Creating HLS VOD Content

MPEG-DASH is one way to create adaptive bitstream playback; but it's not the only one. In this post we'll explore Apple's HLS (HTML Live Streaming) a direct competitor for MPEG-DASH and see how to build a package ready for Video On Demand.

I will use FFMPEG to encode the video as required in the HLS specification. This is the same tool that I've used to create videos with other codecs.

HLS Creation: Single Rendition

> At WWDC 2017, Apple anounced support for HEVC (H265), letting you work with both types of stream in the same play list.
>
> The technologies to implement HEVC in HLS is significantly different than what we discuss here so I'll probably do a separate post on it rather than confuse matters. I'm also banking on the fact that H264 has better support in both Apple and non-Apple devices.

HLS works differently than DASH with different formats and parameters. We will use FFMPEG command line tool to create the HLS video. It provides a lot of tools and supports the creation of the necessary files out of the box.

HLS works with Transport Stream segments and has much tighter control over the characteristics of the streams (or renditions in HLS parlance).

Encoding a single rendition of your video looks like this for a H264 product.

```
# Create the directory to store the files
mkdir serenity
# run the command
ffmpeg -i serenity.mp4 \
-vf scale=w=640:h=360:force_original_aspect_ratio=decrease \
-c:a aac -ar 48000 -b:a 96k \
-c:v h264 \
```

```
-profile:v main \
-crf 20 \
-g 48 -keyint_min 48 \
-sc_threshold 0 \
-b:v 800k -maxrate 856k -bufsize 1200k \
-hls_time 4 \
-hls_playlist_type vod \
-hls_segment_filename serenity/360p_%03d.ts \
serenity/360p.m3u8
```

The explanation for each of the parameters in the example are described in the following list:

- `-i serenity.mp4` - set serenity.mkv as input file
- `-vf "scale=w=1280:h=720:force_original_aspect_ratio=decrease"` - scale video to maximum possible within 1280x720 while preserving aspect ratio
- `-c:a aac -ar 48000 -b:a 96k` - set audio codec to AAC with sampling of 48kHz and bitrate of 128k
- `-c:v h264` - set video codec to be H264 which is the standard codec of HLS segments
- `-profile:v main` - set H264 profile to main - this means support in modern devices
- `-crf 20` - Constant Rate Factor, high level factor for overall quality
- `-g 48 -keyint_min 48` - create key frame (I-frame) every 48 frames (~2 seconds) - will later affect correct slicing of segments and alignment of renditions
- `-sc_threshold 0` - don't create key frames on scene change - only according to -g
- `-b:v 800k -maxrate 856k -bufsize 1200k` - limit video bitrate, these are rendition specific and depends on your content type
- `-hls_time 4` - segment target duration in seconds - the actual length is constrained by key frames
- `-hls_playlist_type vod` - adds the #EXT-X-PLAYLIST-TYPE:VOD tag and keeps all segments in the playlist
- `-hls_segment_filename beach/360p_%03d.ts` - explicitly define segments files' names
- `serenity/360p.m3u8` - path of the playlist file - also tells ffmpeg to output

HLS (.m3u8)

Running the code above will produce a set of files:

- One or more transport stream segments with a `.ts` extension. In this case it produced 34 segments for the video I chose
- The playlist with a `.m3u8` format

The playlist for the video looks like this:

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-TARGETDURATION:4
#EXT-X-MEDIA-SEQUENCE:0
#EXT-X-PLAYLIST-TYPE:VOD
#EXTINF:4.004267,
360p_000.ts
#EXTINF:4.004267,
360p_001.ts
#EXTINF:4.004267,
360p_002.ts
#EXTINF:4.004267,
360p_003.ts
#EXTINF:4.004267,
360p_004.ts
#EXTINF:4.004267,
360p_005.ts
#EXTINF:4.004267,
360p_006.ts
#EXTINF:4.004267,
360p_007.ts
#EXTINF:4.004267,
360p_008.ts
#EXTINF:4.004267,
360p_009.ts
#EXTINF:4.004267,
360p_010.ts
#EXTINF:4.004267,
360p_011.ts
```

```
#EXTINF:4.004267,
360p_012.ts
#EXTINF:4.004267,
360p_013.ts
#EXTINF:4.004267,
360p_014.ts
#EXTINF:4.004267,
360p_015.ts
#EXTINF:4.004267,
360p_016.ts
#EXTINF:4.004267,
360p_017.ts
#EXTINF:4.004267,
360p_018.ts
#EXTINF:4.004267,
360p_019.ts
#EXTINF:4.004267,
360p_020.ts
#EXTINF:4.004267,
360p_021.ts
#EXTINF:4.004267,
360p_022.ts
#EXTINF:4.004267,
360p_023.ts
#EXTINF:4.004267,
360p_024.ts
#EXTINF:4.004267,
360p_025.ts
#EXTINF:4.004267,
360p_026.ts
#EXTINF:4.004267,
360p_027.ts
#EXTINF:4.004267,
360p_028.ts
#EXTINF:4.004267,
360p_029.ts
#EXTINF:4.004267,
360p_030.ts
```

```
#EXTINF:4.004267,
360p_031.ts
#EXTINF:4.004267,
360p_032.ts
#EXTINF:4.004267,
360p_033.ts
#EXTINF:3.503733,
360p_034.ts
#EXT-X-ENDLIST
```

HLS Creation: Multiple Renditions And Master Playlist

All the work we did in the previous section was for one rendition. You will definitely want more than one to acomoadte both your low-end, low-bandwidth users as well as those who are watching in the latest 4k iMac on fiber connections. We can create multiple renditions and let the player decide what's the best one to play at any given time of the playback.

This command will create 4 versions of the video along with a playlist for each:

- 360p
- 480p
- 720p
- 1080p

```
ffmpeg -hide_banner -y -i serenity.mp4 \
  -vf scale=w=640:h=360:force_original_aspect_ratio=decrease \
  -c:a aac -ar 48000 -c:v h264 -profile:v main \
  -crf 20 -sc_threshold 0 -g 48 -keyint_min 48 -hls_time 4 -hls_playlist_t
  -bufsize 1200k -b:a 96k -hls_segment_filename serenity/360p_%03d.ts sere
  -vf scale=w=842:h=480:force_original_aspect_ratio=decrease \
  -c:a aac -ar 48000 -c:v h264 -profile:v main \
  -crf 20 -sc_threshold 0 -g 48 -keyint_min 48 -hls_time 4 -hls_playlist_t
  -bufsize 2100k -b:a 128k -hls_segment_filename serenity/480p_%03d.ts ser
  -vf scale=w=1280:h=720:force_original_aspect_ratio=decrease \
```

```
  -c:a aac -ar 48000 -c:v h264 -profile:v main \
  -crf 20 -sc_threshold 0 -g 48 -keyint_min 48 -hls_time 4 -hls_playlist_t
  -bufsize 4200k -b:a 128k -hls_segment_filename serenity/720p_%03d.ts ser
  -vf scale=w=1920:h=1080:force_original_aspect_ratio=decrease \
  -c:a aac -ar 48000 -c:v h264 -profile:v main \
  -crf 20 -sc_threshold 0 -g 48 -keyint_min 48 -hls_time 4 \
   -hls_playlist_type vod -b:v 5000k -maxrate 5350k \
  -bufsize 7500k -b:a 192k -hls_segment_filename serenity/1080p_%03d.ts se
```

This created the media but there's no way for the player to know what's available. To fix this we create a master playlist that tells the player what renditions are available for the video.

```
#EXTM3U
#EXT-X-VERSION:3
#EXT-X-STREAM-INF:BANDWIDTH=800000,RESOLUTION=640x360
360p.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=1400000,RESOLUTION=842x480
480p.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=2800000,RESOLUTION=1280x720
720p.m3u8
#EXT-X-STREAM-INF:BANDWIDTH=5000000,RESOLUTION=1920x1080
1080p.m3u8
```

Note that this playlist doesn't include the stream transport segments. The master list only tells the player where to look for the corresponding renditions.

Yes, this is cumbersome to do by hand and, when also working with DASH packages, it can be hard to manage.

There are different tools to automate the generation of the HSL Renditions and play lists.

Newer versions of Google's Shaka Packager support creating HLS segments and playlists. They also support creating HLS and DASH content with a single command (although this appears to only with single MP4 files).

Assuming that you've already downloaded/compiled the packager, the

command to genererate DASH and HSL look llike this:

```
packager \
  in=h264_baseline_360p_600.mp4,stream=audio,output=audio.mp4,playlist_nam
  in=h264_baseline_360p_600.mp4,stream=video,output=h264_360p.mp4,playlist
  in=h264_main_480p_1000.mp4,stream=video,output=h264_480p.mp4,playlist_na
  in=h264_main_720p_3000.mp4,stream=video,output=h264_720p.mp4,playlist_na
  in=h264_main_1080p_6000.mp4,stream=video,output=h264_1080p.mp4,playlist_
  --hls_master_playlist_output h264_master.m3u8 \
  --mpd_output h264.mpd
```

The result produces the following files:

```
.
├── audio.m3u8
├── audio.mp4
├── h264.mpd
├── h264_1080p.m3u8
├── h264_1080p.mp4
├── h264_1080p_iframe.m3u8
├── h264_360p.m3u8
├── h264_360p.mp4
├── h264_360p_iframe.m3u8
├── h264_480p.m3u8
├── h264_480p.mp4
├── h264_480p_iframe.m3u8
├── h264_720p.m3u8
├── h264_720p.mp4
├── h264_720p_iframe.m3u8
├── h264_baseline_360p_600.mp4
├── h264_main_1080p_6000.mp4
├── h264_main_480p_1000.mp4
├── h264_main_720p_3000.mp4
└── h264_master.m3u8
```

And these can be used to play either DASH or HLS video. We'll tackle playback next.

Playback

I know how to use Shaka Player to play DASH content. It should be possible to use the same player to play both DASH and HLS content although it shouldn't be necessary to play both, the player scripts should take care of cross platform issues.

Another option is to use Video.js with the HSL plugin and the DASH if you need to serve both formats on the same page.

The basic functionality looks like this:

```
<video  id=example-video
        width=600 height=300
        class="video-js vjs-default-skin"
        controls>
  <source
        src="https://example.com/index.m3u8"
        type="application/x-mpegURL">
</video>

<source
        src="https://example.com/index.m3u8"
        type="application/x-mpegURL"><!--
  Put the lines below at the bottom of the page,
  right before the closing body tag
-->pt>
var player = videojs('example-video');
player.play();
</script>
</script>
```

I'm tracking issue 1365 in the Video.js HSL Plugin Repository to track the issue of content not playing in Chrome.

I'm also researching if I can combine DASH and HLS in a single player for either Shaka Player or Video.js.

Encoding Suggestions

Like DASH, HLS allows for multiple renditions of cotnebt at different resolutions. What these resolutions are will depend on your content and your target audience.

Each row presents to bitrate values, one for low motion content like conference presentations and other activities where there is little motion. The other is for the opposite type of video like sports and other high motion activities. The adio bitrate remains constat for both low and high motion videos.

The table is taken from Peer 5's Creating A Production Ready Multi Bitrate HLS VOD stream

Quality	Resolution	bitrate - low motion	bitrate - high motion	audio bitrate
240p	426x240	400k	600k	64k
360p	640x360	700k	900k	96k
480p	854x480	1250k	1600k	128k
HD 720p	1280x720	2500k	3200k	128k
HD 720p 60fps	1280x720	3500k	4400k	128k
Full HD 1080p	1920x1080	4500k	5300k	192k
Full HD 1080p 60fps	1920x1080	5800k	7400k	192k
4k	3840x2160	14000k	18200	192k
4k 60fps	3840x2160	23000k	29500k	192k

References

- Docs
  - HTTP Live Streaming
  - What is HLS Streaming and when should you use it?
  - HLS Authoring Requirements For Apple Devices
  - Microsoft Edge HTML5 Video
- Tutorials
  - Creating A Production Ready Multi Bitrate HLS VOD stream

- Tools
    - Sample script to generate HLS files
    - HLS Stream Creator
- Playback
    - Video.js
    - Video.js DASH Plugin
    - Video.js HSL Plugin