MOBILE VIDEO PLAYBACK ON THE AT&T WEBSITE

This document provides an overview of the factors affecting video playback in the AT&T website on mobile devices.

Statistics from 2011 provided by Ooyala, a video deployment company, show that web video consumers using mobile devices are more likely than desktop users to watch web video and more likely to finish an entire video. Other mobile video viewing trends are:

- Tablet users spend 30% more time watching video per internet session than desktop computer users.
- Tablet users were twice as likely to complete a video compared to desktop users.
- Tablet users were 2.5 times more likely to watch 75% of a web video compared to desktop users.
- Mobile device users were more likely to complete a video compared to desktop users, internet equipped television users, and gaming console users.
- Smartphone users were twice as likely to watch at least 75% of a web video compared to desktop users.
- The growing popularity of tablets and smartphones means that web video will be consumed in greater quantity as time goes on.
- Mobile devices are likely to supplant desktops for the majority of computing needs for the general population.
- Due to the screen size, video can be more accessible to mobile phone users than text.

Given these trends and the fact that AT&T sells mobile devices, we can safely conclude that video on AT&T's website must reliably play on mobile devices as well as computers.

Unfortunately, even though **computers**, **tablets** and **phones** are all electronic devices capable of computing and communicating with other electronic devices via wired or wireless networks, they have differing characteristics that affect video playback. Identifying **mobile device characteristics** and designing for them is the subject of this paper.

Before identifying the mobile device characteristics, the devices must be differentiated.

By **computers**, we mean traditional computing devices such a desktop computers and laptop computers. Such devices have screens and keyboards. The screen and keyboard of such a device may or may not be physically incorporated into the body of the device.

By **tablets**, we mean electronic devices that tend to be smaller than traditional computing devices and larger than phones. A tablet is small enough to be held with two or even one hand. A tablet has a screen incorporated into its body. The screen is usually the primary user interface. The device usually has no keyboard and a few hardwired buttons. It may provide a way to connect to a keyboard.

By **phones**, we mean electronic devices with a physical form that lends itself to voice and other communication via wired or wireless networks. Phones are smaller than computers and tablets; small enough to be held in one hand. A phone has a few hardwired keys or perhaps an entire keyboard incorporated into its body. A phone has a screen incorporated into its body. The screen area may be between two and ten square inches. The screen may or may not be the primary user interface.

In this paper, we hope to provide information to answer the following questions for mobile devices capable of playing video:
• What characteristics can and should be read when determining how to play a video?
• How can and should those characteristics be read?
• How can and should those characteristics be used to determine how to play a video?

The most critical device characteristic is the **video player**. All other device characteristics contribute to a video player's operating environment and so affect its performance which provides the user video playback experience. Hence, video players will be examined in detail in their own section followed by a section for the other characteristics.

MOBILE VIDEO PLAYERS AND THEIR CHALLENGES

The most popular mobile operating systems are the Apple iOS, Google Android, Nokia Symbian and the RIM BlackBerry OS. With Microsoft's purchase of Nokia, Symbian is expected to be phased out and replaced with the Windows Phone OS around the world. Android and iOS are the fastest-growing. There are many mobile video players available in the Android Market or Apple App Store.

There are hundreds of **FLV players** available, many free or shareware. Popular ones are: RealPlayer, GOM Media Player, UMPlayer, VLC Media Player and FLV Player.

Popular mobile video players, excluding FLV players, are listed below by OS.

Google Android - Android native, Act 1 Video Player, mVideoPlayer, MX Video Player, Plex, RealPlayer Android, VPlayer, MoboPlayer, Mobile QQ, RockPlayer

Apple iOS - Apple native, Azul Media Player, CineXPlayer, It's Playing, VLC

Windows Phone OS - Windows Media Player, Media Player Classic, MPlayer, GOM Player, VLC, KMPlayer

Blackberry OS - RealPlayer, The Player, FreeSmith, Splash Lite, Best4video Player, Kite Player

No matter which mobile video player is used, their performance is limited by the mobile device. Relative to video playback on desktop or laptop computers, video playback on mobile devices consumes a large portion of the available computing power and can push the CPU and graphics subsystem near their limits. What seem to be a trivial features and easy to play videos on a desktop or laptop computer, can be daunting on a mobile device. Hence, it is critical that a feature or video be matched well with their mobile video player.

While there are encoding techniques to help match a video to a mobile video player, the features that users want, and the performance of those features, are dependent on the coding of the player.

Desirable mobile video player features include, but are not limited to:
- Playback of videos in formats not natively supported
- More elaborate interface providing extra information about the video
- Additional swipe and gesture controls
- Better media management of video playlists and libraries
- Preferences for video and audio enhancement
- Social network connection in real-time

Unfortunately, the first feature in the list, playing videos in formats not natively supported, is a feature in great demand because those videos usually come from a user's personal library or from a friend's library. Poorly matched videos could be run through a converter to create a compatible version, then be transferred to the mobile device. However, users often install a third party player on the mobile device to play the original video format.

It may seem that AT&T is immune to poorly matched videos, but our work with business partners and third party vendors can cause poorly matched videos to get into our system, especially if proper procedure is hurried or not followed at all.

An important parameter in matching videos to video players is video file format.

VIDEO FILE FORMATS

A video format comprises a codec algorithm and a container. A **codec algorithm** specifies how video tools and players compress and decompress video data. A **container** is a data structure used to deliver video, audio, and related content information in a cohesive, predictable form. A container brings assets together. A codec manipulates these assets.

The dominate video formats and their codecs and containers are a part of HTML5 or Flash or both. The primary containers supported by HTML5 and Flash are MPEG-4, Ogg, WebM. HTML5- compatible and Flash-compatible video containers support additional features such as subtitles, chapter information, and meta-data, as well as how all of these data types are synchronised. Certain containers also support digital rights management (DRM), basic 3D rendering options, and hardware rendering support.

In the past, Flash dominated. Though it is now being overtaken by HTML5, it is still widely available. Flash Video has been accepted as the default online video format by many sites like YouTube, Hulu, VEVO, Yahoo! Video, metacafe, Reuters.com, and other news providers.

Flash Video is a container file format. There are two Flash Video file formats: **FLV** and **F4V.** Flash Video content may also be embedded within SWF files. F4V is based on the ISO base media file format. FLV files usually contain material encoded with codecs following the Sorenson Spark or VP6 video compression formats. The Flash Player supports FLV, F4V and H.264 video.

HTML5 video is a set of elements and additions to the JavaScript DOM with the primary element being the video element. There is basic HTML5 video feature compatibility in the following browsers and mobile operating systems for their current and future versions:
IE 9.0, Firefox 3.5, Safari 3.1, Chrome 3.0, Opera 10.5, iOS 1.0, Android 2.0.

The HTML5 specification does not rigidly define all aspects of video delivery. This allows video players to support different video codecs and, as of this writing, no format is supported by all versions of all browsers. For now, a video server must transcode a video into an appropriate format on the fly or separate files must be created for each video format. However, the web development community is converging on a common HTML5 video format, i.e., WebM or H.264, and 80% of H.264 video can be played in HTML5 video element.

With modern browsers, it's possible to reach all HTML5 users with a combination of MPEG-4, WebM and Ogg videos. It is hoped that the rapid adoption of HTML5 and its video element will accelerate this convergence. At this time, over 50% of all browsers in use, mobile and otherwise, are HMTL5 compatible. This will go up as more users upgrade their browsers.

POPULAR VIDEO PLAYERS

1. Video for Everybody - uses HTML5's ability to move to the next supported object if the prior object fails. Similar to the example above, it uses the <video> element to enclose each supported HTML5 video container source, and follows the collective <source> elements with Flash markup. The result is that the browser tries to display each successive HTML5 source and, if all fail, tries to show the video using a Flash player. Failing that, a static image is displayed. Uses no JavaScript. Won't work on Android OS prior to version 2.3 (Gingerbread). There is a Video For Everybody Generator, which assists in generating the code.

2. VideoJS - is a free, open source HTML5 Video Player that falls back to Flash or the alternate video playback technology of your choice, such as Silverlight or QuickTime. It starts with embedded code based on Video for Everybody, which includes basic playback functionality such as autoplay and preload. Uses JavaScript to fix cross-browser inconsistencies, add support for the HTML5 tags in older versions of IE, and add features such as fullscreen display and subtitles. For the dependency-conscious, its use of JavaScript is library-independent. Provides a consistent API and UI. The comprehensive API is easy to use and the UI is managed with easily edited HTML and CSS that affects not only JavaScript but also the lightweight Flash-based player provided. As an open source player, anyone can alter the code and contribute updates to forks on GitHub. The main files are also hosted meaning that no downloads are required, the code is always updated when new features, devices, and platforms are supported, and the files may even already be cached on your visitors' systems. And, unlike some hosted solutions, if you prefer to download a copy of the player, say for use without an internet connection (or to eliminate reliance on any server other than your own), you can do that, too. VideoJS has a five-second setup, includes a testing demo on the home page that lets you switch between players to check out the consistent UI, and can also be used as a WordPress plug-in.

3. JW Player - is the oldest and most powerful video player in this list. Has robust HTML5 features and features supported by the Flash Player, like progressive download and RTMP streaming, adaptive bitrate and DVR support. It also features native support for several content delivery networks (CDNs) including Akamai, Amazon CloudFront, BitGravity, CDNetworks, Highwinds, and Limelight. Has flexible skinning options, with many existing skins to choose from. The player has a setup wizard and also has the most diverse CMSintegration options. Open source, has no library dependencies, and has free licenses for non-commercial use and a variety of paid licenses for commercial use.

4. MediaElement.js - includes an HTML5 video and audio player as well as custom Flash and Silverlight players that mimic the HTML5 MediaElement API for support in older browsers. This provides developers with a consistent API, and users with a consistent viewing experience regardless of platform or browser. It supports oft-desired features such as standard play controls, skinning, and fullscreen video, but also has a plug-in architecture that enables more unique features. These include looping, auto translation (provided by Google Translate), postroll (displaying custom HTML after a video finishes), and even a virtual backlight that frames the video with matching colors taken from the video during playback (HTML5 only, inspired by Ambilight by Philips). MediaElement supports the <track> element and a range of accessibility standards including WebVTT. It's open source, hosted on GitHub, and also comes in plug-in form for WordPress, Drupal, Joomla, jQuery, and more. MediaElement relies on jQuery to function.

5. SublimeVideo is a hosted service rather than a distributed player. While you can't download the player for use on your own server, there is a free service plan. It includes unlimited plays but has limited features and displays a branding badge in the lower right corner of the video for the first five seconds or so. Like most of the other options discussed herein, Sublime uses JavaScript to integrate HTML5 and Flash in a graceful fallback scenario. It uses a consistent UI for both HTML5 and Flash, and has Lightbox built in for framing the video. It supports playlist playback, SD/HD switching, and an impressive real-time analytics feature. Because it's hosted in the cloud, the player is always up to date, and use couldn't be simpler. You only need to include one line of code in your page and you're up and running in seconds. SublimeVideo has no other JavaScript library dependencies and is also available as a WordPress plug-in.

MOST USED CONTAINERS

MPEG-4, specifically MPEG-4, Part 14, is a proprietary container controlled by the Moving Picture Experts Group. File extensions .mp4 or .m4v. It has its roots in Apple's QuickTime container, which uses .mov extension. Most common choice for desktop and mobile. Used by the iTunes Mac/Windows/iPhone/iPad/iTouch eco-system. Due to patent and licensing issues, this is slowly changing and the HTML5 initiative is accelerating that change. As a container, MPEG-4 is notable because it supports digital rights management – an essential feature for some content creators, and is also attractive because hardware acceleration is widely available for the format. In addition to its inclusion in the HTML5 spec, Flash Player also supports the MPEG-4 container.

Ogg is a free, open container format maintained by the Xiph.Org Foundation. File extensions .ogv or ogg. Native in Linux OS. Supported in the Mac OS and Windows OS by QuickTime components. The Windows OS also uses a Windows Media Player extension or DirectShow filters to display Ogg content.

WebM was introduced by Google as a vehicle for free, open video and audio content. It uses the .webm file extension and is based on the Matroska container. Hardware acceleration is available for WebM files and future versions of Flash Player will support it.

Flash Video container uses two containers. **FLV** is an older, proprietary format that supports Flash Player versions going back to version 7, introduced in 2003. Flash Player version 9 update 3 introduced support for the **F4V** format, which is based on MPEG-4. Flash Player does not rely on file extensions to play compatible video files, but the extensions commonly used for each container are .flv and .f4v, respectively. Flash Player 10.2 and later supports hardware acceleration for Flash Video.

MOST USED CODECS

Codecs are algorithms used for encoding assets at authortime and decoding them for playback at runtime using factors such as quality, file size, bandwidth. There are many codecs. The four primary codecs for HTML5 and Flash are: H.264, Theora, VP8, and VP6. H.264 and VP6 charge licensing fees. Theora and VP8 are free.

H.264 - Also referred to as MPEG-4 Part 10 or Advanced Video Coding (AVC). High-quality optimised for low- and high-bandwidth/CPU devices, from phones to Blu-ray players. Its specification includes "profiles," ranging from Baseline to High, that dictate varying degrees of quality and optional features. It also includes scalable profiles that allow a single file to adjust quality based on playback capabilities. It's supported through both software and hardware acceleration and is widely used in mobile devices to high definition video in broadcast, DVD, and similar environments. H.264 is part of the MPEG-4 container and widely used in Flash Video directly or as part of the F4V container.

Theora is a free, open. While other contains can use Theora as a video codec, it is most often associated with the Ogg format. On2 Technologies originally developed what would become Theora, as VP3. Theora was derived from VP3 after On2 released VP3 into the public domain.

VP8 - Also developed by On2 Technologies, VP8 is known for quality similar to the H.264 High profile, but with a low decoding complexity similar to the H.264 Baseline profile. Google acquired VP8 from On2 to become the video codec of its WebM container. Google provided an irrevocable promise not to enforce its related patents, making VP8 royalty-free, making it an attractive alternative to H.264. It's typically supported by software encoding and decoding, but hardware acceleration is already in use and growing.

VP6 is most commonly used in the FLV Flash Video Format. It's a reasonably high quality codec that also includes attractive features such as alpha channel support.

OTHER DEVICE CHARACTERISTICS THAT AFFECT MOBILE VIDEO PLAYBACK

The device characteristics described in this section contribute to a video player's operating environment and affect its performance. The characteristics are analyzed according to their role in providing acceptable performance during a **video playback session**.

A video playback session, i.e., a **session**, is a time span in which video information is presented; a video is selected; and the video plays until it is paused, stopped or completed. During a session, a video may also be restarted or replayed. If another video is selected, that is considered the beginning of another session.

We divide the device characteristics that affect performance into three categories.

Category 1: Characteristics of the video playback system of the device
These characteristics define the broadest scope for devices. Though these characteristics provide a good first approximation of how to treat a particular device, they are not always sufficient to provide acceptable video playback. The characteristics are:

a. Web browser engine
b. Web browser
c. Operating system
d. Operating system version

Category 2: Characteristics of video transcoding
These characteristics can be used to further refine the identity of a particular device. How well this identification works depends on how many possible values are used for each characteristic. For example, six values of screen area should provide a more accurate sieve than three values. They have direct effect on playback. The characteristics are:

a. Screen area
b. Screen aspect ratio
c. Screen resolution
d. Screen color resolution

Category 3: Situational characteristics video playback performance
These characteristics decribe the immediate environment in which a browser and video player operate. Characteristics "b" and "d" are static and can be relied on throughout a playback session. The remaining characteristics are dynamic within a playback session. The characteristics are:

a. Total memory
b. Available memory
c. Computing power
d. Nominal communication speed
e. Actual communication speed

The primary source of information for the above characteristics is the user agent string.

Can get the size of the browser window to get the size of screen on a phone.

MOBILE DEVICE SCREEN SIZES

If a video does not fit the dimensions or aspect ratio of a screen, a different sized version of the video should be used or the video should be resized. The video should not be cropped.

Screen size - Actual physical size, measured as the screen's diagonal.

Screen density - The quantity of pixels within a physical area of the screen; usually referred to as dots per inch (dpi). For example, a "low" density screen has fewer pixels within a given physical area, compared to a "normal" or "high" density screen.

Orientation - The orientation of the screen from the user's point of view. This is either landscape or portrait, meaning that the screen's aspect ratio is either wide or tall, respectively. Be aware that not only do different devices operate in different orientations by default, but the orientation can change at runtime when the user rotates the device.

Resolution - The total number of physical pixels on a screen. When adding support for multiple screens, applications do not work directly with resolution; applications should be concerned only with screen size and density, as specified by the generalized size and density groups.

Density-independent pixel (dp) - A virtual pixel unit that you should use when defining UI layout, to express layout dimensions or position in a density-independent way. The density-independent pixel is equivalent to one physical pixel on a 160 dpi screen, which is the baseline density assumed by the system for a "medium" density screen. At runtime, the system transparently handles any scaling of the dp units, as necessary, based on the actual density of the screen in use. The conversion of dp units to screen pixels is: px = dp * (dpi / 160). For example, on a 240 dpi screen, 1 dp equals 1.5 physical pixels. You should always use dp units when defining your application's UI, to ensure proper display of your UI on screens with different densities.

An application achieves **density independence** when it preserves the physical size from the user's point of view of visual elements when displayed on screens with different densities. Maintaining density independence is important because, without it, a visual element such as a button, appears physically larger on a low density screen and smaller on a high density screen. Such density-related size changes can cause problems in your application layout and usability.

When screens are considered, two factors are taken into account when it comes to measurements: 1) the size, and 2) the resolution. The size is generally measured diagonally in inches, and the resolution is on the number of pixels displayed on the screen.

Screen resolution is frequently referred to as a modification of the classic VGA resolution, which is 640×480 pixels. Years before, the size of mobile screens used to be 128 x 128. Today, the biggest screen size for mobile phones is 800 x 480 (WVGA). The smaller screens have a portrait orientation, while the large screens have landscape orientation. Today, a lot of mobile phones can change orientation. Most of these mobile phones have a dominant overall screen size of 240 x 320 or QVGA.

Today, however, a lot of companies are beginning to refer to resolution as a fraction of 1920×1080 pixels, or full high definition.

Here's an example of how companies today refer to resolution:
QVGA: quarter VGA (240×320 pixels)
HVGA: half VGA (320×480 pixels)
WVGA: wide VGA (480×800 pixels)
FWVGA: full wide VGA (480×854 pixels)
nHD: one-ninth high definition (360×640 pixels)
qHD: one-quarter high definition (540×960 pixels)

The most popular mobile screen dimensions are:
96X65
128X128
176X208/220
240X320
320X480
640X200/360/480
800X352/400/480

This shows the the most significant screen sizes, from the smallest to the largest. Over the years the relative screen size difference has increased. The difference between the smallest (128 x 128) and the largest (800 x 480) is now a factor of 23. That means the largest screen is 23 times bigger than the smallest one. You can see that the smaller screens have a portrait orientation and the large screens have a landscape orientation. Between them are the phones that can change orientation, they can work in both landscape and portrait. 240 x 320 is the dominant screen size overall.

Most of the devices uses 3–3.5 inch screens, something that does not leave much space for a keyboard. The phones are either clamshell or pure touch. They are also fairly similar in physical size. Phone screen sizes has had a tendency to come in pairs. Each manufacturer had their own variation on large-screen high-end models and small-screen low-end" models.

Manufacturer	small screen	big screen
Sony Ericsson	128 x 160	176 x 220
Nokia	128 x 128	176 x 208
Samsung	128 x 160	176 x 220
Siemens	130 x 130	132 x 176

USER AGENT STRING

The User-Agent string format is currently specified by Section 14.43 of RFC 2616 (HTTP/1.1) The format of the User-Agent string in HTTP is a list of product tokens with optional comments. The "most important" product token is listed first.

For example if your product were called WikiBrowser, your user agent string might be WikiBrowser/1.0 Gecko/1.0. The parts of this string are as follows:
Product name and version - WikiBrowser/1.0
Layout engine and version - Gecko/1.0

The User-Agent request-header field contains information about the user agent originating the request. This is for statistical purposes, the tracing of protocol violations, and automated recognition of user agents for the sake of tailoring responses to avoid particular user agent limitations. User agents should include this field with requests. The field can contain multiple product tokens and comments identifying the agent and any subproducts which form a significant part of the user agent. By convention, the product tokens are listed in order of their significance for identifying the application.

Many web servers were configured to only send web pages that required advanced features to clients that were identified as some version of Mozilla. For this reason, most Web browsers use a User-Agent value as follows:
Mozilla/[version] ([system and browser information]) [platform] ([platform details]) [extensions].

For example, Safari on the iPad has used the following:
Mozilla/5.0 (iPad; U; CPU OS 3_2_1 like Mac OS X; en-us) AppleWebKit/531.21.10 (KHTML, like Gecko) Mobile/7B405

The tokens of this string are as follows:
Mozilla/5.0 - Previously used to indicate compatibility with the Mozilla rendering engine
(iPad; U; CPU OS 3_2_1 like Mac OS X; en-us) - Details of the system in which the

browser runs
AppleWebKit/531.21.10 - The platform the browser uses
(KHTML, like Gecko) - Browser platform details
Mobile/7B405 - This is used by the browser to indicate specific enhancements that are available directly in the browser or through third parties.

CONCLUSION

It is clear that there are many factors affecting video playback in the AT&T website on mobile devices. Certain ones, like the introduction of new devices, change from week to week. To ensure that all videos play on all devices will require a way to track these factors and how they change over time and a timetable for adapting AT&T's video playback mechanisms.

REFERENCES

• **WEBSITE: Advanced Distributed Learning (ADL) Initiative Mobile Learning Handbook**
https://sites.google.com/a/adlnet.gov/mobile-learning-guide/home

The Advanced Distributed Learning (ADL) Initiative involves multi-national groups from industry, academia, and government to define learning industry standards and develop tools and content to those standards. ADL is sponsored by the Office of the Under Secretary of Defense for Personnel and Readiness (OUSD P&R).

• **BOOK: Video Compression for Flash, Apple Devices and HTML5**
http://www.amazon.com/Video-Compression-Flash-Apple-Devices/dp/0976259508/ref=sr_1_1?s=books&ie=UTF8&qid=1342467093&sr=1-1&keywords=encoding+video+for+mobile+devices

• **ARTICLE: How to Choose an Enterprise Video Encoder**
http://www.streamingmedia.com/Articles/Editorial/Featured-Articles/How-to-Choose-an-Enterprise-Video-Encoder-83037.aspx

• **ARTICLE: Recent Statistics Show that Mobile Device Users Watch Video Longer**
http://www.fairfaxvideostudio.com/blog/recent-statistics-show-that-mobile-device-users-watch-video-longer.cfm

• **ARTICLE: Supporting Multiple Screens**
http://developer.android.com/guide/practices/screens_support.html

• **SPECIFICATION: Device Adaptation**
http://www.w3.org/2012/05/mobile-web-app-state/#Device_Adaptation

• **STATISTICS: Ooyala - Enterprise video deployment technologies**
http://www.ooyala.com/

- **Chart: Comparison of container formats**
http://en.wikipedia.org/wiki/Comparison_of_container_formats