# HLS Live

https://stackoverflow.com/questions/37347894/hls-live-streaming-from-static-files

https://www.toptal.com/apple/introduction-to-http-live-streaming-hls

HLS Live Testing: https://hls-js.netlify.app/demo/

- Video bandwidth comparison for MPEG-1, MPEG-2, MPEG-4 — Usually MPEG-4 is 2-2.5 times smaller than MPEG-2 and 3-4 times smaller than MPEG-1. MPEG-2 is 30% smaller than MPEG-1.
- Simple HLS server in Python using the Flask web framework — In this example, the Flask application has two routes: /live/<stream_id>.m3u8 and /live/<stream_id>/<int:sequence>.ts. The first route uses the ffmpeg command-line tool to convert an RTMP stream into an HLS stream, and returns the contents of the .m3u8 playlist file. The second route returns the contents of individual .ts segments.
  To run this server, you'll need to have the Flask web framework and the ffmpeg command-line tool installed. You can start the server by running the Python script and access the HLS stream by opening http://localhost:5000/live/<stream_id>.m3u8 in a media player that supports HLS.
- Streams from multiple HLS (HTTP Live Streaming) manifest files with multiple resolutions in a 30-second window — There are several classes defined: Segment, Playlist, and HLSManifest. The Segment class has two properties: resolution and url. The Playlist class has a vector of Segment objects and a method addSegment to add segments to the vector. The HLSManifest class has a map of Playlist objects, where the key is the resolution and the value is the Playlist object. It also has a method addPlaylist to add playlists to the map and a method `
- Streams HLS based on two different files as a live service within a 30-second window in C++
- Inserts an SCTE-104/35 AD Marker into a .m3u8 file — There are two differents identifying markers aka SCTE-104, SCTE-35 that created for distinguishing media stream like the original broadcast and advertisement. SCTE-104 is mainly created at SDI Feed and SCTE-35 is created at Encoder.
- Implement an HLS stream in C++ using two different sources based on .m3u8 files — In this example, the code reads the URLs from two different `.m3u8` files `source1.m3u8` and `source2.m3u8`, combines and sorts them, and writes the combined list of URLs to a new `.m3u8` file `stream.m3u8`. The output file can be used as the manifest for an HLS stream.
- An example of an HLS manifest file (.m3u8) that includes a subtitle track — HLS supports subtitle as a part of HLS manifest (.m3u8)
- Resolve MUST fix issues for Measured peak bitrate in HLS with mediastreamvalidator — You need to set buff_size = max_buffer_size * 1.5, min_buffer_size = max_buffer_size * 0.92, avg_buffer_size = max_buffer_size * 0.96, gop=60, fps=30 in ffmpeg to avoid MUST fix issues with Apple meadiastreamvalidator.
- Example HLS Manifest (m3u8)
- SCTE35 Encoder in Python
- SCTE 35 PID(Packet IDentifier)
- PlutoTV HLS Manifest (m3u8) example
- Apple's Media Stream Validator (mediastreamvalidator) to verify HLS — Apple provides HTTP Live Streaming Tools to help you set up an HTTP Live Streaming service. Media Stream Validator(mediastreamvalidator) is a tool to verify if your HLS has no compatibility issue or not. You can download it at https://developer.apple.com/documentation/http_live_streaming/about_apple_s_http_live_streaming_tools
- Ad Marker - CUE/CUE-OUT/CUE-IN and SCTE35
- Burnt-in SRT subtitle based on ffmpeg — ffmpeg provides the straight and easy way to burn-in SRT subtitle. By forece_style, you can customize your subtitle format like font, fontsize,outline,outlinecolor,borderstyle and so on. The disadvantage of burnt-in caption is that can make bad TV experience for those who does not want to see the subtitle on top of the screen. On the contrary, the benefit of the burnt-in subitie is that doesn't require any additional technical tools/modules on the client.
- EXT-X-DISCONTINUITY in HLS — EXT-X-DISCONTINUITY marks a discontinuity between two consecutive segments. Your discontinuity is between segment1.060.ts and file.000.ts. There is no discontinuity between file.000.ts and file.001.ts so no need to re-insert the tag