

## SCALING AND ALIGNING CONTENT

### CHOOSE A SCALING MODE APPROPRIATE FOR YOUR CONTENT

In the object/embed tag, you can set the **scale** parameter to one of the following:

- **showall** – default mode – ensures that all content that fits the stage is visible
- **exactfit** – almost never desirable – stretches/distorts content
- **noborder** – always fills the available area – but frequently you lose some of your content around the border
- **noscale** – although it is not trivial to implement, this is the best possible recommendation for multi-screen content – it does require that some type of liquid layout solution is implemented, so that you can keep text readable, icons recognizable and make the best use of the available screen real estate.

Alternatively, in ActionScript, you can accomplish the same thing dynamically by setting the **stage.scaleMode** property to **StageScaleMode.SHOW\_ALL**, **StageScaleMode.EXACT\_FIT**, **StageScaleMode.NO\_BORDER** or **StageScaleMode.NO\_SCALE** respectively.

And of course if you're using SWFObject, you can pass the scale in with the other parameters.

Example:

```
var params = {      id: "flashcontent", name: "flashcontent",
                   menu: "false", allowFullScreen: "true",
                   fullScreenOnSelection: "true", scale:"showall",
                   valign:"middle",
                   valign:"middle"};
```

## ALIGN YOUR CONTENT RELATIVE TO THE BROWSER/SCREEN

There are two object/embed parameters that apply here: **align** (specifies where the area specified by the width and height is aligned in relationship to the browser boundaries) which can take the values 'l','r' and 't' or 'middle' which specify that the Flash content is aligned against the left right or top borders, respectively (or default, is centered). If the width and height are both 100%, then the align value should have no effect. In addition, it should have no effect when in full screen mode.

As well, there is the scale alignment (**salign**) parameter (specifies where the stage content is relative to the width, height which was specified in the object/embed tag) which can take the values 'l','t','r','tl' and 'tr' as well as 'b', 'bl', 'br'; (the default value is not yet determined). This parameter still applies in full screen mode or if either/both the width & height are 100%. This parameter does not apply however if the scale mode is set to 'exactfit'.

## DEVICE ORIENTATION

### LOCKING YOUR CONTENT INTO LANDSCAPE MODE

Some gaming content is orientation neutral (the UI is roughly square), while a majority of gaming content seems to be desktop-biased (landscape) although there are certainly a number of games that play better in portrait mode.

The orientation API is not available now but is on the roadmap for a future release. In the interim, there is a compromise solution available that can solve the need for gaming content that benefits from being locked into the landscape orientation:

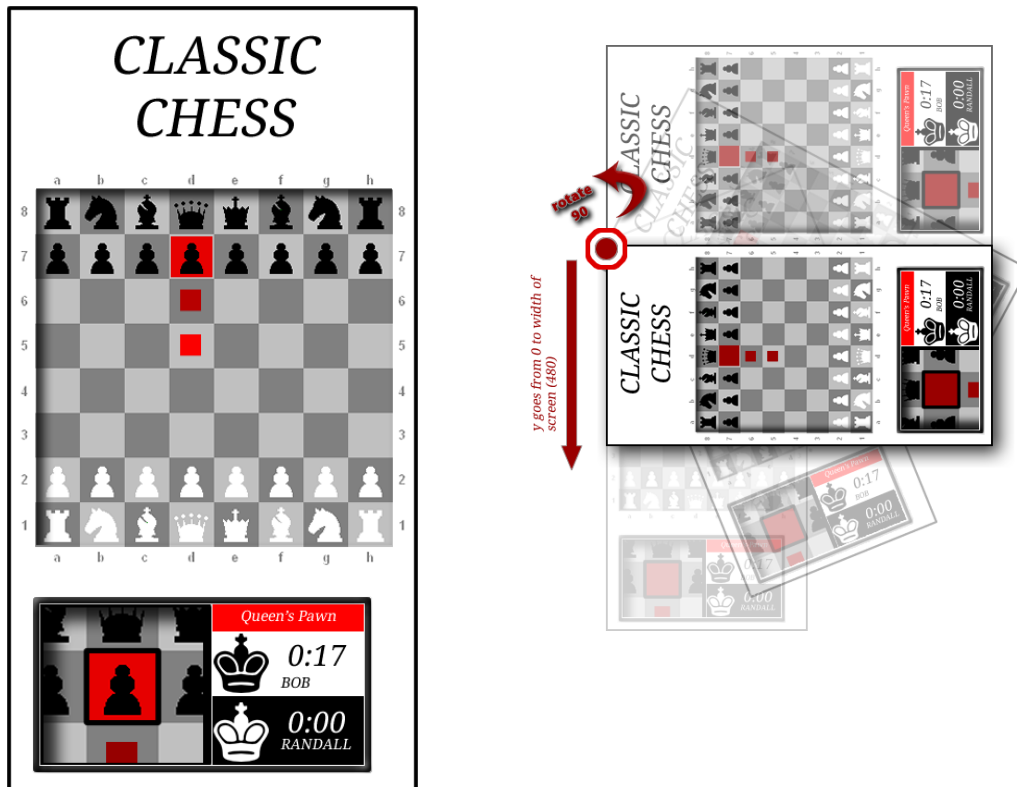
To force landscape mode, you simply need to include a video object if you are using Flash CS4 or Flash CS5. Or, using ActionScript, instantiate a video object and attach it to the stage. You don't need to do anything further with the video object. When (and only when) your content goes full-screen it will automatically lock into the landscape orientation, as a general rule. However, if you specify using `FullScreenSourceRect` dimensions that are indicative of portrait mode (`height > width`), then the content orientation will not be locked in.

```
video:Video = new Video();

private function MyClassicChessGame() {
    super();
    addChild(video);
}
```

## LOCKING YOUR CONTENT INTO PORTRAIT MODE

There is no similar feature for portrait mode. However, you can achieve a similar result by locking the stage into landscape orientation, and then re-positioning and rotating your content to re-orient it.



Assuming that all of your visible game content is in a Sprite called game,

```
game.rotation=-90;
game.y=stage.fullScreenHeight;
```

Remember that the DPAD and trackball are going to report their data assuming the landscape orientation, so you will need to explicitly translate the events (up arrow becomes a right-arrow event, etc.)

## DETECTING A CHANGE IN ORIENTATION OR FULL SCREEN MODE

To detect when your content enters or leaves full screen mode, you can listen for a `FullScreen` event.

```
import flash.events.FullScreenEvent;

function fullScreenChange(event:FullScreenEvent):void {
    if(event.fullscreen) {
        // entered full screen
    } else {
        // leaving full screen
    }
}

function init():void {
    stage.addEventListener(FullScreenEvent.FULL_SCREEN, fullScreenChange);
    ...
}
```

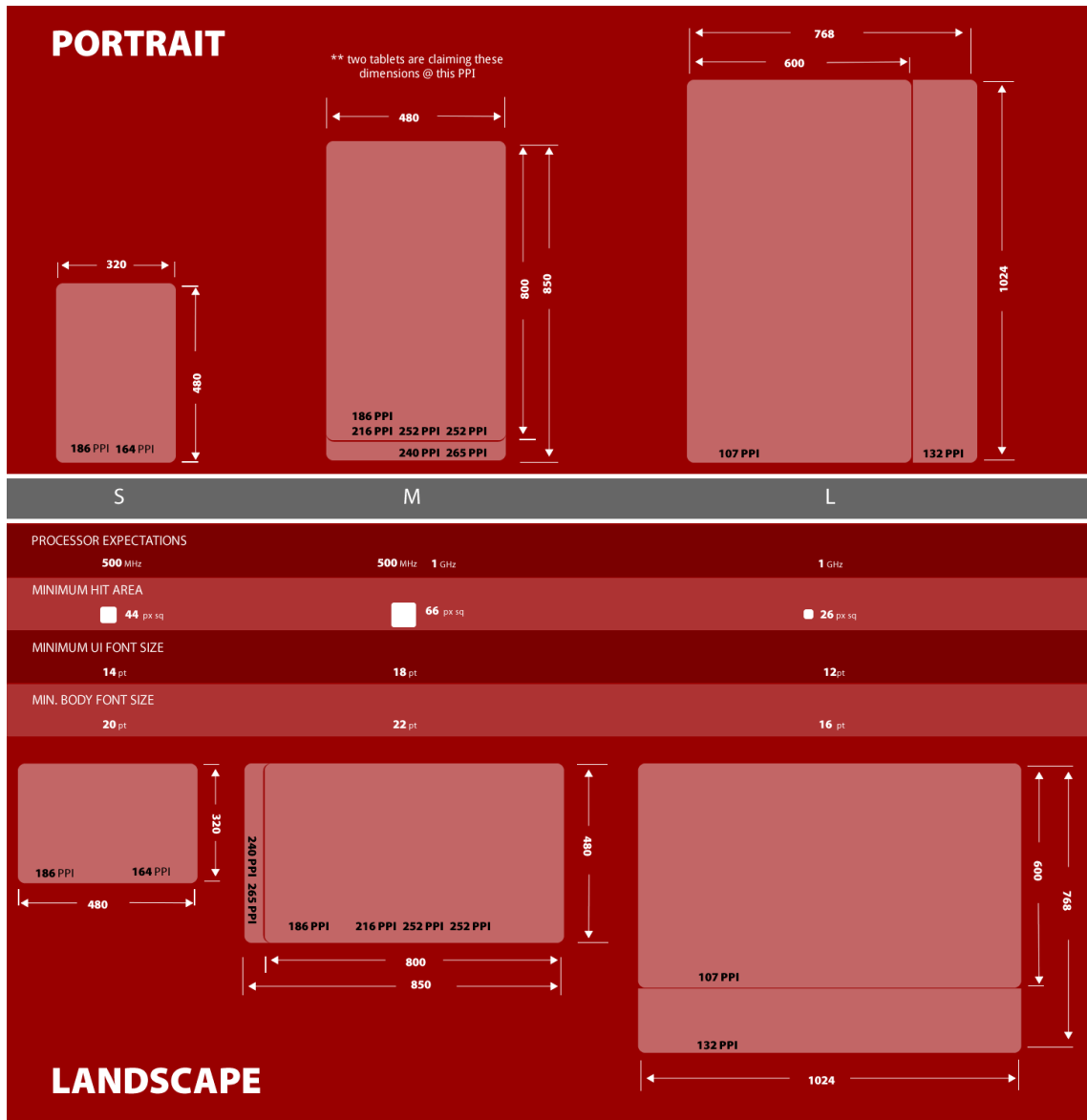
[http://help.adobe.com/en\\_US/ActionScript/3.0\\_ProgrammingAS3/WS2E9C7F3B-6A7C-4c5d-8ADD-5B23446FBEEB.html](http://help.adobe.com/en_US/ActionScript/3.0_ProgrammingAS3/WS2E9C7F3B-6A7C-4c5d-8ADD-5B23446FBEEB.html)

You can also detect a full screen change by listening for the stage's `RESIZE` event, although that logic is a little more complex. The resize event though will tell you if the orientation changed by comparing `stage.stageWidth` to `stage.stageHeight`.

I recommend that games, upon detecting that the user has left full-screen mode, pause the game, and that the resume button then re-invokes full-screen – **particularly** if you are attempting to lock in the orientation as the orientation can't be locked when displayed as embedded content.

## SIZING YOUR CONTENT APPROPRIATELY

The diagram below represents some common mobile screen specifications that you should expect. This should in general, give you an idea of what to aim for in terms of hit areas and font sizes to ensure that your content is readable.



On a Nexus One, the available height for browser content after subtracting the notification bar and browser chrome is 678 pixels (out of 800). If you don't want to change the aspect ratio of your content once it goes full-screen on a Nexus One, you would create your content targeting 480 x 480, then you would choose 406 x 678 for the object/embed area (assuming you don't go with 100%/100%) and then specify 'showall' for your scale mode. It means that you will lose some quality in text and images when in the browser, but in full-screen mode, the experience will be pixel-perfect. To be 100% certain, when you detect that the user has gone full-screen you can force the content to be non-scaled and aligned at the top-left corner.

The advantage of being pixel-perfect is that not only will there not be odd graphical behaviors (a line seeming to disappear because of being on an odd boundary, etc.) but your performance will improve as the Flash Player doesn't have to do the additional work of scaling your content.

## ADDITIONAL NOTES

- **wmode**

The **wmode** parameter of the OBJECT/EMBED tag for **Flash Player 10.1 on mobile** supports:

- **window** (default which means that the movie is stuck exactly where it's specified by the OBJECT/EMBED tag),
- **opaque** (meaning it can be moved or resized by JavaScript but no transparency is supported and it is not recommended that you try to overlay any content over the Flash content),
- **transparent** (content can be moved & resized — and allows HTML content to show through Flash content where the alpha channel allows) (although it's **not recommended** and only supports the Flash over HTML use case, not HTML over Flash, or Flash over Flash).

However, **Flash Player 10.1 on mobile** does not support the **direct** or **gpu** wmodes. If specified, its behavior will be identical to that of **window** mode (as hardware-based acceleration is not meaningfully supported on mobile for anything other than video). Also, Pixel Bender requires hardware-based acceleration features and is not available on platforms where hardware acceleration is not supported.

## APPENDIX: AVOID TRANSCODING

The largest US mobile carrier, Verizon Wireless, has started funneling traffic between Verizon feature phones and the web through a transcoder from Novarra. Verizon calls the service "*Optimized View*" and has added [promotional information](#) and a [FAQ](#) to their customer website. There is also [a page on the carrier's developer site](#) which has links to a opt-out form and to a PDF document detailing the rules that the transcoder uses to determine which sites to transcode.

Verizon/Novarra say that they won't change the User Agent header or transcode sites that have:

- submitted an opt-out request or have a URL corresponding to one of these patterns: \*.mobi, m.\*, mobile.\*, avantango.\*, wap.\*, iphone.\*, <domain>/m/\*, <domain>/mobile/\*, pda.\*, wireless.\*, wml.\*, xhtml.\*, <domain>/m/, <domain>/gmm/, <domain>/portable

or transcode documents that use one of the following document types:

- **XHTML Mobile**

```
<!DOCTYPE html PUBLIC "-//OMA//DTD XHTML Mobile 1.2//EN"  
"http://www.openmobilealliance.org/tech/DTD/xhtml-mobile12.dtd">
```

```
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.1//EN"  
"http://www.wapforum.org/DTD/xhtml-mobile11.dtd">
```

```
<!DOCTYPE html PUBLIC "-//WAPFORUM//DTD XHTML Mobile 1.0//EN"  
"http://www.wapforum.org/DTD/xhtml-mobile10.dtd">
```

- **XHTML Basic**

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.1//EN"
```

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML Basic 1.0//EN"  
"http://www.w3.org/TR/xhtml-basic/xhtml-basic10.dtd">
```

or transcode documents that use one the following MIME types:

- Open Mobile Alliance Recommendation (largely ignored)  
`application/vnd.wap.xhtml+xml`
- `text/vnd.wap.wml`

or transcode documents which contain a self referencing link rel tag:

- `<link rel="alternate" media = "handheld" href=mymobilesite.com/>`

or transcode documents which send a "Cache-Control: no-transform" header or corresponding metatag in the document `<head>`.

Transcoding Resources:

- <http://wapreview.com/blog/?tag=verizon-mobile-content-transformation>
- <http://www.w3.org/TR/xhtml-media-types/>