HTML Content Creation Guidelines for Cisco Vision Dynamic Signage Director

First Published: 2017-11-21

Revised Date: 2018-02-08

This module provides some general guidelines for creating HTML5 content for playback on the media players in Cisco Vision Dynamic Signage Director.

Note: These guidelines should generally work for the media players in Cisco Vision Dynamic Signage Director. However, there might be cases where certain elements do not render as expected. Be sure to test all of your HTML content before putting it into production to confirm its display meets your expectations. For more information, see Verification of HTML Content in Cisco Vision Dynamic Signage Director, page 56.

Best Practices for HTML Features with Hardware-Acceleration

Note: This section describes some of the hardware-accelerated features and guidelines that generally should work for the media players. However, there might be caveats and limitations that are not documented here.

The following features and effects are hardware-accelerated and recommended:

- SVG graphics combined with CSS transforms
- Effects that use `-webkit-transform` (preferred), or `transform`
- Animations that use `-webkit-animation-*` and `-webkit-keyframes`
- Any CSS transitions (for example, `-webkit-transition`)

Note: The use of the “`-webkit-`” prefix is not required, but it is recommended as a best practice.

The following features are not hardware-accelerated and should be avoided:

- `-ms-transform`, `-moz-transform`, or `-o-transform`

Note: This restriction includes avoiding other unsupported Chromium features that begin with -ms- and -moz- and -o-.

- Javascript-based animations and effects
- Video only when tagged with the `hwz` attribute

  Note: The video element, by default, is not hardware-accelerated and is not recommended or supported—unless you tag the video with the `hwz` attribute. Only 2 video regions can be on the screen at the same time. This includes other live and local video regions from Cisco Vision Dynamic Signage Director.

Content Restrictions for HTML5

The following content restrictions are associated with HTML5 pages:

- HTML5 content should not be used as a general purpose web browser. The media players are an HTML5 player with interactive capabilities, rather than a web-surfing tool.
- The media players do not support Flash content. Any HTML5 pages that have embedded Flash content will not display correctly. Most Flash authoring applications, including the Adobe Creative Suite, have tools that allow you to export Flash content as HTML5.
- The media players do not support Media Streaming Extensions (MSE).
- Certain video streaming sites use codecs that are unsupported.
- The image size on HTML5 pages is limited to 1920x1080x32bpp. The media players will fail to display pages that contain images that are larger than this restriction.
- Avoid loading or referencing sites that load all content into one very large HTML page, that requires a large amount of physical and virtual memory.

  Note: Beginnning in Release 6.0 with DMP firmware version 6.2.105, you can mitigate requirements for additional memory by allocating virtual memory from the SSD with the following code:

  ```
  v=createobject("rovirtualmemory")
  v.AddSwapFile({ filename: "SSD:/swapfile", megabytes: 2000 })
  ```

  If the specified file does not exist then it will be created. Be aware that this is a very slow operation, but only needs to be done once as long as you do not need to change the size.

- Do not use warping animations when displaying a new piece of content. This does not work well with video, because the hardware decoders require rectangular video content.

Guidelines for Creation of HTML5 Pages

Follow these guidelines when creating HTML5 pages:

- Be sure that the HTML5 page has the same aspect ratio as your signage display.

  If you are displaying HTML5 content in a widget that is smaller than the screen, fit the page to the same aspect ratio as the widget.

- Use a master Div aligned to 0,0 when building an HTML page. This will ensure correct alignment.
- Be aware of page refresh behavior—Every time a page is refreshed, the player will retrieve all page elements again (without caching them). If the page is being hosted on a remote server, the page elements will be loaded as they arrive over the connection, resulting in poor aesthetics for pages that are frequently refreshed.

  For pages that will be refreshed often, the best practice is to include code (for example, JavaScript, Ajax, other) to ensure that only dynamic elements on the page will be reloaded when the page is refreshed.

- If you will be using the web page for digital signage, you might consider hiding scrollbars.

  To hide scrollbars, you can add the following snippet to your CSS code:

  ```
  ::-webkit-scrollbar { width: 0px; height: 0px;
  background: black;}
  ```

HTML5 for Touchscreens

For proper touchscreen operation with HTML5 pages, be sure that your touchscreen devices are human interface device (HID)-compliant and are using standard HID drivers.

Note: Some manufacturers claim support for HID while using custom drivers. Be sure that standard HID drivers are used.

Animations and Add-on Libraries

This section provides some general rules about support for animations and add-on libraries for the WebKit engine on the media players, including:

- JavaScript Animations, page 55
- Vector Animations, page 55
- Bitmap Animations, page 55
- CSS Transforms, page 55
- Add-on Libraries, page 56
- Push Technology, page 56

Caution: Performance is not guaranteed or tested for external web content. External web content can negatively impact the operation of your DMPs and should be tested.

JavaScript Animations

Animations that use JavaScript timers, including the JQuery .animate() library, do not make an efficient use of GPU resources and are not accurate enough to achieve smooth animations. For this reason, we recommend using CSS animations whenever possible. The JQuery Transit library uses CSS animations and provides an API similar to the .animate() library.

Vector Animations

The SVG protocol should be used to specify vector animations.

Bitmap Animations

Bitmap animations display smoothly when they are 1/3 or less of a 1080p HTML canvas. Setting the canvas size to 720p allows for larger high-quality animations to occupy the screen.

CSS Transforms

All CSS transforms should be specified as WebKit transforms. When performing a transform on a `<div>` or graphics element, we do not recommend specifying the transform in-line.

Animations that use the “top” and “left” properties are rendered using the CPU. We recommend using the `translate()` and `translate3d()` methods instead to offload work onto the GPU, ensuring smoother animations.

The following code shows an example of an effective CSS transform:

```
<style>
.flipme{
-webkit-animation-name:flipon;
-webkit-animation-fill-mode:forwards;
-webkit-animation-iteration-count:1;
-webkit-animation-duration:2s;
}
@-webkit-keyframes flipon
{
0% {-webkit-transform:rotateY(0deg);}
30% {-webkit-transform:rotateY(-90deg);}
100% {-webkit-transform: rotateY(360deg);} image
}
</style>
```

Add-on Libraries

The jQuery and Prototype libraries are supported on the media players. As a general rule, any add-on libraries for animation will work if they use WebKit-based transformations. To determine whether a certain library is compatible, you can look at a non-minified version of the library to see if it uses WebKit-based transforms.

Push Technology

The WebSocket protocol and long polling technique have been tested and proven to work on the media players.

Network Latency Workaround

When the media player loads HTML content from a URL, there might be a delay based on network latency. You can add a preloaded image to mitigate this issue.

Verification of HTML Content in Cisco Vision Dynamic Signage Director

The best practice is to always test your content with Cisco Vision Dynamic Signage Director and the DMP.

- Although you can choose to test your HTML content independently of the Cisco Vision Dynamic Signage Director system, you should be aware that performance varies from browser to browser.
- In addition, the results that you get from independent testing on your laptop is not a guarantee for expected results within Cisco Vision Dynamic Signage Director.

  Your computer has virtual memory and physical memory beyond the capacity of the DMP, as well as differences in graphics processing.

If you want to do high-level testing of renditions only to test basic functionality of HTML features, consider the following guidelines:

- The best practice is to install a version of Chromium browser software on your laptop that matches what is used in the DMP firmware for your release.
- The version of Chromium varies by DMP firmware release.

  DMP firmware for Cisco Vision Dynamic Signage Director Release 6.0 uses Chromium version 45.

Known Issues with Firmware

The following are known firmware issues.

- Rotated 4K Output, page 57
- Time Localization in JavaScript, page 57

Rotated 4K Output

Displaying an HTML page and rotated video while using a 4K output mode will cause the video to glitch. This is the case even if the video is not part of the HTML page

Time Localization in JavaScript

The JavaScript `toLocaleTimeString()` call does not retrieve localized time formats (i.e. 24-hour vs. 12-hour clock): Instead, the hour/minute clock defaults to 24-hour time on the media player. The below code provides a workaround in JavaScript if you would like to display time using a 12-hour clock:

1. Create the following function:

```
function format12Hour(date)
{
var zero = '0';
hh = date.getHours();
mm = date.getMinutes(); ss = date.getSeconds() if((hh % 12) == 0) hh = 12; else
hh %= 12;
// Pad zero values to 00 hh = (zero+hh).slice(-2); mm = (zero+mm).slice(-2);
ss = (zero+ss).slice(-2);
return hh + ':' + mm + ':' + ss + ' ' + ((date.getHours()
< 12) ? 'AM' :
'PM');
}
```

2. Optionally, if you would prefer not to display seconds information, you can replace the above "return" line with the following:

```
return hh + ':' + mm + ' ' + ((date.getHours() < 12) ? 'AM' : 'PM');
```

3. Implement the function in the HTML script as follows:

```
var dateString = (startJSDate.getMonth() + 1) + "/" + startJSDate.getDate();
if (!startDateTime.isDateOnly()) {
dateString += " -- " + format12Hour(startJSDate);
}
```