Media Manipulation Cheat Sheet

By Joseph Medley
Technical Writer

This document shows in order commands needed to get from a raw mov file to encrypted assets packaged for DASH or HLS. For the sake of having a goal to illustrate, I'm converting my source file to a bitrate of 8Mbs at a resolution of 1080p (1920 x 1080). Adjust these values as your needs dictate.

Conversion is done with two applications: Shaka Packager and ffmpeg, version 3.2.2-tessus. Although I've tried to show equivalent operations for all procedures, not all operations are possible in both applications.

In many cases, commands may be combined in a single command line operation, and in actual use would be. For example, there's nothing preventing you from setting an output file's bitrate in the same operation as a file conversion. For this cheat sheet, I will show these operations (among others) as separate commands for the sake of clarity.

Please let me know of useful additions or corrections. Pull requests are welcome.

Display characteristics

```
packager input=myvideo.mp4 --dump_stream_info
```

```
ffmpeg -i myvideo.mp4
```

Technically, ffmpeg always requires an output file format. Calling ffmpeg this way will give you an error message explaining that; however, it lists information not available from the Shaka Packager command.

Demux (split) audio and video

Most of you wouldn't care about this, but Shaka Packager requires demuxing in order to convert.

mp4/Shaka Packager

```
packager input=myvideo.mp4,stream=video,output=myvideo_video.mp4
packager input=myvideo.mp4,stream=audio,output=myvideo_audio.m4a
```

With Shaka Packager you can combine these.

```
packager \
  input=myvideo.mp4,stream=video,output=myvideo_video.mp4 \
  input=myvideo.mp4,stream=audio,output=myvideo_audio.m4a
```

webm/Shaka Packager

```
packager \
  input=myvideo.webm,stream=video,output=myvideo_video.webm \
  input=myvideo.webm,stream=audio,output=myvideo_audio.webm
```

mp4/ffmpeg

```
ffmpeg -i myvideo.mp4 -vcodec copy -an myvideo_video.mp4
ffmpeg -i myvideo.mp4 -acodec copy -vn myvideo_audio.m4a
```

webm/ffmpeg

```
ffmpeg -i myvideo.webm -vcodec copy -an myvideo_video.webm
ffmpeg -i myvideo.webm -acodec copy -vn myvideo_audio.webm
```

Change characteristics

Bitrate

For ffmpeg, I can do this while I'm converting to mp4 or webm.

```
ffmpeg -i myvideo.mov -b:v 350K myvideo.mp4
ffmpeg -i myvideo.mov -vf setsar=1:1 -b:v 350K myvideo.webm
```

File type

Shaka Packager cannot process mov files and hence cannot be used to convert files from that format.

mov to mp4

```
ffmpeg -i myvideo.mov myvideo.mp4
```

mov to webm

When converting a file to webm, ffmpeg doesn't provide the correct aspect ratio. Fix this with a filter (`-vf setsar=1:1`).

```
ffmpeg -i myvideo.mov -vf setsar=1:1 myvideo.webm
```

Resolution

```
ffmpeg -i myvideo.webm -s 1920x1080 myvideo_1920x1080.webm
```

Synchronize audio and video

To ensure that audio and video synchronize during playback insert keyframes.

```
ffmpeg -i myvideo.mp4 -keyint_min 150 -g 150 -f webm -vf setsar=1:1 out.webm
```

Codec

You might have an older file whose codec you want to update. The examples below change codecs, but do not demux.

mp4/H.264

```
ffmpeg -i myvideo.mp4 -c:v libx264 -c:a copy myvideo.mp4
```

Audio for an mp4

```
ffmpeg -i myvideo.mp4 -c:v copy -c:a aac myvideo.mp4
```

webm/VP9

```
ffmpeg -i myvideo.webm -v:c libvpx-vp9 -v:a copy myvideo.webm
```

Audio for a webm

```
ffmpeg -i myvideo.webm -v:c copy -v:a libvorbis myvideo.webm
ffmpeg -i myvideo.webm -v:c copy -v:a libopus myvideo.webm
```

Package

DASH/MPD

Dynamic Adaptive Streaming over HTTP is a web-standards-based method of presenting video-on-demand for the web.

```
packager \
  input=myvideo.mp4,stream=audio,output=myvideo_audio.mp4 \
  input=myvideo.mp4,stream=video,output=myvideo_video.mp4 \
  --mpd_output myvideo_vod.mpd
```

HLS

HTTP Live Streaming (HLS) is Apple's standard for live streaming and video on demand for the web.

```
ffmpeg -i myvideo.mp4 -c:a copy -b:v 8M -c:v copy -f hls -hls_time 10 \
       -hls_list_size 0 myvideo.m3u8
```

Clear Key Encryption

Create a key

You can use the same method to create a key for both DASH and HLS. Do this using openssl. The following will create a key made of 16 hex values.

```
openssl rand -out media.key 16
```

This command creates a file with white space and new line characters, which are not allowed by Shaka Packager. You'll need to open the key file and manually remove all whitespace including the final carriage return.

Encrypt

For the **-key** flag use the key created earlier and stored in the media.key file. However, when entering it on the command line, be sure you've removed its whitespace. For the **-key_id** flag

repeat the key value.

```
packager \
  input=myvideo.mp4,stream=audio,output=glocka.m4a \
  input=myvideo.mp4,stream=video,output=glockv.mp4 \
  --enable_fixed_key_encryption \
  -key INSERT_KEY_HERE -key_id INSERT_KEY_ID_HERE \
```

Create a key information file

To encrypt for HLS you need a key information file in addition to a key file. A key information file has the following format.

```
key URI
key file path
```

For example:

```
https://example.com/media.key
media.key
```

Encrypt for HLS

```
packager \
  'input=input.mp4,stream=video,segment_template=output$Number$.ts,playlist_name=
  'input=input.mp4,stream=audio,segment_template=output_audio$Number$.ts,playlist
  --hls_master_playlist_output="master_playlist.m3u8" \
  --hls_base_url="http://localhost:1000/"
```

This command will accept a key with either 16 or 32 characters.

```
ffmpeg -i myvideo.mov -c:v libx264 -c:a aac -hls_key_info_file key_info myv
```

Widevine Encryption

```
packager \
  input=glocken.mp4,stream=video,output=enc_video.mp4 \
  input=glocken.mp4,stream=audio,output=enc_audio.m4a \
  --enable_widevine_encryption \
  --key_server_url "https://license.uat.widevine.com/cenc/getcontentkey/widevine_
```

```
--content_id "Hex_converted_unique_ID" --signer "widevine_test" \
--aes_signing_key "1ae8ccd0e7985cc0b6203a55855a1034afc252980e970ca90e5202689f94
--aes_signing_iv "d58ce954203b7c9a9a9d467f59839249"
```

All Together Now

DASH/webm with Shaka Packager

Not all steps are possible with Shaka Packager, so I'll use ffmpeg when I need to.

1. Convert the file type and codec.

   For this command you can use either `liborbis` or `libopus` for the audio codec.

   ```
   ffmpeg -i glocken.mov -c:v libvpx-vp9 -c:a libvorbis -b:v 8M -vf setsar=1:1
   -f webm glocken.webm
   ```

2. Create a Clear Key encryption key.

   You'll need to open the key file and manually remove all whitespace including the final carriage return.

   ```
   openssl rand -out media.key 16
   ```

3. Demux the audio and video, encrypt the new files, and output a media presentation description (MPD) file.

   The `-key` and `-key_id` flags are copied from the `media.key` file.

   ```
   packager \ input=myvideo.webm,stream=video,output=myvideo_video.webm \
   input=myvideo.webm,stream=audio,output=myvideo_audio.webm \ --
   enable_fixed_key_encryption --enable_fixed_key_decryption \ -key
   INSERT_KEY_HERE -key_id INSERT_KEY_ID_HERE \ --mpd_output myvideo_vod.mpd
   ```

DASH/mp4 with Shaka Packager

Not all steps are possible with Shaka Packager, so I'll use ffmpeg when I need to.

1. Convert the file type, video codec and bitrate.

   The default pixel format, yuv420p is used because one isn't supplied in the command line. The app will give you an error message that it is deprecated. I've chosen not to override the default because, though deprecated yuv420p is the most widely supported.

   ```
   ffmpeg -i mymovie.mov -c:v libx264 -c:a aac -b:v 8M -strict -2 mymovie.mp4
   ```

2. Create a Clear Key encryption key.

   You'll need to open the key file and manually remove all whitespace including the final carriage return.

   ```
   openssl rand -out media.key 16
   ```

3. Demux the audio and video, encrypt the new files, and output a media presentation description (MPD) file.

   The `-key` and `-key_id` flags are copied from the `media.key` file.

   ```
   packager \
   input=mymovie.mp4,stream=audio,output=myaudio.m4a \
   input=mymovie.mp4,stream=video,output=myvideo.mp4 \
   --enable_fixed_key_encryption --enable_fixed_key_decryption \
   -key INSERT_KEY_HERE -key_id INSERT_KEY_ID_HERE \
   --mpd_output myvideo_vod.mpd
   ```

Widevine

The two previous examples used Clear Key encryption. For widevine the final two steps are replaced with this.

Everything in this command except the name of your files and the `--content-id` flag should be copied exactly from the example. The `--content-id` is 16 or 32 random hex digits. Use the keys provided here instead of your own. (This is how Widevine works.)

```
packager \
  input=mymovie.mp4,stream=audio,output=myaudio.m4a \
  input=mymovie.mp4,stream=video,output=myvideo.mp4 \
  --enable_widevine_encryption \
  --key_server_url "https://license.uat.widevine.com/cenc/getcontentkey/widevine_
  --content_id "fd385d9f9a14bb09" --signer "widevine_test" \
  --aes_signing_key "1ae8ccd0e7985cc0b6203a55855a1034afc252980e970ca90e5202689f94
  --aes_signing_iv "d58ce954203b7c9a9a9d467f59839249"
```

HLS/mp4

HLS only supports mp4, so first you'll need to convert to the MP4 container and supported codecs. Not all steps are possible with Shaka Packager, so I'll use ffmpeg when I need to.

1. Convert the file type, video codec, and bitrate.

The default pixel format, yuv420p is used because one isn't supplied in the command line. The app will give you an error message that it is deprecated. I've chosen not to override the default because, though deprecated yuv420p is the most widely supported.

```
ffmpeg -i mymovie.mov -c:v libx264 -c:a aac -b:v 8M -strict -2 mymovie.mp4
```

2. Create a Clear Key encryption key.

   You'll need to open the key file and manually remove all whitespace including the final carriage return.

   ```
   openssl rand -out media.key 16
   ```

3. Create a key information file

   ```
   packager \
   'input=input.mp4,stream=video,segment_template=output$Number$.ts,
   playlist_name=video_playlist.m3u8' \
   'input=input.mp4,stream=audio,segment_template=output_audio$Number$.ts,
   playlist_name=audio_playlist.m3u8,hls_group_id=audio,hls_name=ENGLISH' \
   --hls_master_playlist_output="master_playlist.m3u8" \
   --hls_base_url="http://localhost:1000/" \
   --enable_fixed_key_encryption --enable_fixed_key_decryption \
   -key INSERT_KEY_HERE -key_id INSERT_KEY_ID_HERE \
   ```

Last updated July 2, 2018.