NETINT

Encoding and Packaging x265/HDR with FFmpeg, Bento4 and mp4box

Jan Ozer

Marketing

NETINT

jan.ozer@netint.com

Encoding HEVC

- About HEVC
- Producing HEVC using FFmpeg and x265
- Packaging with open-source options
- HDR 0utputs (HDR10/HDR10+)

My Assumptions

- New to FFmpeg/x265, not video encoding
- Will explain operation, not fundamental concepts underlying operation
  - Not, here's what CBR is and why/when you want it
  - Rather, here's how you produce a CBR stream
- You have PDF – don't need to explain details
  - Try to focus on key concepts

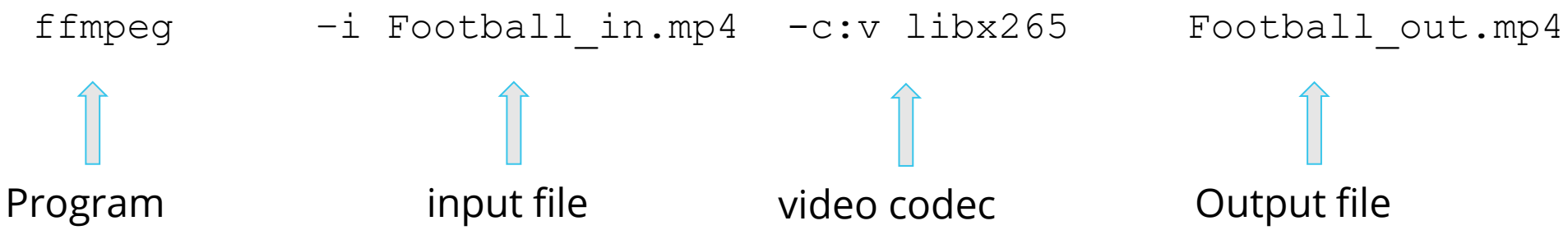
About x265

- HEVC is a standard with many compliant HEVC codecs
  - x265, Fraunhofer, MainConcept, Beamr, others (software)
  - NETINT, NVIDIA, Intel, AMD, others (hardware)
  - x265 is the open-source HEVC encoder included with FFmpeg
    - Middle of the road in terms of quality compared to other HEVC codecs
    - Very widely used because it’s open-source

Producing HEVC using FFmpeg and the x265 codec

- Introduction – stream fundamentals
- Default operation
- Single Pass vs Capped CRF
- Profiles
- Presets
- Reference frames

Choosing the Codec With FFmpeg



- Input file: 1080p file in MP4 format
  - H.264 8-bit video
  - AAC audio
- Simple script means that you accept most FFmpeg defaults

Encoding Output - Default

- Codec: x265
  - Data rate: 3 Mbps
  - Bitrate control: average bitrate
  - Key frame: 250
  - Scene change: Yes
  - Resolution: same (1080p)
  - Frame rate: same (29.97)
  - Profile: Main (because 8-bit source)
  - x265 preset: Medium
    - B-frames: preset (4)
    - B-adapt: preset (2)
    - Reference frames preset (3)
- Audio codec: AAC
  - Audio channels: 2 (same)
  - Audio samples: 48 khz (same)
  - Audio bitrate: 128 kbps

Key Takeaway

- This command string is fine for casual encodes
- For ABR streaming need:
  - Bitrate control
  - GOP control
  - Resolution (for encoding ladder)
  - Choose profile
  - Choose preset

Working with x265

```
ffmpeg -i Football_in.mp4 -c:v libx265 -s 1280x720 -preset slow -an
-x265-params keyint=48:min-keyint=48:scenecut=0:ref=5:bframes=3:b-
adapt=2:bitrate=1000:vbv-maxrate=1100:vbv-bufsize=1000:pass=1 -f mp4 NUL && \
```

- Will detail commands later, but recognize that there are two sets of commands
  - **Generic FFmpeg – preset, resolution, audio, null characters (for two-pass)**
    - **Presented as normal FFmpeg commands**
  - **x265-specific commands – must be presented behind `x265-params` switch**
    - **Bitrate control**
    - **GOP control (keyframe, B-frame, reference frame)**
    - **One string of commands, separated by colon, no spaces until finished**
- If you don't put x265 command behind x265-params switch, FFmpeg won't error - it just won't execute those options. So,
  - No bitrate control, no GOP control

Two-Pass for Windows vs. Two-Pass for Linux/Mac

```
ffmpeg -i Football_in.mp4 -c:v libx265 -s 1280x720 -preset slow -an  -
x265-params keyint=48:min-keyint=48:scenecut=0:ref=5:bframes=3:b-
adapt=2:bitrate=1000:vbv-maxrate=1100:vbv-bufsize=1000:pass=1 -f mp4
NUL && \
```

```
ffmpeg -i Football_in.mp4 -c:v libx265 -s 1280x720 -preset slow -an  -
x265-params keyint=48:min-keyint=48:scenecut=0:ref=5:bframes=3:b-
adapt=2:bitrate=1000:vbv-maxrate=1100:vbv-bufsize=1000:pass=1 -f mp4
/dev/null
```

Bitrate Control

```
ffmpeg -i Football_in.mp4 -c:v libx265 -x265-params
bitrate=3000:vbv-maxrate=6000:vbv-bufsize=6000
Football_1080p.mp4
```

- **`bitrate`** – target bitrate
- **`vbv-maxrate`** – max bitrate
  - VOD – typically 200% constrained VBR (as shown)
  - Live – typically 100% CBR (not addressed)
- **`vbv-bufsize`** – VBV buffer size
  - Impacts quality (slightly) and stream variability (slightly)
  - Typically, 100-200% of target

Two-Pass Operation

```
ffmpeg -y -i Football_in.mp4 -c:v libx265 -x265-params bitrate=3000:vbv-maxrate=6000:vbv-bufsize=6000:pass=1 -f mp4 NUL && \

ffmpeg -i Football_in.mp4 -c:v libx265 -x265-params bitrate=3000:vbv-maxrate=6000:vbv-bufsize=6000:pass=2 Football_1080p_3M_2p.mp4
```

- Line 1:
  - **-y** - overwrite existing log file (otherwise overwrite error)
  - **pass=1** - first pass, no output file
  - **-f mp4** - output format second pass
  - **NUL** - creates log file cataloguing encoding complexity (can name log file if desired)
  - **&& ** - run second pass if first successful

- Line 2:
  - **pass=2** - find and use log file for encode
  - **Test_1080p_2P.mp4** - output file name
  - Note - all commands in first pass must be in second pass; can add additional commands in second line (more later)

Working with x265 – Other Common Commands

```
ffmpeg -i Football_in.mp4 -c:v libx265 -s 1280x720 -an -x265-params keyint=60:min-keyint=60:scenecut=0:ref=3:bframes=3:b-adapt=2:bitrate=2400:open-gop=0:pass=1 -f mp4 NUL && \

ffmpeg -i Football_in.mp4 -c:v libx265 -s 1280x720 -an  -x265-params keyint=60:min-keyint=60:scenecut=0:ref=3:bframes=3:b-adapt=2:bitrate=2400:vbv-maxrate=4800:vbv-bufsize=4800:open-gop=0:pass=2 football_720p_2M.mp4
```

- Commands (outside params)
  - **`-s`** – resolution
  - **`-an`** – no audio

I typically don't specify b-frames and ref frames; here for informational purposes only.

- Commands (x265-params)
  - **`bitrate/vbv-maxrate/vbv-bufsize`**
  - **`keyint`** – GOP size/keyframe interval
  - **`min-keyint=60:scenecut=0`** – fixed interval (for ABR)
  - **`bframes`** – set number of b-frames
  - **`b-adapt`** – 2 ensures max b-frames
  - **`ref`** – set number of reference frames
  - **`open-gop=0`**– closes GOP (for ABR)

2-Pass VBR vs. Capped CRF

Should be: 4:38

- Issue: Two-pass encoding roughly doubles the encoding time/cost. Is CRF a better alternative?
- What is CRF
- What is Capped CRF
- How does it compare to 2-pass VBR?

Constant Rate Factor (CRF) Encoding

- CRF is an encoding mode in some but not all codecs
  - Yes – x264, x265, VP9, AV1
  - No – MainConcept H.264/HEVC codecs, AWS Elemental codecs
- VBR and CBR attempts to hit the specified data rate
  - Adjusts ***quality*** to meet the data rate (quality is secondary)
- CRF encodes deliver specified quality (on a scale from 1-51)
  - Adjusts ***data rate*** to meet quality level (data rate is secondary)

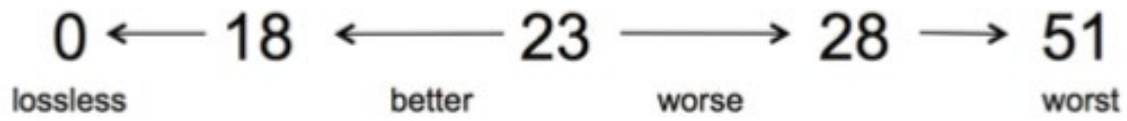
Producing CRF Encoded Video

```
ffmpeg -i Test_1080p.mp4 -c:v libx265 -crf 23 Test_CRF23.mp4
```

Commands

`-crf –` quality setting. Most producers use from 19 – 235 with **lower** numbers delivering **better** quality

- Potential issue – no data rate control – can’t be used for streaming



What is Capped CRF?

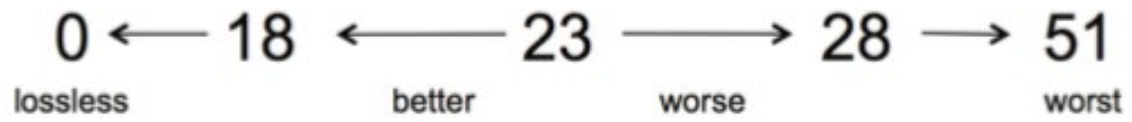
- Capped CRF applies CRF ***and*** a Bitrate Cap (Maximum)
  - Delivers specified quality with a maximum bitrate
  - Single pass
  - Used by some companies as a per-title encoding technique

Producing Capped CRF

```
ffmpeg -i Test_1080p.mp4 -c:v libx265 -crf 23 -maxrate 3500k -bufsize 7000k
Test_CRF23_3500.mp4
```

Commands

- `-crf` – quality setting
- `-maxrate 3500k` – capping the data rate
- `-bufsize 7000k` – capping the buffer size



- Talking head
  - CRF sets quality @ 1 Mbps (saving bandwidth)
  - VBR @ 3.5 Mbps
- Soccer match
  - CRF – cap controls quality @ 3.5 Mbps
  - VBR @ 3.5 Mbps

Capped CRF vs. 2-Pass VBR (9 Short Files)

Let’s compare

- Encoding time
- Bitrate
- Overall VMAF quality (harmonic mean)
- Low-Frame VMAF quality (potential for transient issues)
- VMAF Standard Deviation (quality variability)

Test Command Strings – For the Record

Two-Pass CBR

```
FFmpeg -y -i football.mp4 -c:v libx265 -an -preset veryslow -x265-
params keyint=50:min-keyint=50:scenecut=0:bitrate=2400:open-
gop=0:pass=1 -f mp4 NUL && \

ffmpeg -y -i football.mp4 -c:v libx265 -an -preset veryslow -x265-
params keyint=50:min-keyint=50:scenecut=0:bitrate=2400:vbv-
maxrate=4800:vbv-bufsize=4800:open-gop=0:pass=2 football_baseline.mp4
```

Capped CRF

```
ffmpeg -y -i football.mp4 -c:v libx265 -an -preset veryslow -crf 25 -
x265-params keyint=50:min-keyint=50:scenecut=0:vbv-maxrate=2400:vbv-
bufsize=4800:open-gop=0 football_CRF_25.mp4
```

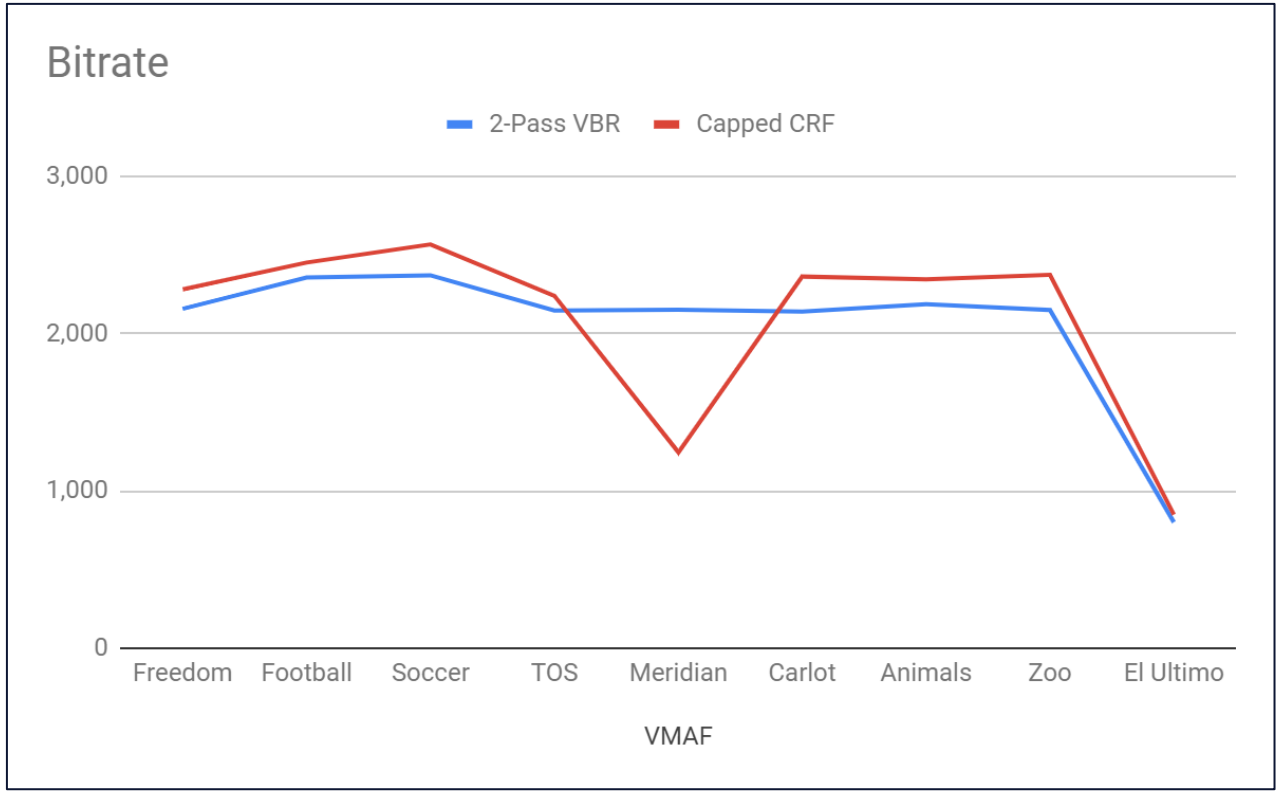
Capped CRF vs. 2-Pass VBR – Encoding Time

Encoding time (9 short files)	
Average - 2-pass VBR	0:03:05
Average - Capped	0:01:33
Time saving	49.64%

- Saves 49.64%, essentially halving your encoding time

Capped CRF vs. 2-Pass VBR – File Size

Bitrate		
	2-Pass VBR	**Capped CRF**
Freedom	2,156	2,280
Football	2,355	2,450
Soccer	2,369	2,565
TOS	2,145	2,239
Meridian	**2,151**	**1,244**
Carlot	2,138	2,361
Animals	2,186	2,343
Zoo	2,149	2,372
El Ultimo	800	849
Average	**2,050**	**2,078**
Exclude Meridian	**2,037**	**2,182**

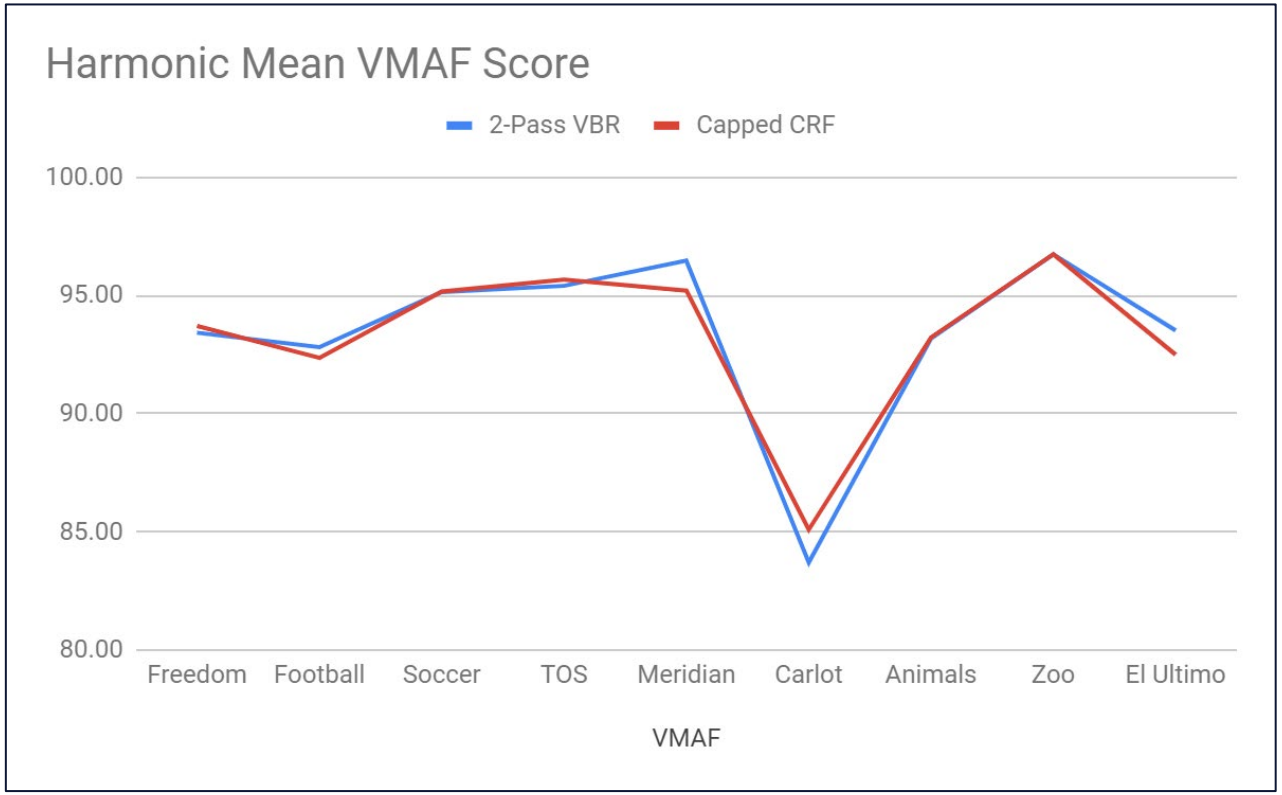


- Close enough for quality comparisons

- All but Meridian are hard-to-encode files, so not leveraging C-CRF's per-title capability
  - Would show much more file savings on easy to encode or even mixed files (or higher rates)

Capped CRF vs. 2-Pass VBR – Overall VMAF Quality

VMAF - Harmonic Mean		
	2-Pass VBR	**Capped CRF**
Freedom	93.42	93.71
Football	92.81	92.36
Soccer	95.14	95.16
TOS	95.40	95.67
Meridian	96.47	95.20
Carlot	83.70	85.09
Animals	93.18	93.22
Zoo	96.74	96.74
El Ultimo	93.52	92.49
Average	**93.38**	**93.29**

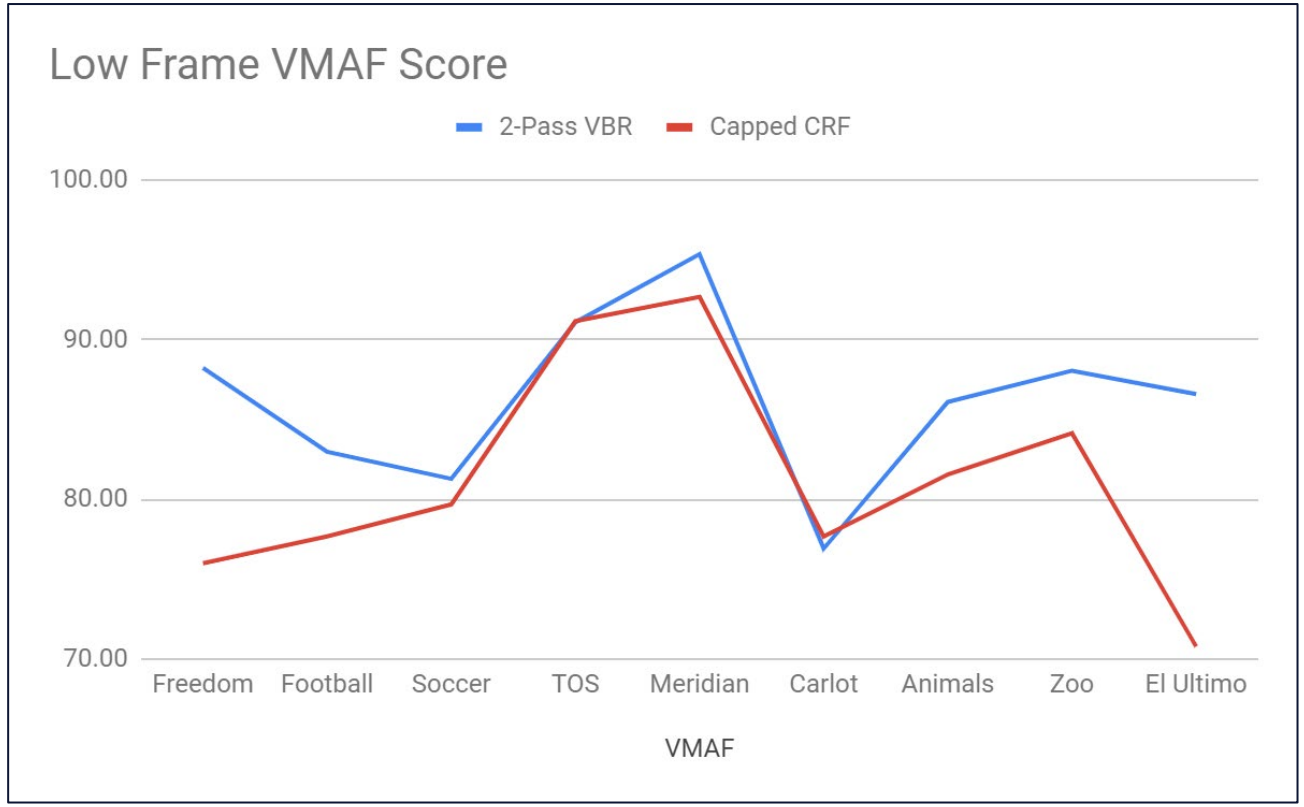


- Very little delta
  - Mostly on Meridian file (1.37 points) but since both over 95 not visible to viewers

- 50% encoding time savings; no real impact on overall quality
  - Looking good so far

Capped CRF vs. 2-Pass VBR – Low-Frame Quality

VMAF - Low Frame		
	2-Pass VBR	**Capped CRF**
Freedom	88.21	76.00
Football	82.97	77.67
Soccer	81.28	79.67
TOS	91.07	91.14
Meridian	95.33	92.66
Carlot	76.91	77.67
Animals	86.09	81.55
Zoo	88.05	84.14
El Ultimo	86.58	70.78
Average	**86.28**	**81.25**

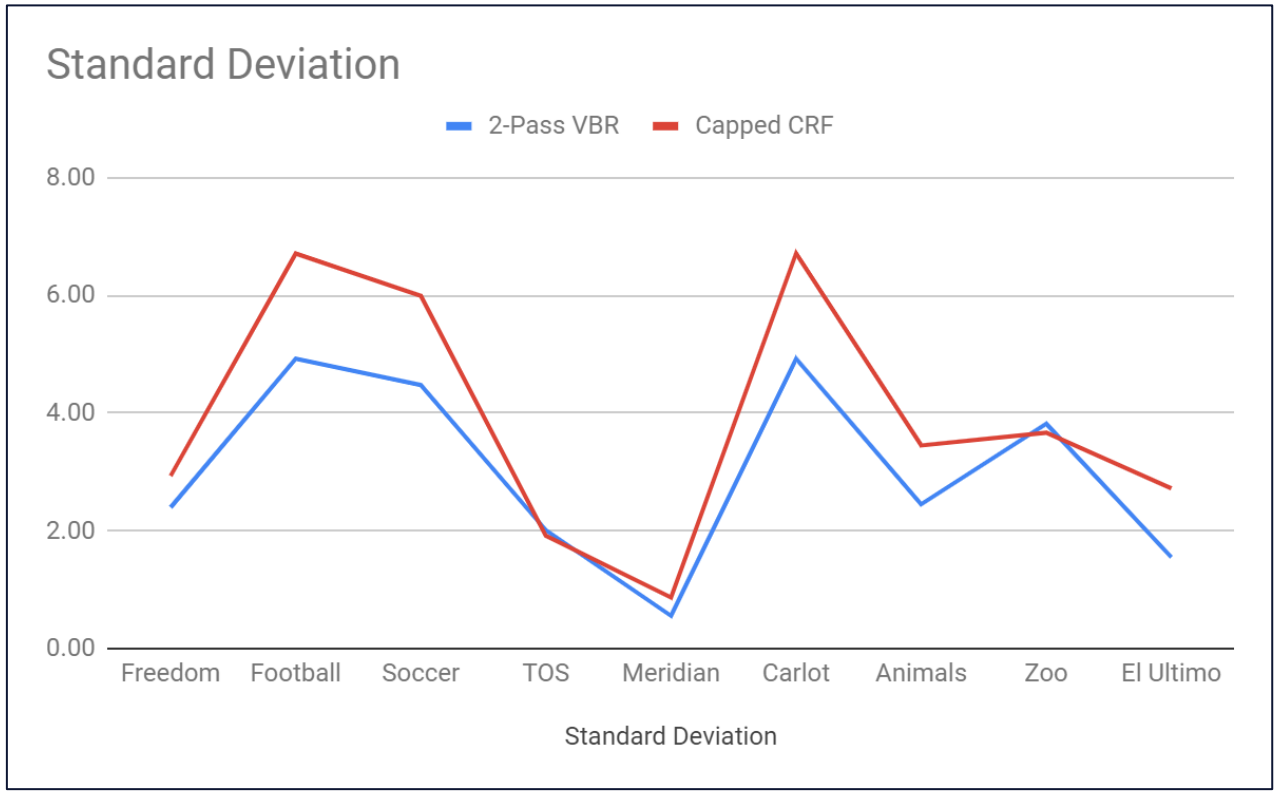


- Over 5-point delta – meaningful

- Capped CRF introduces risk of transient quality problems

Capped CRF vs. 2-Pass VBR – Standard Deviation

Standard Deviation		
	2-Pass VBR	**Capped CRF**
Freedom	2.40	2.93
Football	4.92	6.71
Soccer	4.48	5.99
TOS	2.01	1.92
Meridian	0.56	0.87
Carlot	4.92	6.71
Animals	2.45	3.45
Zoo	3.82	3.67
El Ultimo	1.55	2.72
Average	**3.01**	**3.89**



- ~30% increased quality variability

- Capped CRF introduces higher quality variability, which reduces QoE

Capped CRF vs. 2-Pass VBR - Summary

- The scorecard
  - Encoding time - **cuts by 50%**
  - Bitrate - **should drop significantly in mixed data set**
  - Overall VMAF – **nearly identical**
  - Low-Frame VMAF quality - **meaningfully lower**
  - VMAF Standard Deviation - **meaningfully higher - worse**

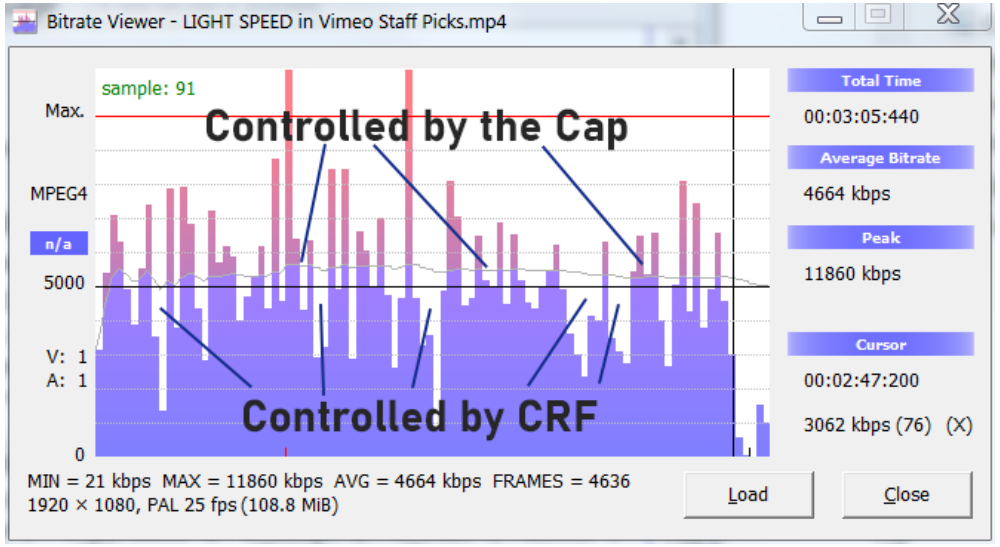
- Observations
  - Consider for volume-centric distribution
    - UGC
    - Mobile video
  - Not a slam dunk for premium content

Key Concept

- When evaluating any encoding configuration (preset, codec, encoder, reference frame), always consider:
  - Encoding time (cost)
  - Quality
    - Overall quality – harmonic mean value
    - Low-frame quality – measures likelihood of transient quality issues
    - Standard deviation – measures quality variability

More on Capped CRF

- Saving on H.264 Encoding and Streaming: Deploy Capped CRF
  - bit.ly/SLC_Capped_CRF
- Choosing the Optimal CRF Value for Capped CRF Encoding
  - bit.ly/SLC_Capped_CRF



Here, CRF value sets quality

	CRF 19	
	Data Rate	VMAF
Easy clips		
Big Buck Bunny	4,001	97.08
Tutorial	1,496	97.14
Talkinghead	4,992	96.47
Average	**3,496**	**96.90**
Delta		
Hard clips		
Elektra	4,767	95.10
Football	4,965	93.34
Freedom	4,930	94.82
Haunted	4,975	91.69
Sintel	4,555	96.10
TOS	4,845	96.14
Zoolander	5,024	95.58
Average	**4,866**	**94.68**
Delta		

VMAF score too high

Here, cap sets quality

HEVC Profiles

Should be: 4:46

- What profiles are and what they do
- Choosing a profile in FFmpeg

What HEVC Profiles are and Why They Exist

- Profiles use different encoding features to enable varying levels of quality and decoding complexity
  - But, with HEVC version 1, main difference is bit depth
  - Very little quality differential

Feature	Version 1	
	Main	Main 10
Bit depth	8	8 to 10
Chroma sampling formats	4:2:0	4:2:0
4:0:0 (Monochrome)	No	No
High precision weighted prediction	No	No
Chroma QP offset list	No	No
Cross-component prediction	No	No
Intra smoothing disabling	No	No
Persistent Rice adaptation	No	No
RDPCM implicit/explicit	No	No
Transform skip block sizes larger than 4x4	No	No
Transform skip context/rotation	No	No
Extended precision processing	No	No

https://en.wikipedia.org/wiki/High_Efficiency_Video_Coding

Choosing the HEVC Profile

- Profile is ***automatically*** set by input format
  - If 8-bit input, FFmpeg produces Main output
  - If 10-bit input, FFmpeg produces Main10 output
- Can change input format to produce other profile
  - If 10-bit input, add `-pix_fmt yuv420p` to change to 8-bit input, and produce Main output
  - If 8-bit input, add `-pix_fmt yuv420p10le` to change to 10-bit input and produce Main10

```
ffmpeg -i 10-bit-input.mov -c:v libx265 -pix_fmt yuv420p ... Main_output.mp4

ffmpeg -i 8-bit-input.mov -c:v libx265 -pix_fmt yuv420p10le ...
Main10_output.mp4
```

Main or Main10?

- Absolute requirements
  - HDR – Main10
  - HLS/DASH – either
- Otherwise stay in your lane
  - If 8-bit – encode Main – no quality benefit to using Main 10
  - If 10-bit – encode Main10
    - Very minor compatibility improvement for Main
    - No significant quality loss for Main

Working with x265 Presets

Should be: 4:48

- x265 presets – what they do and how they work
- Visualizing the cost/encoding time tradeoffs
- Designating a preset in FFmpeg

What Presets are and Why They Exist

- Created by x265 developers to provide simple choices that trade off encoding time for encoding quality
- Here are the presets and the configuration options for each
- If you don't specify a preset, FFmpeg uses the medium preset

0. ultrafast
1. superfast
2. veryfast
3. faster
4. fast
5. medium **(default)**
6. slow
7. slower
8. veryslow
9. placebo

preset	0	1	2	3	4	5	6	7	8	9
ctu	32	32	64	64	64	64	64	64	64	64
min-cu-size	16	8	8	8	8	8	8	8	8	8
bframes	3	3	4	4	4	4	4	8	8	8
b-adapt	0	0	0	0	0	2	2	2	2	2
rc-lookahead	5	10	15	15	15	20	25	30	40	60
lookahead-slices	8	8	8	8	8	8	4	4	1	1
scenecut	0	40	40	40	40	40	40	40	40	40
ref	1	1	2	2	3	3	4	4	5	5
limit-refs	0	0	3	3	3	3	3	2	1	0
me	dia	hex	hex	hex	hex	hex	star	star	star	star
merange	57	57	57	57	57	57	57	57	57	92
subme	0	1	1	2	2	2	3	3	4	5
rect	0	0	0	0	0	0	1	1	1	1
amp	0	0	0	0	0	0	0	1	1	1
limit-modes	0	0	0	0	0	0	1	1	1	0
max-merge	2	2	2	2	2	2	3	3	4	5
early-skip	1	1	1	1	0	0	0	0	0	0
recursion-skip	1	1	1	1	1	1	1	1	0	0
fast-intra	1	1	1	1	1	0	0	0	0	0
b-intra	0	0	0	0	0	0	0	1	1	1
sao	0	0	1	1	1	1	1	1	1	1
signhide	0	1	1	1	1	1	1	1	1	1
weightp	0	0	1	1	1	1	1	1	1	1
weightb	0	0	0	0	0	0	0	1	1	1
aq-mode	0	0	1	1	1	1	1	1	1	1
cuTree	1	1	1	1	1	1	1	1	1	1
rdLevel	2	2	2	2	2	3	4	6	6	6
rdoq-level	0	0	0	0	0	0	2	2	2	2
tu-intra	1	1	1	1	1	1	1	2	3	4
tu-inter	1	1	1	1	1	1	1	2	3	4
limit-tu	0	0	0	0	0	0	0	4	4	0

http://x265.readthedocs.io/en/default/presets.html

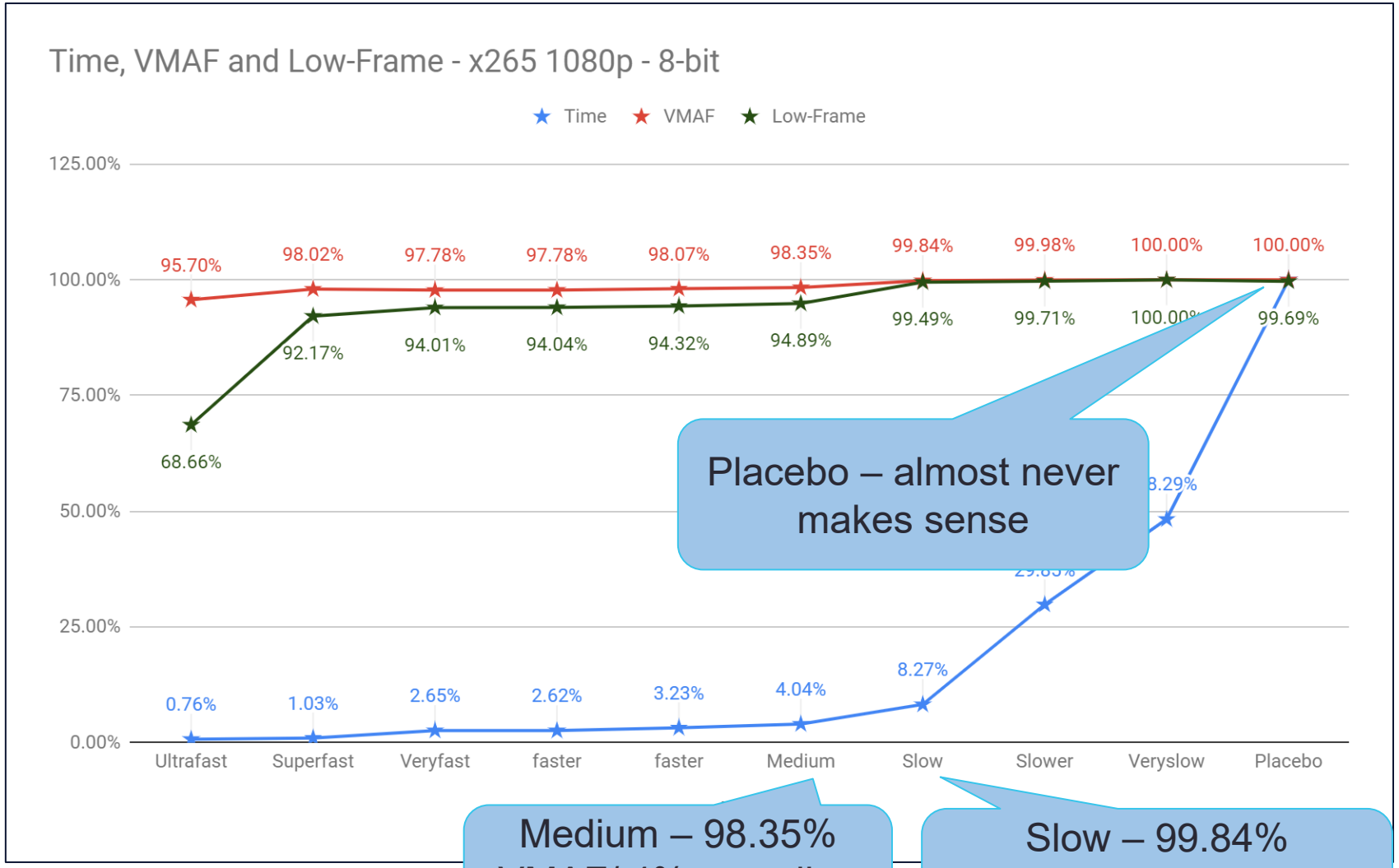
Exploring Presets

- Quiz: Does the preset control distribution quality?
  - Yes?
  - No? 

Preset Role

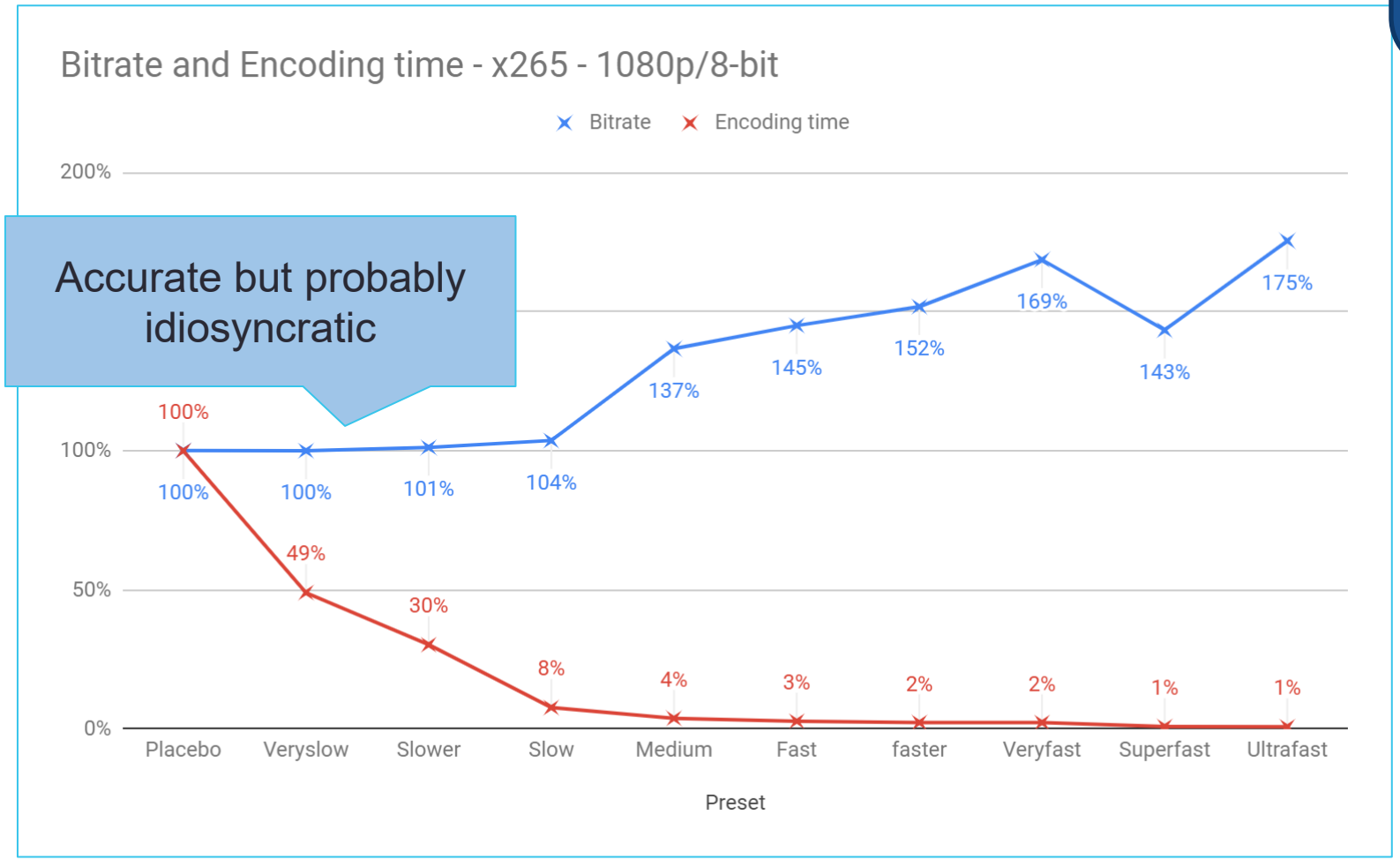
- Presets control ***encoding time***
- Most producers:
  - Choose quality level (VMAF 93-95/PSNR 45) and encode to match that quality level
- If preset doesn’t achieve target quality, you boost the bitrate
  - So, preset doesn’t control ***quality***, it controls encoding cost, but it necessarily impacts ***bandwidth***
  - Choosing a preset is ***always*** a tradeoff between encoding cost and bandwidth cost

- Two files encoded using all presets
- Measure encoding time
- Harmonic mean VMAF
- Low-frame VMAF
- Display as a % of maximum time/score
- What's the best preset?



HEVC - 8-bit 1080p Preset

How much do we have to increase the bitrate for different presets to match the quality of the highest quality preset (slower, veryslow, placebo)



Preset	**Bitrate**	**Encoding time**
slower	175%	1%
Superfast	143%	1%
Veryfast	169%	2%
faster	152%	2%
Fast	145%	3%
Medium	137%	4%
Slow	104%	8%
Slower	101%	30%
Veryslow	100%	49%
Placebo	100%	100%

x265 - 1080p - Viewer Count Breakeven - $0.08/GB

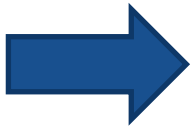
- Bitrate @ 100%
- Encoding cost @ 100%
- Bandwidth cost at 100% (here 8 cents/GB)
- Then interpolate to different presets
  - Ultrafast
  - Placebo
- Compute for different view counts
  - Color code to show highest and lowest costs

Input parameters

		Bitrate	2500						
		MBytes per hour	1.125		Cost per GB		0.08		
		Encode/hr	$5.5						
Preset	**Encode**	**Bandwidth**			**50**	**100**	**250**	**500**	**1000**	**5000**
Ultrafast	$0.53	2.19	$0.18		$9	$18	$44	$88	$176	$876
Superfast	$0.59	1.92	$0.15		$8	$16	$39	$77	$154	$767
Veryfast	$0.73	1.69	$0.13		$7	$14	$34	$68	$136	$675
faster	$0.99	1.41	$0.11		$7	$12	$29	$57	$114	$564
fast	$1.25	1.40	$0.11		$7	$12	$29	$57	$114	$563
Medium	$1.44	1.25	$0.10		$6	$11	$27	$52	$102	$503
Slow	$2.08	1.20	$0.10		$7	$12	$26	$50	$98	$483
Slower	$2.95	1.17	$0.09		$8	$12	$26	$50	$97	$471
Veryslow	$5.50	1.13	$0.09		$10	$15	$28	$51	$96	$456
Placebo	$21.89	1.13	$0.09		$26	$31	$44	$67	$112	$473

x265 - 1080p - Viewer Count Breakeven - $0.04/GB

- Because bandwidth savings are reduced, encoding costs remain relevant longer (faster, not medium)



		Bitrate		2500						
		MBytes per hour		1125		Cost per GB		0.04		
		Encode/hr		5.5						
Preset	**Encode**	**Bandwidth**			**50**	**100**	**250**	**500**	**1000**	**5000**
Ultrafast	$0.53	2.19	$0.09		$5	$9	$22	$44	$88	$438
Superfast	$0.59	1.92	$0.08		$4	$8	$20	$39	$77	$384
Veryfast	$0.73	1.69	$0.07		$4	$7	$18	$34	$68	$338
faster	$0.99	1.41	$0.06		$4	$7	$15	$29	$57	$282
fast	$1.25	1.40	$0.06		$4	$7	$15	$29	$57	$282
Medium	$1.44	1.25	$0.05		$4	$6	$14	$27	$52	$252
Slow	$2.08	1.20	$0.05		$4	$7	$14	$26	$50	$243
Slower	$2.95	1.17	$0.05		$5	$8	$15	$26	$50	$237
Veryslow	$5.50	1.13	$0.05		$8	$10	$17	$28	$51	$231
Placebo	$21.89	1.13	$0.05		$24	$26	$33	$44	$67	$247

x265 - Viewer Count Breakeven - $0.02/GB

- Because bandwidth savings are reduced, encoding costs remain relevant longer (faster, not medium)
- But even at $0.02, at 500 viewers, veryslow is the cheapest alternative

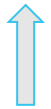
		Bitrate		2500						
		MBytes per hour		1125		Cost per GB		0.02		
		Encode/hr		5.5						
Preset	**Encode**	**Bandwidth**			**50**	**100**	**250**	**500**	**1000**	**5000**
Ultrafast	$0.53	2.19	$0.04		$3	$5	$11	$22	$44	$219
Superfast	$0.59	1.92	$0.04		$3	$4	$10	$20	$39	$192
Veryfast	$0.73	1.69	$0.03		$2	$4	$9	$18	$34	$169
faster	$0.99	1.41	$0.03		$2	$4	$8	$15	$29	$142
fast	$1.25	1.40	$0.03		$3	$4	$8	$15	$29	$142
Medium	$1.44	1.25	$0.03		$3	$4	$8	$14	$27	$127
Slow	$2.08	1.20	$0.02		$3	$4	$8	$14	$26	$122
Slower	$2.95	1.17	$0.02		$4	$5	$9	$15	$26	$120
Veryslow	$5.50	1.13	$0.02		$7	$8	$11	$17	$28	$118
Placebo	$21.89	1.13	$0.02		$23	$24	$28	$33	$44	$135

Preset Summary

- Run your own tests on your own files (results will vary by content, resolution, etc.)
- Perform your own calculations
- If your typical video is viewed over 10,000 times (or so), it almost always pay to use the slower preset (or your highest quality preset)
  - Placebo almost never delivers the best quality and almost always takes much, much longer to encode

Specifying the Preset

```
-preset slower
```



Set preset

`-Preset slower` – Choosing the faster preset

If you don’t choose a preset, FFmpeg uses medium, which is the default.

Reference Frames

Should be: 4:55 – skippable

Writing library : x265 3.5+96-9c9ab68fc:[Windows][GCC 12.2.0][64 bit] 8bit+10bit+12bit

cpuid=1111039 / frame-threads=5 / numa-pools=32,32 / wpp / no-pmode / no-pme / no-psnr / no-ssim / log-level=2 / input-csp=1 / input-res=1920x1080 / interlace=0 / total-frames=0 / level-idc=0 / high-tier=1 / uhd-bd=0 / ref=5 / no-allow-non-conformance / no-repeat-headers / annexb / no-aud / no-eob / no-eos / no-hrd / info / hash=0 / temporal-layers=0 / open-gop / min-keyint=60 / keyint=60 / gop-lookahead=0 / bframes=8 / b-adapt=2 / b-pyramid / bframe-bias=0 / rc-lookahead=40 / lookahead-slices=0 / scenecut=0 / no-hist-scenecut / radl=0 / no-splice / no-intra-

- Veryslow preset use 5 reference frames
  - How much encoding time does this take?
  - How much quality do they add?
  - Is it worth it?

Encode 2 files at differing values

	Baseline	Ref=1	Ref=2	Ref=3	Ref=4	Delta
Freedom	0:04:56	0:03:36	0:04:04	0:04:24	0:04:34	27.03%
Football	0:08:34	0:06:22	0:07:04	0:07:42	0:07:58	25.68%
Average	0:06:45	0:04:59	0:05:34	0:06:03	0:06:16	26.17%
Bitrate	**Baseline**	**Ref=1**	**Ref=2**	**Ref=3**	**Ref=4**	**Delta**
Freedom	2,159	2,156	2155	2,161	2,156	0.28%
Football	2,349	2,354	2354	2,354	2,350	0.21%
Average	**2,254**	**2,255**	**2,255**	**2,258**	**2,253**	**0.20%**
VMAF	**Baseline**	**Ref=1**	**Ref=2**	**Ref=3**	**Ref=4**	**Delta**
Freedom	93.45	93.19	93.33	93.40	93.42	0.27%
Football	93.74	93.53	93.59	93.66	93.70	0.22%
Average	**93.59**	**93.36**	**93.46**	**93.53**	**93.56**	**0.25%**
Low Frame	**Baseline**	**Ref=1**	**Ref=2**	**Ref=3**	**Ref=4**	**Delta**
Freedom	88.47	88.03	88.20	88.23	88.50	0.54%
Football	83.54	83.28	83.39	83.28	83.47	0.31%
Average	**86.00**	**85.65**	**85.79**	**85.75**	**85.99**	**0.41%**
Standard Deviation	**Baseline**	**Ref=1**	**Ref=2**	**Ref=3**	**Ref=4**	**Delta**
Freedom	2.51	2.57	2.55	2.52	2.52	2.54%
Football	4.18	4.21	4.20	4.19	4.18	0.76%
Average	**3.34**	**3.39**	**3.38**	**3.35**	**3.35**	**1.44%**

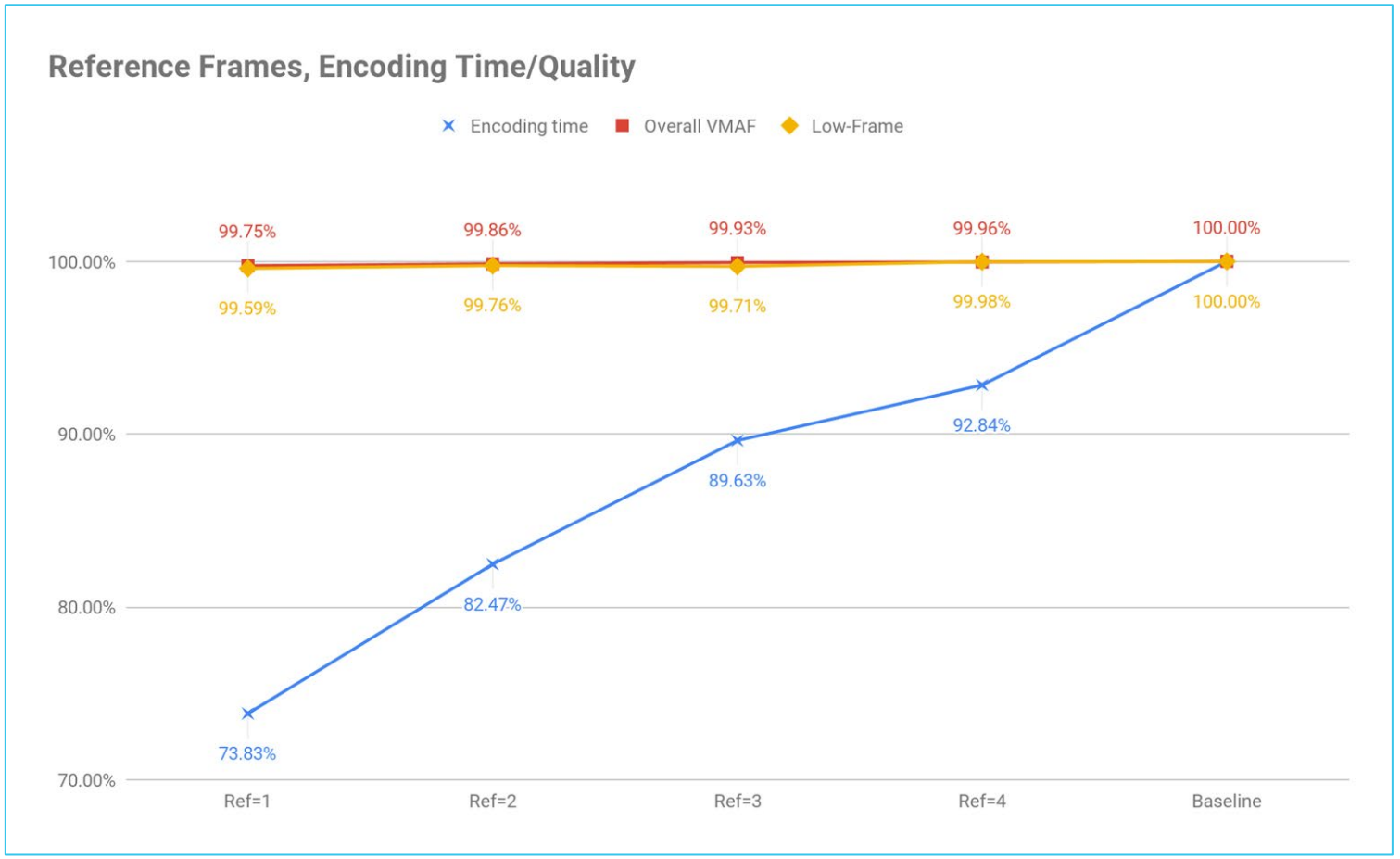
26% encoding time

Overall VMAF

Low Frame VMAF

VMAF Std. Dev.

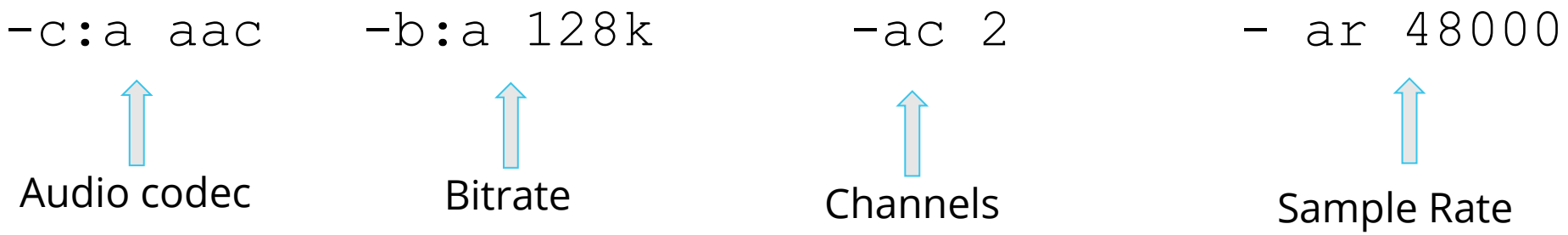
Reference Frames - x265



- Reduce encoding time by 26%; minimal quality delta.
- May work differently with different source clips
  - I tested football and a concert video
- Test - may be able to shave 25%+ from encoding time with minimal quality impact

Audio

Should be: 5:00



- Default:
  - AAC for MP4
  - Channels: source
  - Sample rate: source
  - Data rate: inconsistent

- HE, HE2 are different codecs
- Channels
  - 1 = mono
  - 2 - stereo

Working with x265 – Creating a Ladder - Football

```
FFmpeg -y -i Football_in.mp4 -c:v libx265 -an -preset veryslow -x265-params
keyint=60:min-keyint=60:scenecut=0:bitrate=3000:open-gop=0:pass=1 -f mp4 NUL && \
ffmpeg -y -i Football_in.mp4 -c:v libx265 -an -preset veryslow -x265-params
keyint=60:min-keyint=60:scenecut=0:bitrate=3000:vbv-maxrate=6000:vbv-bufsize=6000:open-
gop=0:pass=2 football_1080p.mp4

ffmpeg -y -i Football_in.mp4 -c:v libx265 -an -s 1280x720 -preset veryslow -x265-params
keyint=60:min-keyint=60:scenecut=0:bitrate=1500:open-gop=0:pass=1 -f mp4 NUL && \
ffmpeg -y -i Football_in.mp4 -c:v libx265 -an -s 1280x720 -preset veryslow -x265-params
keyint=60:min-keyint=60:scenecut=0:bitrate=1500:vbv-maxrate=3000:vbv-bufsize=3000:open-
gop=0:pass=2 football_720p.mp4

ffmpeg -y -i Football_in.mp4 -c:v libx265 -an -s 960x540 -c:v libx265 -preset veryslow
-x265-params keyint=60:min-keyint=60:scenecut=0:bitrate=800:open-gop=0:pass=1 -f mp4
NUL && \
ffmpeg -y -i Football_in.mp4 -c:v libx265 -an -s 960x540 -preset veryslow -x265-params
keyint=60:min-keyint=60:scenecut=0:bitrate=800:vbv-maxrate=1600:vbv-bufsize=1600:open-
gop=0:pass=2 football_540p.mp4

ffmpeg -y -i Football_in.mp4 -c:a aac -b:a 128k -vn football_audio.mp4
```

Working with x265 – Creating a Ladder - Jan

```
ffmpeg -y -i jan.mp4 -c:v libx265 -preset veryslow -c:a aac -b:a 128k -x265-params
keyint=60:min-keyint=60:scenecut=0:bitrate=2000:open-gop=0:pass=1 -f mp4 NUL && \
ffmpeg -y -i jan.mp4 -c:v libx265 -an -preset veryslow -x265-params keyint=60:min-
keyint=60:scenecut=0:bitrate=2000:vbv-maxrate=4000:vbv-bufsize=4000:open-gop=0:pass=2
jan_1080p_2M.mp4

ffmpeg -y -i jan.mp4 -s 1280x720 -c:v libx265 -preset veryslow -c:a aac -b:a 128k -x265-params
keyint=60:min-keyint=60:scenecut=0:bitrate=1200:open-gop=0:pass=1 -f mp4 NUL && \
ffmpeg -y -i jan.mp4 -c:v libx265 -s 1280x720 -an -preset veryslow -x265-params keyint=60:min-
keyint=60:scenecut=0:bitrate=1200:vbv-maxrate=2400:vbv-bufsize=2400:open-gop=0:pass=2
jan_720p_1_2M.mp4

ffmpeg -y -i jan.mp4 -s 960x540 -c:v libx265 -preset veryslow -c:a aac -b:a 128k -x265-params
keyint=60:min-keyint=60:scenecut=0:bitrate=600:open-gop=0:pass=1 -f mp4 NUL && \
ffmpeg -y -i jan.mp4 -c:v libx265 -s 960x540 -an -preset veryslow -x265-params keyint=60:min-
keyint=60:scenecut=0:bitrate=600:vbv-maxrate=1200:vbv-bufsize=1200:open-gop=0:pass=2
jan_540p_600k.mp4

ffmpeg -y -i jan.mp4 -c:a aac -b:a 128k –vn jan_audio.mp4
```

Multi-pass Encoding – Rules of the Road

- Must run separate first pass for each resolution
  - Different for H.264 – one pass can serve multiple rungs
- Which config options in first pass?
  - Frame settings (B-frame/Key frame)
  - Target data rate
  - Some say audio settings
    - My tests haven’t shown this is true

Packaging

- You can output HLS from FFmpeg but
  - It's complex
  - Very hard for 2-pass
- Packagers more easily prepare packages for:
  - HLS
  - DASH
  - CMAF
- We consider two open-source tools
  - Bento4
  - GPAC

- Files to be packaged
  - Jan_1080p.mp4
  - Jan_720p.mp4
  - Jan_540p.mp4
  - Jan_audio.mp4
  - Jan.vtt (captions)
- What's the point
  - A glimpse of functionality and simplicity
  - Key differences in viability

Intro to Bento4

Should be: 5:03

- Set of encoding/packaging utilities
- We will use three of these

What can I do with Bento4?

https://www.bento4.com/

- MPEG DASH with fragmented MP4 files, as defined in the international specification ISO/IEC 23009-1
- HLS with TS or MP4 segments (dual DASH/HLS output), as defined in RFC 8216
- CMAF (Common Media Application Format) as defined in ISO/IEC 23000-19
- MPEG Common Encryption (CENC) as specified in the international specification ISO/IEC 23001-7
- Parsing and multiplexing of H.264 (AVC) video and AAC audio elementary streams
- Support for multiple DRM systems that are compatible with MP4-formatted content (usually leveraging CENC Common Encryption), such as Marlin, PlayReady, Widevine, FairPlay and Adobe Access.
- Support for a wide range of codecs, including H.264 (AVC), H.265 (HEVC), AAC, HE-AAC, xHE-AAC, AC3 and eAC3 (Dolby Digital), AC4, Dolby Atmos, DTS, ALAC, and many more.
- Support for Dolby Vision and HDR.

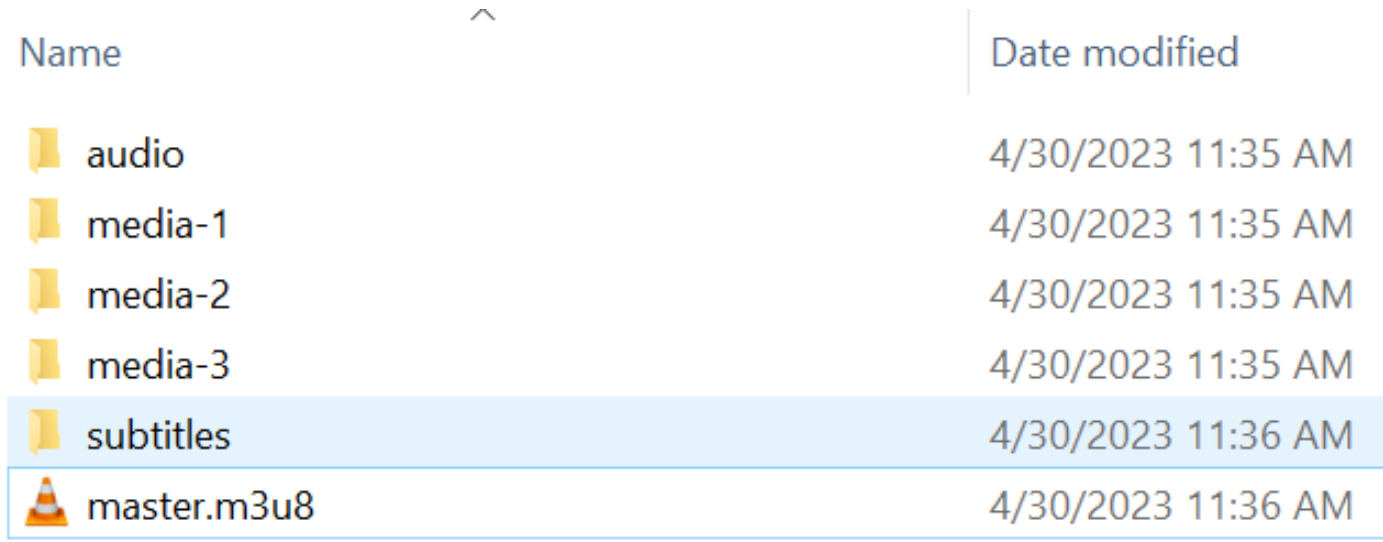
Package to .TS HLS

```
mp4hls --hls-version 7 -f Jan_1080p.mp4 Jan_720p.mp4
Jan_540p.mp4 Jan_audio.mp4 [+format=webvtt,+language=en]jan.vtt
```

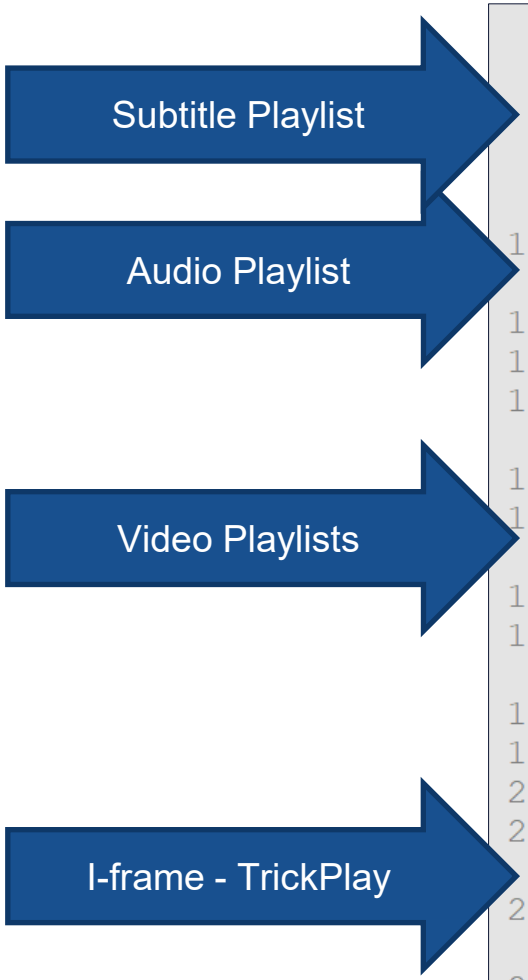
- `mp4hls` – call program
- `hls-version` – compatibility
- `-f` – force overwrite output folder (a production thing)
- List all media files (including captions)

Package to .TS HLS

- Outputs:
  - Master.m3u8
  - Stream.m3u8 for each bitrate
  - Iframe.m3u8 for each bitrate
  - Subtitle m3u8
  - ts fragments for each bitrate



Master Manifest




```
# Subtitles
#EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subtitles",NAME="English",LANGUAGE="en",URI="subtitles/en/subtitles.m3u8"

# Audio
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="audio_aac",NAME="eng",LANGUAGE="eng",AUTOSELECT=YES,DEFAULT=YES,URI="audio/aac/eng/stream.m3u8"

# Media Playlists
#EXT-X-STREAM-INF:AVERAGE-BANDWIDTH=2207363,BANDWIDTH=9107902,CODECS="hev1.1.6.L120.90,mp4a.40.2",RESOLUTION=1920x1080,AUDIO="audio_aac",SUBTITLES="subtitles"
media-1/stream.m3u8
#EXT-X-STREAM-INF:AVERAGE-BANDWIDTH=1394864,BANDWIDTH=4961008,CODECS="hev1.1.6.L93.90,mp4a.40.2",RESOLUTION=1280x720,AUDIO="audio_aac",SUBTITLES="subtitles"
media-2/stream.m3u8
#EXT-X-STREAM-INF:AVERAGE-BANDWIDTH=781875,BANDWIDTH=1805764,CODECS="hev1.1.6.L90.90,mp4a.40.2",RESOLUTION=960x540,AUDIO="audio_aac",SUBTITLES="subtitles"
media-3/stream.m3u8

# I-Frame Playlists
#EXT-X-I-FRAME-STREAM-INF:AVERAGE-BANDWIDTH=205813,BANDWIDTH=8879760,CODECS="hev1.1.6.L120.90",RESOLUTION=1920x1080,URI="media-1/iframes.m3u8"
#EXT-X-I-FRAME-STREAM-INF:AVERAGE-BANDWIDTH=144129,BANDWIDTH=4732867,CODECS="hev1.1.6.L93.90",RESOLUTION=1280x720,URI="media-2/iframes.m3u8"
#EXT-X-I-FRAME-STREAM-INF:AVERAGE-BANDWIDTH=83596,BANDWIDTH=1577622,CODECS="hev1.1.6.L90.90",RESOLUTION=960x540,URI="media-3/iframes.m3u8"
```

Media Manifest/Segments

Segments

Name	Date	Type	Size	Length
segment-1.ts	4/27/2023 4:24 PM	TS File	819 KB	00:00:05
segment-2.ts	4/27/2023 4:24 PM	TS File	738 KB	00:00:05
segment-4.ts	4/27/2023 4:24 PM	TS File	612 KB	00:00:05
segment-3.ts	4/27/2023 4:24 PM	TS File	410 KB	00:00:05
segment-0.ts	4/27/2023 4:24 PM	TS File	400 KB	00:00:05
segment-5.ts	4/27/2023 4:24 PM	TS File	44 KB	
iframes.m3u8	4/27/2023 4:24 PM	M3U8 Other File (VLC)	2 KB	
stream.m3u8	4/27/2023 4:24 PM	M3U8 Other File (VLC)	1 KB	

Regular

master.m3u8 | stream.m3u8

```
#EXTM3U
#EXT-X-VERSION:7
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-INDEPENDENT-SEGMENTS
#EXT-X-TARGETDURATION:6
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:6.006000,
segment-0.ts
#EXTINF:6.006000,
segment-1.ts
#EXTINF:6.006000,
segment-2.ts
#EXTINF:6.006000,
segment-3.ts
#EXTINF:6.006000,
segment-4.ts
#EXTINF:0.033367,
```

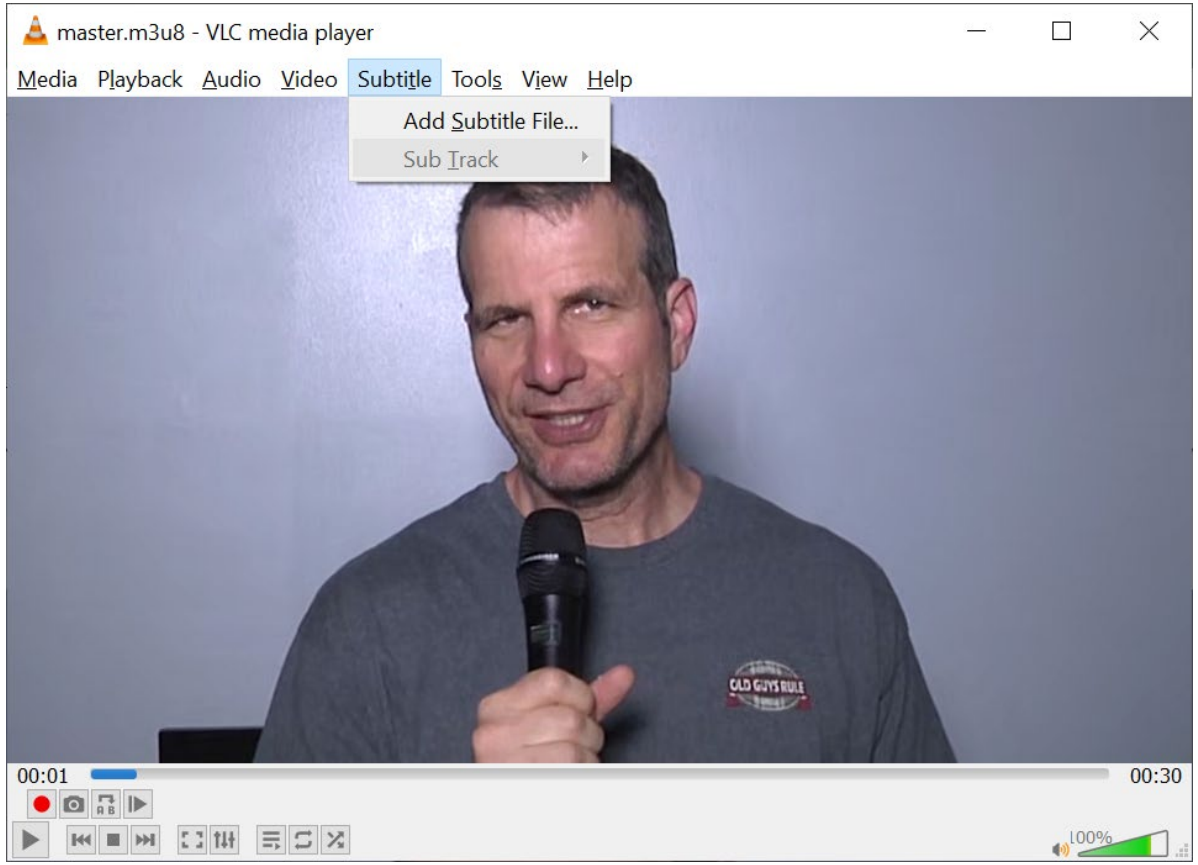
iframes.m3u8

```
#EXTM3U
#EXT-X-VERSION:7
#EXT-X-PLAYLIST-TYPE:VOD
#EXT-X-I-FRAMES-ONLY
#EXT-X-INDEPENDENT-SEGMENTS
#EXT-X-TARGETDURATION:2
#EXT-X-MEDIA-SEQUENCE:0
#EXTINF:2.002000,
#EXT-X-BYTERANGE:16544@376
segment-0.ts
#EXTINF:2.002000,
#EXT-X-BYTERANGE:30080@105468
segment-0.ts
#EXTINF:2.002000,
#EXT-X-BYTERANGE:42488@242332
segment-0.ts
#EXTINF:2.002000,
```

Trick play

Results

- Audio and video play as expected
- No subtitle in subtitle track



Dual HLS and DASH from fMP4

- Convert files to fragmented MP4
- Package

MP4Fragment – Convert to Fragmented MP4

```
mp4fragment Jan_1080p.mp4 Jan_1080p_F.mp4
mp4fragment Jan_720p.mp4 Jan_720p_F.mp4
mp4fragment Jan_540p.mp4 Jan_540p_F.mp4
mp4fragment Jan_audio.mp4 Jan_audio_F.mp4
```

```
C:\Mile_High\Jan_stuff\Bento_4\DASH_CMAF>mp4fragment Jan_1080p.mp4 Jan_1080p_F.mp4
found regular I-frame interval: 60 frames (at 29.970 frames per second)

C:\Mile_High\Jan_stuff\Bento_4\DASH_CMAF>mp4fragment Jan_720p.mp4 Jan_720p_F.mp4
found regular I-frame interval: 60 frames (at 29.970 frames per second)

C:\Mile_High\Jan_stuff\Bento_4\DASH_CMAF>mp4fragment Jan_540p.mp4 Jan_540p_F.mp4
found regular I-frame interval: 60 frames (at 29.970 frames per second)

C:\Mile_High\Jan_stuff\Bento_4\DASH_CMAF>mp4fragment Jan_audio.mp4 Jan_audio_F.mp4
unable to autodetect fragment duration, using default
```

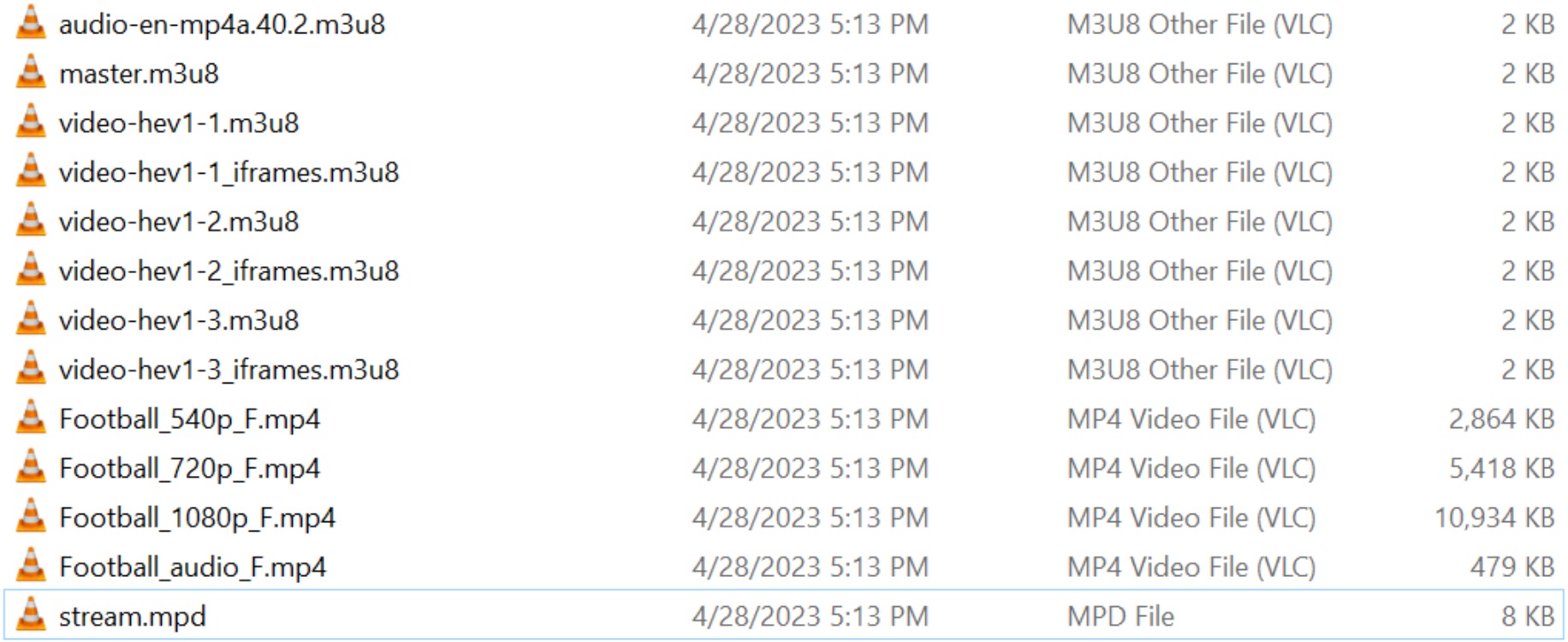
Dual HLS and DASH from fMP4

```
mp4dash --hls --no-split --use-segment-list Jan_1080p_F.mp4
Jan_720p_F.mp4 Jan_540p_F.mp4 Jan_audio_F.mp4
[+format=webvtt,+language=en]jan.vtt
```

HLS manifests

Media files (no Segments)

Dash Manifest



audio-en-mp4a.40.2.m3u8	4/28/2023 5:13 PM	M3U8 Other File (VLC)	2 KB
master.m3u8	4/28/2023 5:13 PM	M3U8 Other File (VLC)	2 KB
video-hev1-1.m3u8	4/28/2023 5:13 PM	M3U8 Other File (VLC)	2 KB
video-hev1-1_iframes.m3u8	4/28/2023 5:13 PM	M3U8 Other File (VLC)	2 KB
video-hev1-2.m3u8	4/28/2023 5:13 PM	M3U8 Other File (VLC)	2 KB
video-hev1-2_iframes.m3u8	4/28/2023 5:13 PM	M3U8 Other File (VLC)	2 KB
video-hev1-3.m3u8	4/28/2023 5:13 PM	M3U8 Other File (VLC)	2 KB
video-hev1-3_iframes.m3u8	4/28/2023 5:13 PM	M3U8 Other File (VLC)	2 KB
Football_540p_F.mp4	4/28/2023 5:13 PM	MP4 Video File (VLC)	2,864 KB
Football_720p_F.mp4	4/28/2023 5:13 PM	MP4 Video File (VLC)	5,418 KB
Football_1080p_F.mp4	4/28/2023 5:13 PM	MP4 Video File (VLC)	10,934 KB
Football_audio_F.mp4	4/28/2023 5:13 PM	MP4 Video File (VLC)	479 KB
stream.mpd	4/28/2023 5:13 PM	MPD File	8 KB

Error Message

```
C:\Mile_High\Jan_stuff\Bento_4\DASH_CMAF>mp4dash -f --hls --no-split --use-segment-list Jan_108
0p_F.mp4 Jan_720p_F.mp4 Jan_540p_F.mp4 Jan_audio_F.mp4 [+format=webvtt]jan.vtt
Parsing media file 1: Jan_1080p_F.mp4
Parsing media file 2: Jan_720p_F.mp4
Parsing media file 3: Jan_540p_F.mp4
Parsing media file 4: Jan_audio_F.mp4
Processing and Copying media file Jan_1080p_F.mp4
Processing and Copying media file Jan_720p_F.mp4
Processing and Copying media file Jan_540p_F.mp4
Processing and Copying media file Jan_audio_F.mp4
Processing and Copying subtitles file jan.vtt
ERROR: local variable 'language_name' referenced before assignment
```

Error Message

What's the point?

Error reported, reported as fixed, not fixed, re-reported, no action:

Support isn't that great

Hello, I'm trying to add subtitles in the conversion to fmp4 as follows;

```
mp4-dash.py --hls MP4/video.mp4 [+format=webvtt,+language=por]MP4/por.vtt
[+format=webvtt,+language=forced]MP4/forced.vtt
```

But I'm getting this error message and no subtitles are added, any solution?

```
Parsing media file 1: MP4/video.mp4
Splitting media file (audio) MP4/video.mp4
Splitting media file (audio) MP4/video.mp4
Splitting media file (video) MP4/video.mp4
Processing and Copying subtitles file MP4/por.vtt
Processing and Copying subtitles file MP4/forced.vtt
ERROR: local variable 'language_name' referenced before assignment
```

barbibulle self-assigned this on Jan 30, 2022

barbibulle added the bug label on Jan 30, 2022

barbibulle commented on Jan 30, 2022

I'm able to reproduce the issue. We'll fix this ASAP.

barbibulle mentioned this issue on Feb 27, 2022

Subtitles (WebVTT) not added to HLS playlist when running mp4dash with --hls #659 Closed

clia6547 commented last month • edited

I'm running version 1.6.0-639 on M2 chip and got the same issue, with or without **+language_name** tag

clia6547 mentioned this issue last month

Subtitles (WebVTT) not added to HLS playlist when running mp4dash with --hls #848 Open

https://github.com/axiomatic-systems/Bento4/issues/667

Missing Subtitles

Audio Playlist

Video Playlists

I-frame - TrickPlay

Subtitle Playlist (not working)

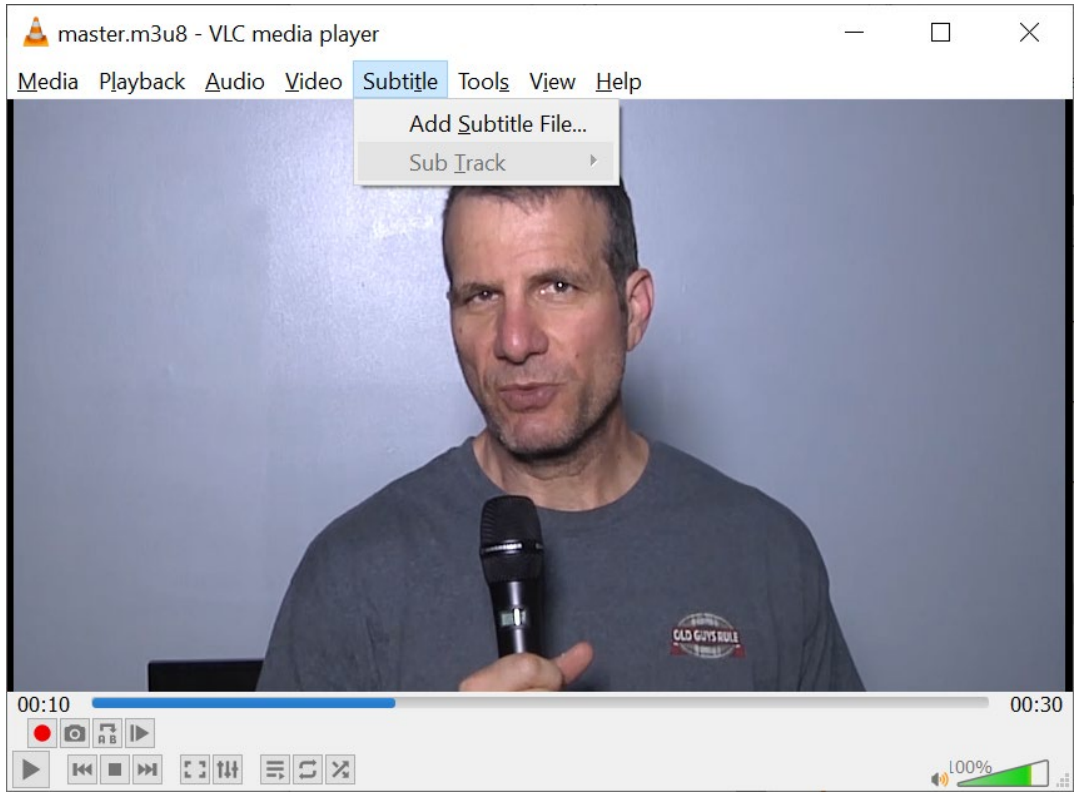
hls_long.txt | master4.m3u8 | master3.m3u8 | DASH.bat | master.m3u8

```
#EXTM3U
# Created with Bento4 mp4-dash.py, VERSION=2.0.0-639
#
#EXT-X-VERSION:6

# Media Playlists

# Audio
#EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="audio",LANGUAGE="en",NAME="English",AUTOSELECT=YES,DEFAULT=YES,CHANNELS="2",URI="audio-en-mp4a.40.2.m3u8"

# Video
#EXT-X-STREAM-INF:SUBTITLES="subtitles",AUDIO="audio",AVERAGE-BANDWIDTH=2135718,BANDWIDTH=3844137,CODECS="hev1.1.6.L120.90,mp4a.40.2",RESOLUTION=1920x1080,FRAME-RATE=29.970
video-hev1-1.m3u8
#EXT-X-STREAM-INF:SUBTITLES="subtitles",AUDIO="audio",AVERAGE-BANDWIDTH=1340440,BANDWIDTH=2422450,CODECS="hev1.1.6.L93.90,mp4a.40.2",RESOLUTION=1280x720,FRAME-RATE=29.970
video-hev1-2.m3u8
#EXT-X-STREAM-INF:SUBTITLES="subtitles",AUDIO="audio",AVERAGE-BANDWIDTH=741613,BANDWIDTH=1439853,CODECS="hev1.1.6.L90.90,mp4a.40.2",RESOLUTION=960x540,FRAME-RATE=29.970
video-hev1-3.m3u8

# I-Frame Playlists
#EXT-X-I-FRAME-STREAM-INF:AVERAGE-BANDWIDTH=200587,BANDWIDTH=8647433,CODECS="hev1.1.6.L120.90",RESOLUTION=1920x1080,URI="video-hev1-1_iframes.m3u8"
#EXT-X-I-FRAME-STREAM-INF:AVERAGE-BANDWIDTH=140102,BANDWIDTH=4580619,CODECS="hev1.1.6.L93.90",RESOLUTION=1280x720,URI="video-hev1-2_iframes.m3u8"
#EXT-X-I-FRAME-STREAM-INF:AVERAGE-BANDWIDTH=80919,BANDWIDTH=1495385,CODECS="hev1.1.6.L90.90",RESOLUTION=960x540,URI="video-hev1-3_iframes.m3u8"

# Subtitles (WebVTT)
```

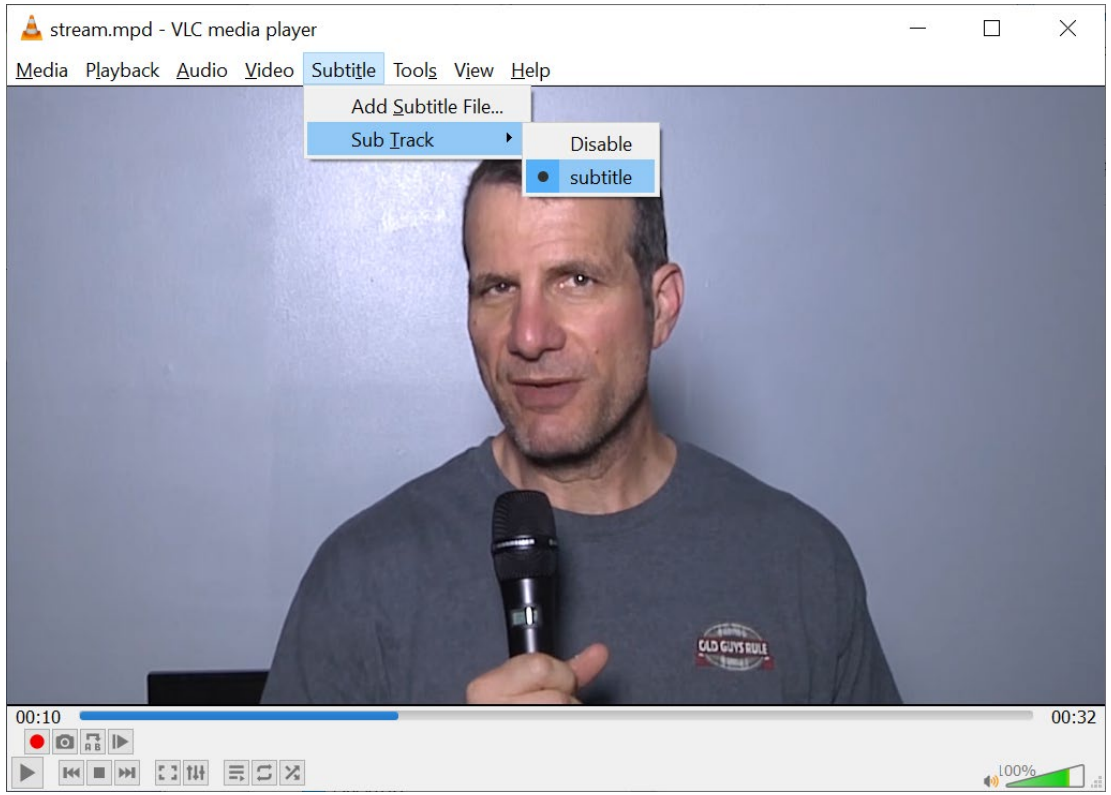
DASH Manifest

Single file with byte range requests to separate media files (3 vid/1 audio/no captions)

Playback



- HLS - again,
  - Synched AV
  - No subtitle track



- DASH
  - Synched AV
  - Subtitle track found, but no subtitles

Bento4 Summary

- Very simple to use open-source tool
- Support lacking
- Challenging for real-world production environment

Intro to GPAC

Should be: 5:12

- GPAC is an open-source multimedia framework used in many media production chains.
- GPAC provides three sets of tools based on a core library called libgpac:
  - A multimedia packager, called MP4Box (we will use this one)
  - A generic media pipeline orchestrator, called gpac, used to build complex media processing sessions (players, transcoders, streamers, packagers, servers, …),
  - Bindings for Python and NodeJS

What can I do with GPAC?

https://gpac.wp.imt.fr/home/gpac-features/

- GPAC features encoders and multiplexers, publishing and content distribution tools for various codec and container formats. Following is a good summary of what GPAC can do for you:
  - MP4/3GP Conversion from MP3, AVI, MEPG-2 TS, MKV, MPEG-PS, etc.
  - Media decoding and encoding in various formats (H264, HEVC, AAC, AC3, etc.)
  - **Preparing MP4, 3GP and MPEG-2 TS files for DASH and HLS streaming**
  - File hinting for RTP/RTSP and QTSS/DSS servers (MPEG-4 / ISMA / 3GP / 3GP2 files)
  - CENC encryption and decryption
  - File layout: fragmentation or interleaving, and cleaning
  - File splitting by size or time, extraction from file and file concatenation
  - XML information dumping for MP4 and RTP hint tracks
  - QT/3GPP timed text tools (SUB/SRT/TTXT/TeXML), VobSub import/export

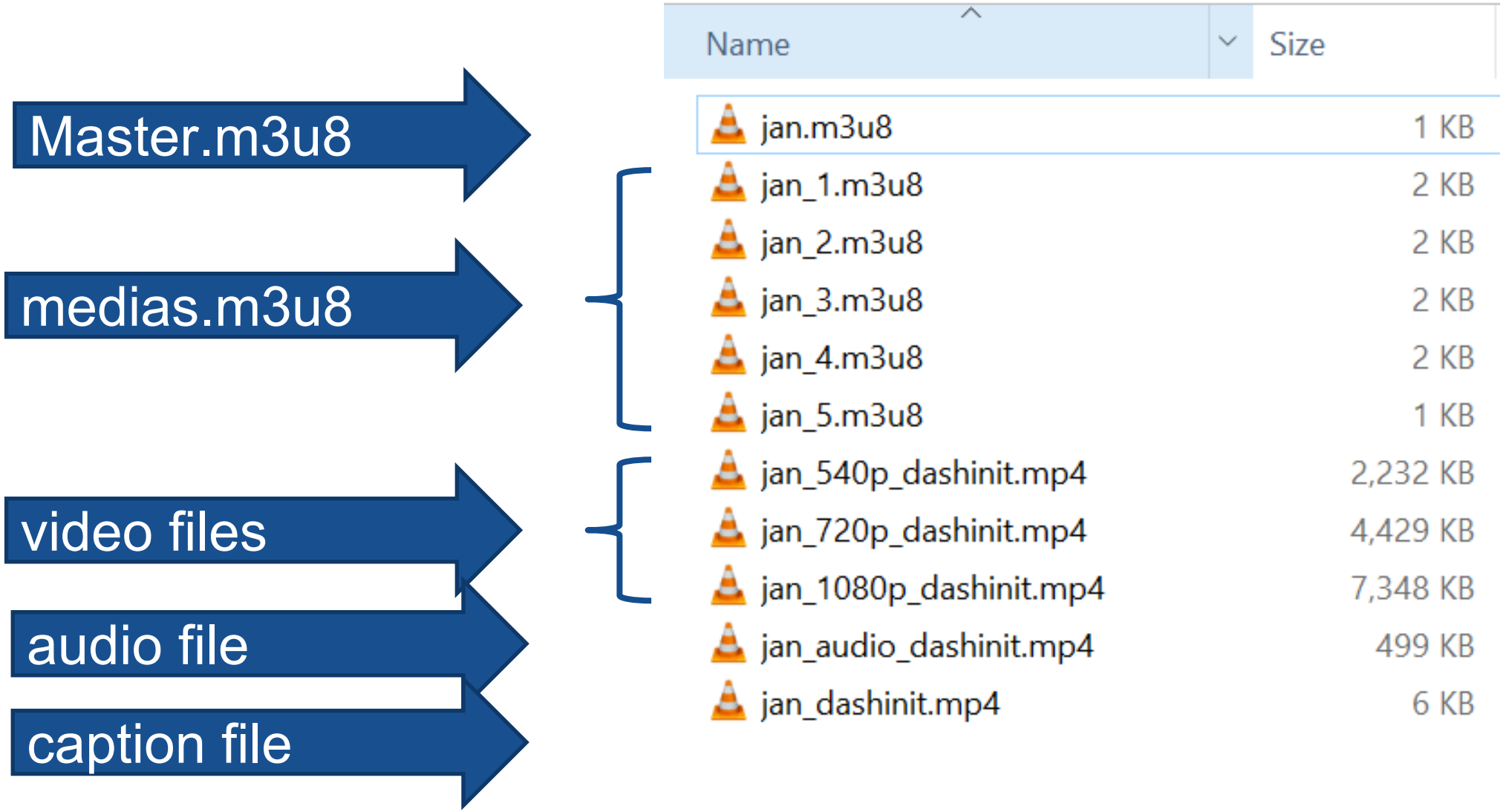
HLS with one file per segment

```
MP4Box -dash 2000 jan_1080p.mp4 jan_720p.mp4 jan_540p.mp4
jan_audio.mp4 jan.vtt -out jan.m3u8
```

- `mp4box` – call program
- `dash 2000` – size of byte range requests
- List all media files (including captions)
- `-out jan.m3u8` – format for HLS
- By default
  - Fragmented MP4 – can do single/multiple .ts

- Outputs:
  - Master.m3u8
  - Stream.m3u8 for each bitrate
  - FMP4 media files (single)
  - No trick play - checking

Package to Fragmented MP4



Master Manifest

Video Playlists

Audio Playlist

No - TrickPlay

jan.m3u8

```
1  #EXTM3U
2  #EXT-X-VERSION:6
3  #EXT-X-INDEPENDENT-SEGMENTS
4
5  #EXT-X-STREAM-INF:BANDWIDTH=1998037,CODECS="hev1.1.6.L120.90,mp4a.40.2",RESOLUTION=1920
   x1080,FRAME-RATE=30,AUDIO="audio2",SUBTITLES="subs3"
6  jan_1.m3u8
7
8  #EXT-X-STREAM-INF:BANDWIDTH=1202760,CODECS="hev1.1.6.L93.90,mp4a.40.2",RESOLUTION=1280x
   720,FRAME-RATE=30,AUDIO="audio2",SUBTITLES="subs3"
9  jan_2.m3u8
10
11 #EXT-X-STREAM-INF:BANDWIDTH=603932,CODECS="hev1.1.6.L90.90,mp4a.40.2",RESOLUTION=960x54
   0,FRAME-RATE=30,AUDIO="audio2",SUBTITLES="subs3"
12 jan_3.m3u8
13
14 #EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="audio2",NAME="4",LANGUAGE="eng",AUTOSELECT=YES,URI="j
   an_4.m3u8",CHANNELS="2"
15 #EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subs3",NAME="5",AUTOSELECT=YES,URI="jan_5.m3u8"
16
```

Subtitles with video

Results

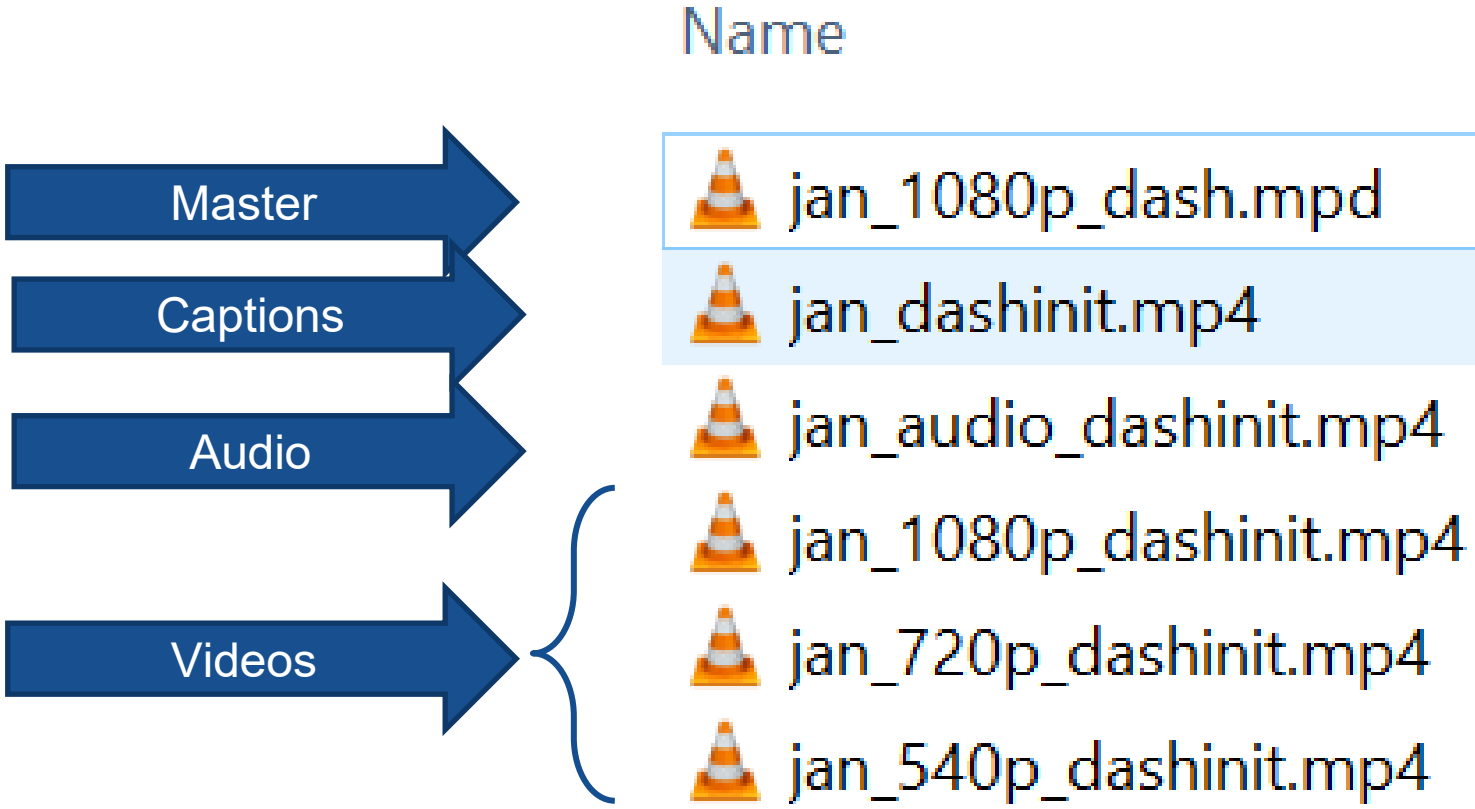
- Audio and video play as expected
- Subtitles available and appear

DASH with One File per Ladder (Fragmented MP4)

```
MP4Box -dash 2000 jan_1080p.mp4 jan_720p.mp4 jan_540p.mp4
jan_audio.mp4 jan.vtt
```

- `mp4box` – call program
- `dash 2000 –` size of byte range requests
- List all media files (including captions)

Package to Fragmented MP4

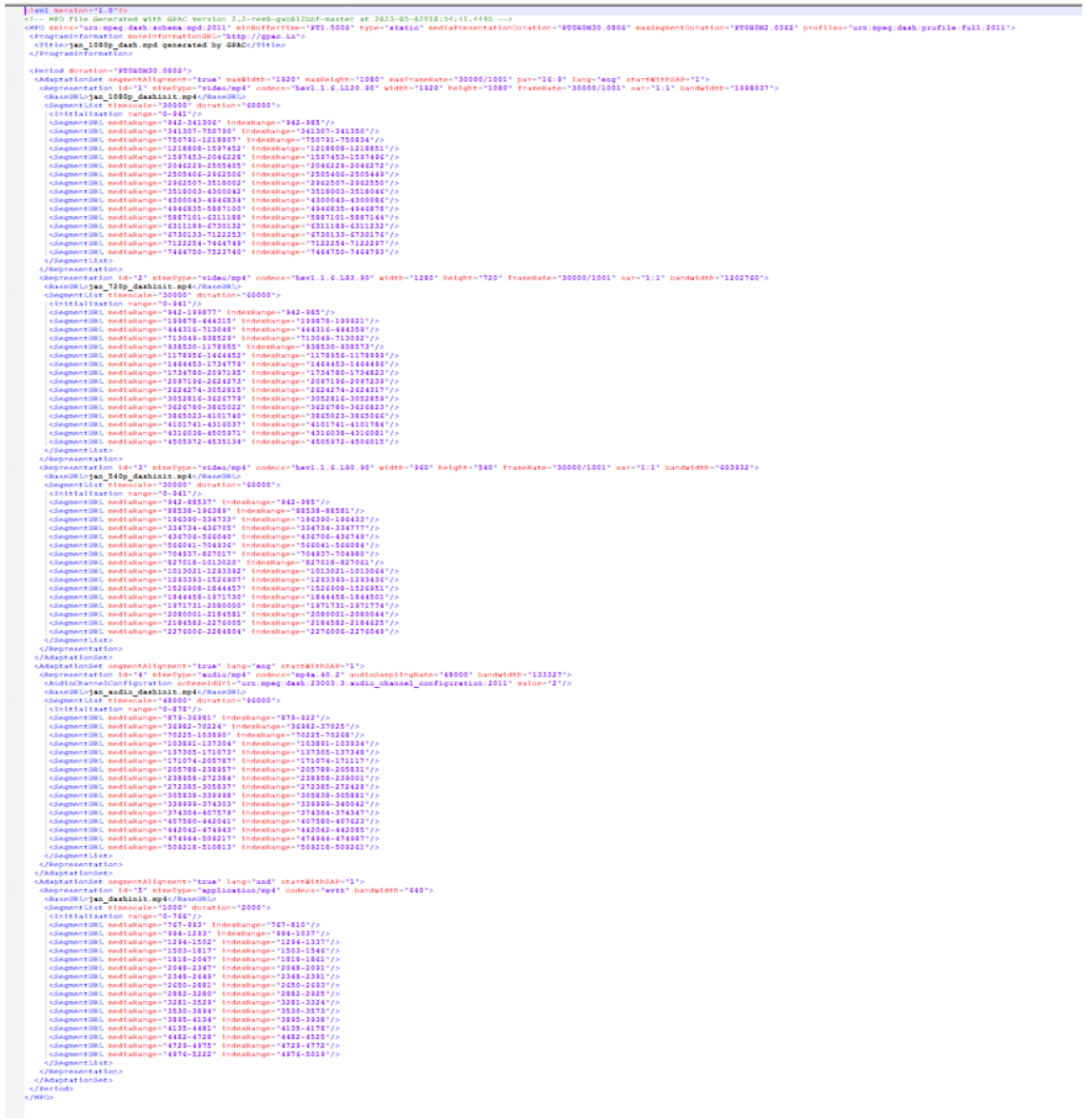


Master Manifest

Video(s) byte range

Audio byte range

Subtitle byte range



Results

- Audio and video play as expected
- Subtitles available and appear

CMAF with One File per Segment

```
MP4Box -dash 2000 jan_1080p.mp4 jan_720p.mp4 jan_540p.mp4
jan_audio.mp4 jan.vtt --cmaf=cmf2 --dual
```

- `mp4box` – call program
- `dash 2000` – size of byte range requests
- List all media files (including captions)
- `cmaf=cmf2` – Output HLS/DASH manifests
- `--dual` Output HLS manifests

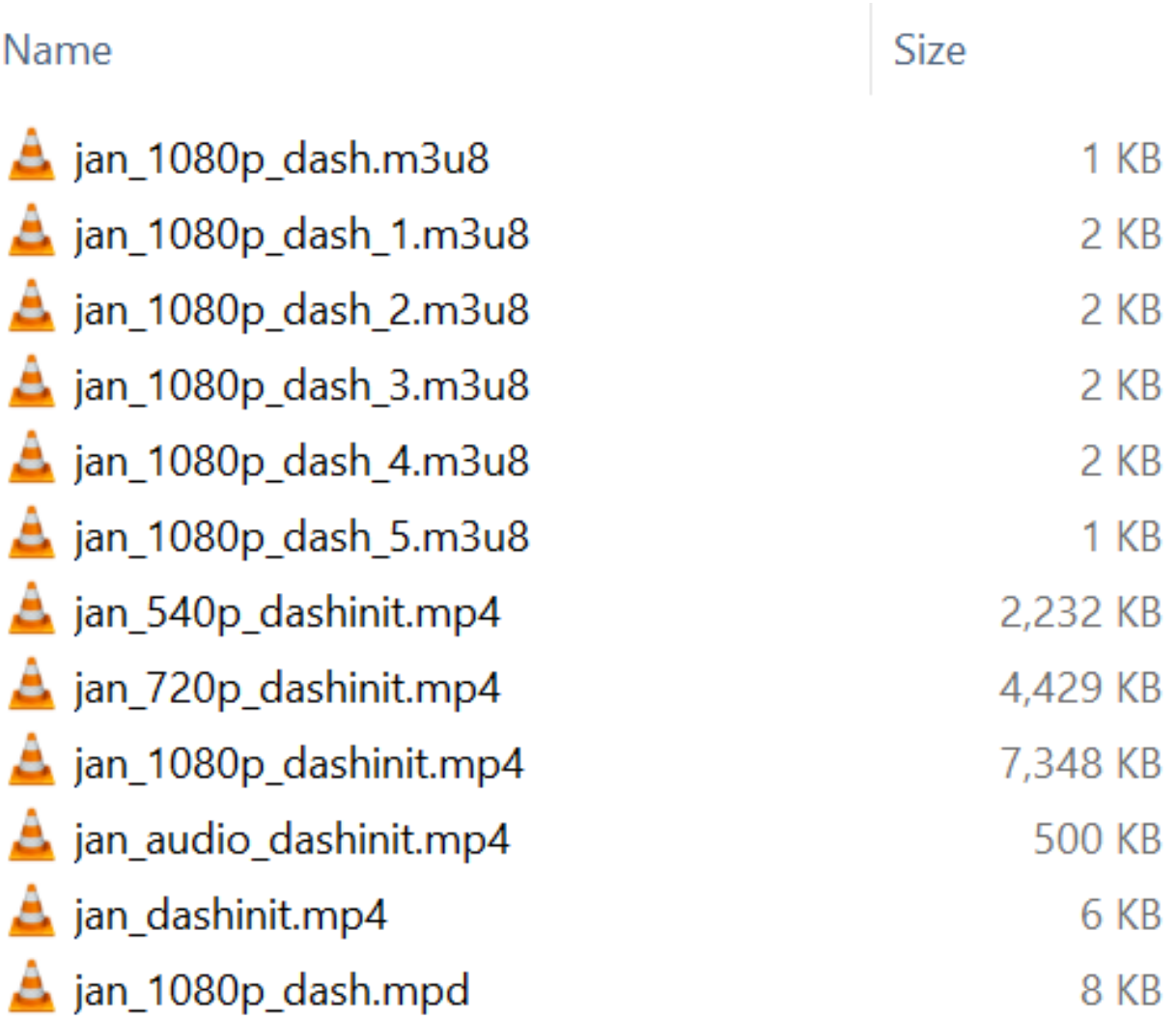
Package to CMAF

main m3u8

media m3u8

shared media files

main mpd



Master Manifest (Just Like HLS)

Video Playlists

Audio Playlist

No - TrickPlay

jan.m3u8 | jan_1080p_dash.mpd | jan_1080p_dash.m3u8

```
1  #EXTM3U
2  #EXT-X-VERSION:6
3  #EXT-X-INDEPENDENT-SEGMENTS
4
5  #EXT-X-STREAM-INF:BANDWIDTH=1998037,CODECS="hev1.1.6.L120.90,mp4a.40.2",RESOLUTION
   =1920x1080,FRAME-RATE=30,AUDIO="audio2",SUBTITLES="subs3"
6  jan_1080p_dash_1.m3u8
7
8  #EXT-X-STREAM-INF:BANDWIDTH=1202760,CODECS="hev1.1.6.L93.90,mp4a.40.2",RESOLUTION=
   1280x720,FRAME-RATE=30,AUDIO="audio2",SUBTITLES="subs3"
9  jan_1080p_dash_2.m3u8
10
11 #EXT-X-STREAM-INF:BANDWIDTH=603932,CODECS="hev1.1.6.L90.90,mp4a.40.2",RESOLUTION=9
   60x540,FRAME-RATE=30,AUDIO="audio2",SUBTITLES="subs3"
12 jan_1080p_dash_3.m3u8
13
14 #EXT-X-MEDIA:TYPE=AUDIO,GROUP-ID="audio2",NAME="4",LANGUAGE="eng",AUTOSELECT=YES,U
   RI="jan_1080p_dash_4.m3u8",CHANNELS="2"
15 #EXT-X-MEDIA:TYPE=SUBTITLES,GROUP-ID="subs3",NAME="5",AUTOSELECT=YES,URI="jan_1080
   p_dash_5.m3u8"
16
```

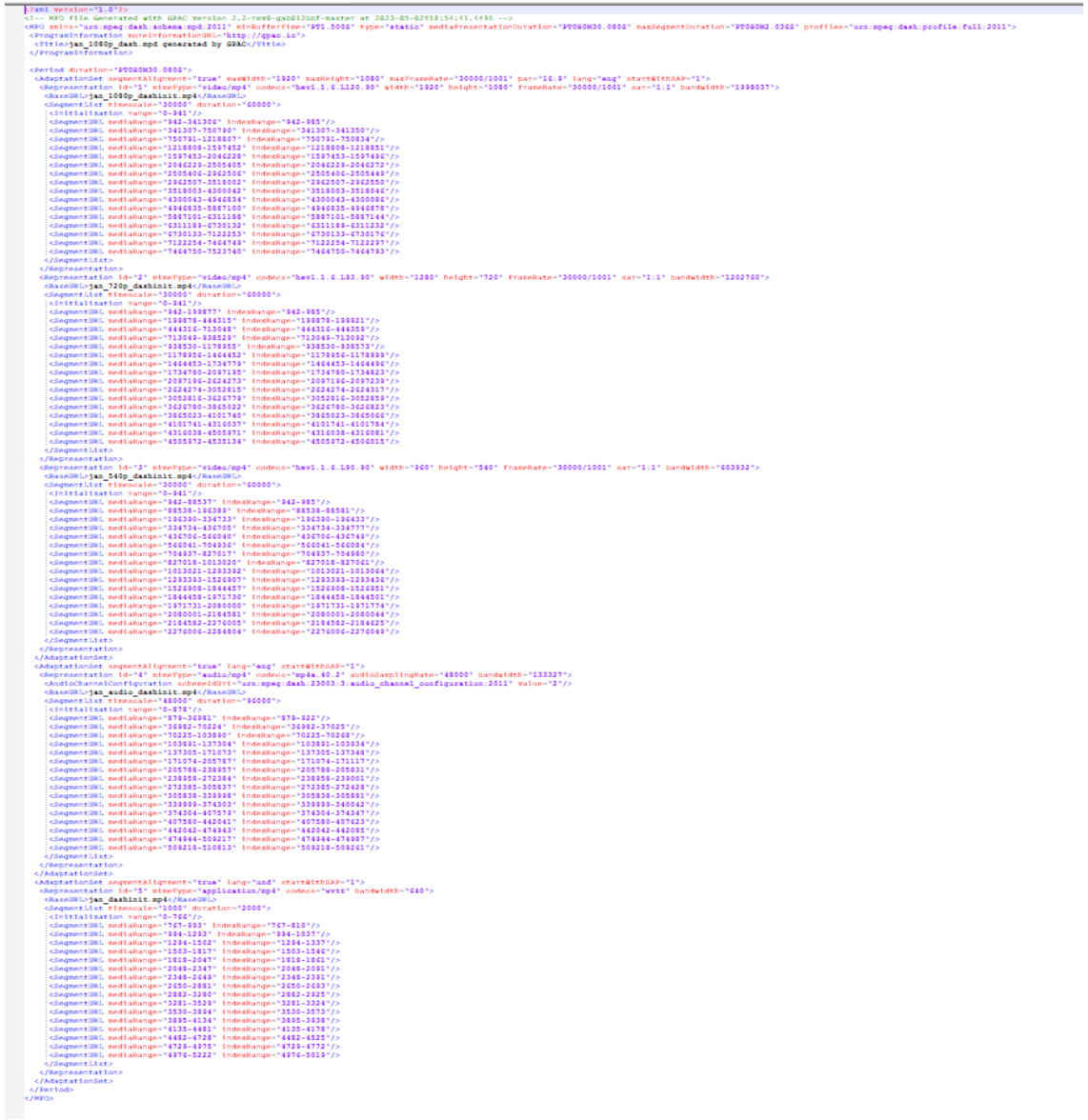
Subtitles with video

Master Manifest

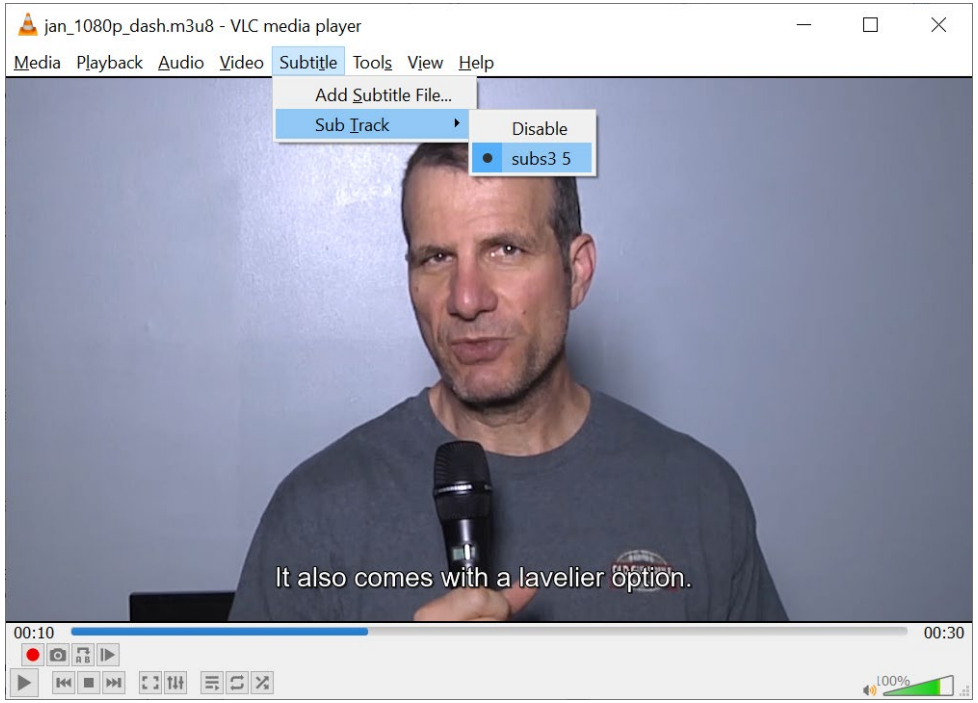
Video(s) byte range

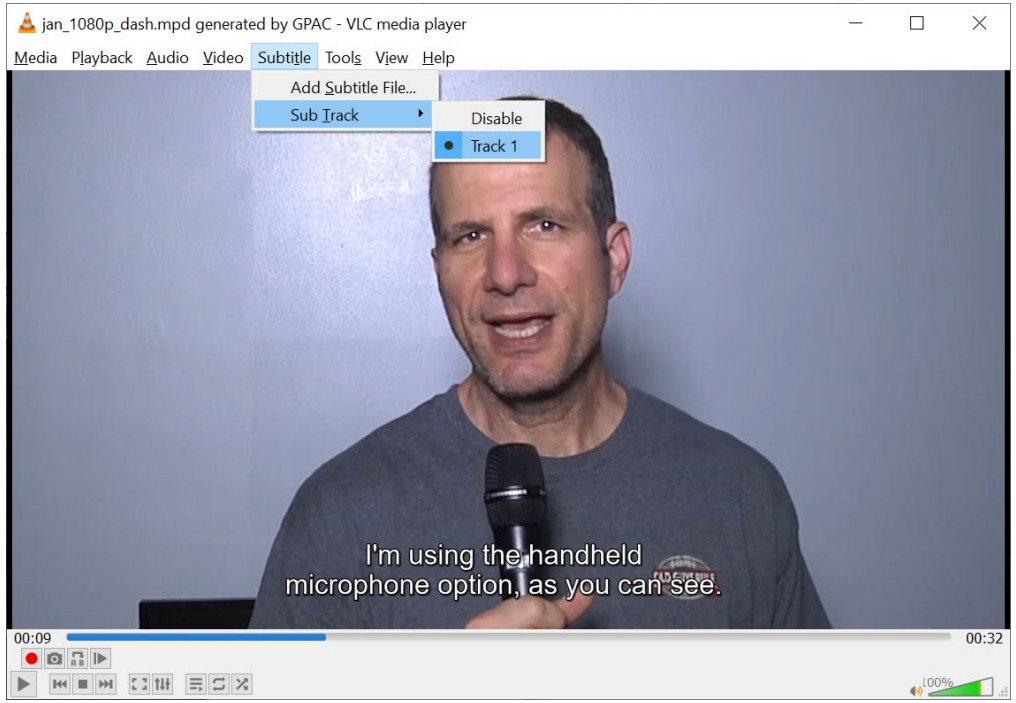
Audio byte range

Subtitle byte range



Playback





- HLS - again,
  - Synched AV
  - Subtitles available
- DASH
  - Synched AV
  - Subtitles available

GPAC Summary

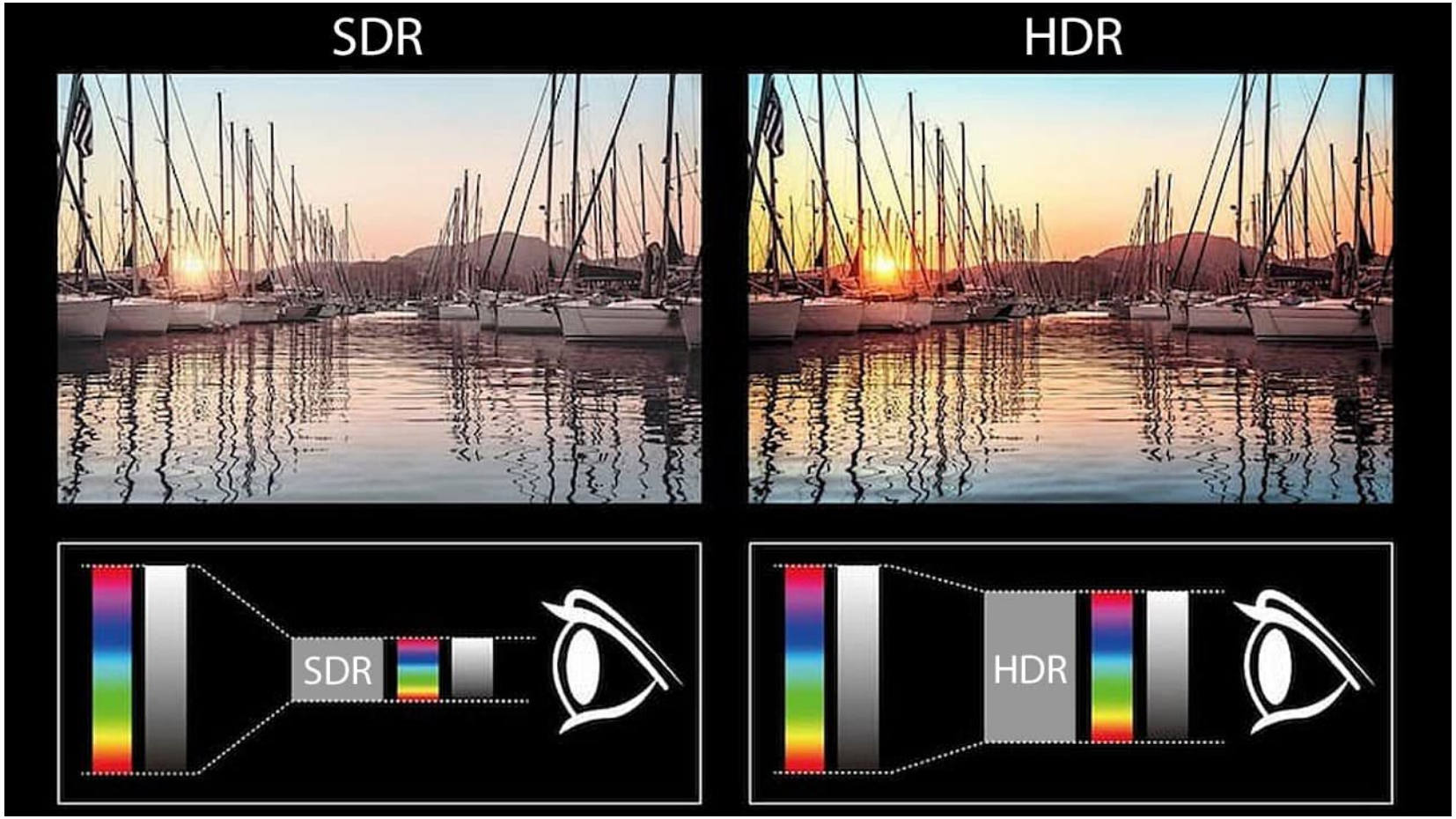
- Very simple to use open-source tool
- Very feature rich
- Works right out of the box
- Vibrant tool, still being developed and supported

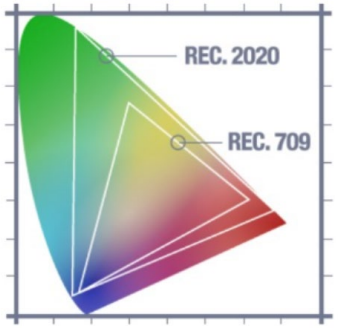
Encoding HDR With FFmpeg

Should be: 5:18

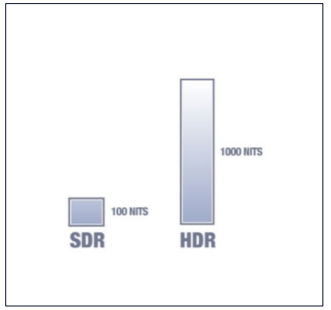
- What is HDR
- The HDR encoding process
- Producing HDR10/HDR10+

Overview: What is High Dynamic Range





More colors



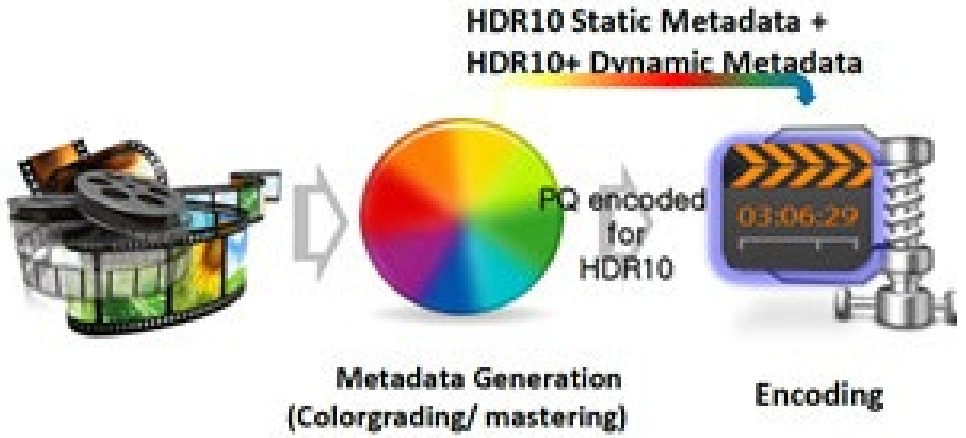
More contrast

https://www.ecoustics.com/ask-an-expert/wtf/wtf-is-hdr-high-dynamic-range/

https://vmi.tv/blog/learn-help/hdr_survival_guide/

Overview: How Does HDR Work? - Metadata

https://thehometheaterdiy.com/hdr/



What is HDR metadata?
HDR Metadata contains the HDR content and mastering device properties (color/brightness)

Lets the display device customize HDR display for its own own **color gamut** and **peak brightness**.

Static

Static metadata – one set of metadata for entire file (HDR 10/HLG)

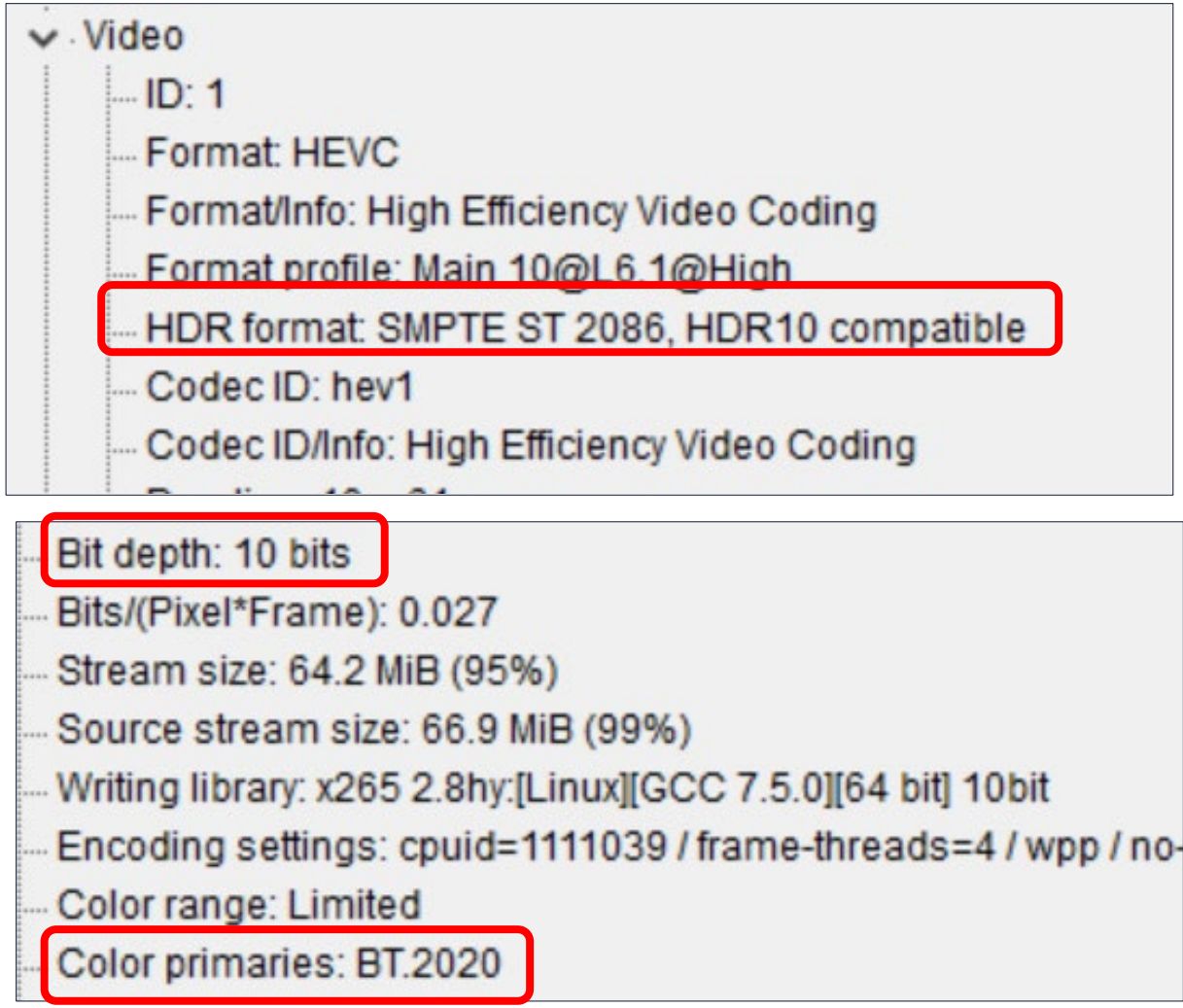
Dynamic

Dynamic metadata – customized scene by scene (HDR 10+/Dolby Vision)

https://bit.ly/Ven_metad

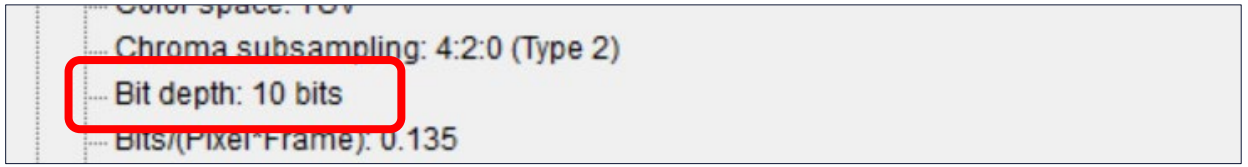
What Do You Need to Produce HDR10

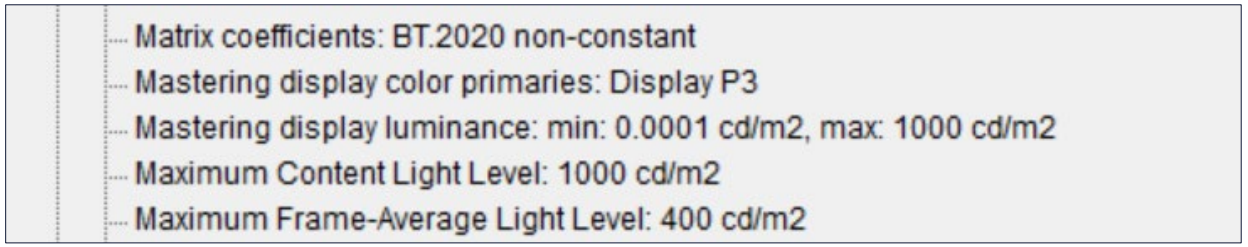
- Source file must contain the metadata:
- Riverplate –
  - SMPTE ST 2086, HDR10 compatible
    - Static: Can produce HDR10 but not HDR10+ or Dolby Vision
  - 10-bits
  - BT 2020 color matrix



What Do You Need to Produce HDR10+

- Source file must contain the metadata:
- HDR10+test_lake
  - SMPTE ST 2094 HDR10+ Profile B
  - 10-bits
  - BT 2020 color matrix
  - Other data

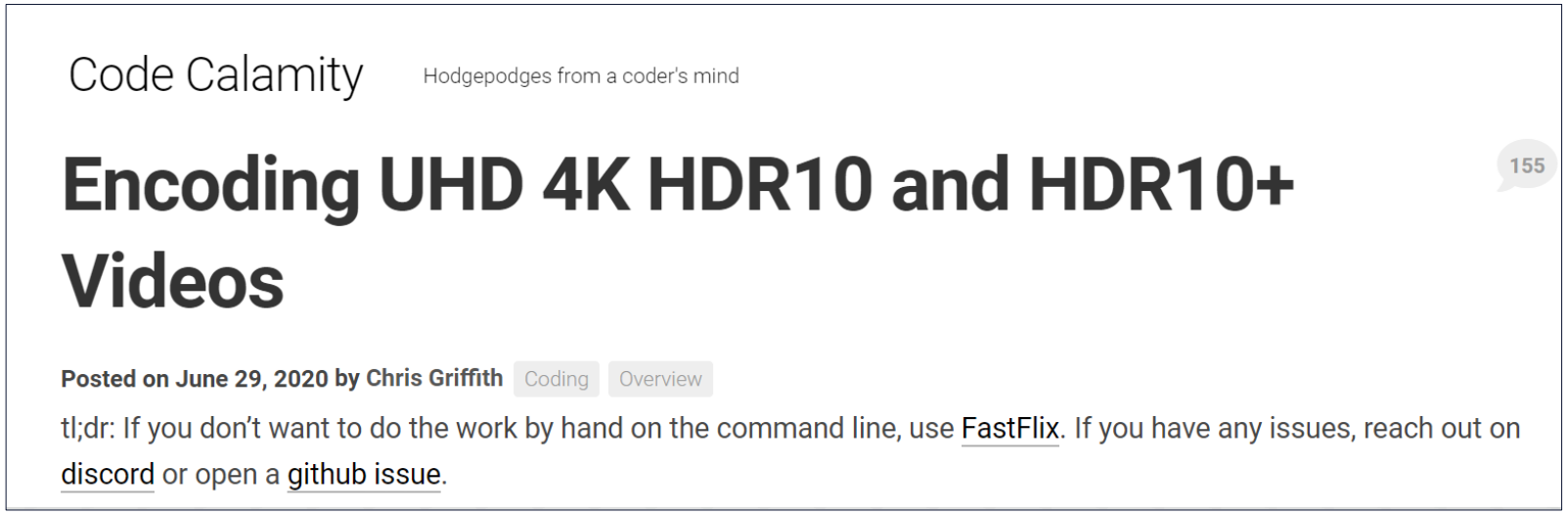




Download test file here:

https://ff.de/hdr10plus-metadata-test/

The Font of All Knowledge



Code Calamity Hodgepodges from a coder's mind

Encoding UHD 4K HDR10 and HDR10+ Videos

155

Posted on June 29, 2020 by Chris Griffith Coding Overview

tl;dr: If you don't want to do the work by hand on the command line, use FastFlix. If you have any issues, reach out on discord or open a github issue.

https://bit.ly/hdr10_plus

- Excellent resource
- Had to work around one or two issues to produce desired output
  - Some scripts / procedures are different (slightly)

Process Overview

1. Extract Metadata
   HDR10 – FFprobe
   HDR10+ - hdr10plus_tool

2. Configure metadata
   HDR10 – manual
   HDR10+ - NA

3. Incorporate into FFmpeg command string

- HDR10
  - Single process for HDR10
- HDR10+
  - Process HDR10 for backwards compatibility (same as for HDR10)
  - Process HDR10+ for devices that accept dynamic metadata

HDR10: Extract Static Metadata with FFprobe

```
ffprobe -select_streams v -print_format json -show_frames -
read_intervals "%+#1" -show_entries
"frame=color_space,color_primaries,color_transfer,side_data_list,pix_f
mt" -i Riverplate_short.mp4 > hdr10metadata.json
```

Key Commands:

- `-select_streams v` - Extract details for video (v) stream
- `-print_format json` - JSON output
- `-read_intervals "%+#1"` - Grab data from the first frame
- `-show_entries ...` - Pick the selected data
- `-i Riverplate_short.mp4` - input (`-i`) is our demo file
- `- > hdr10metadata.json` - output to JSON file

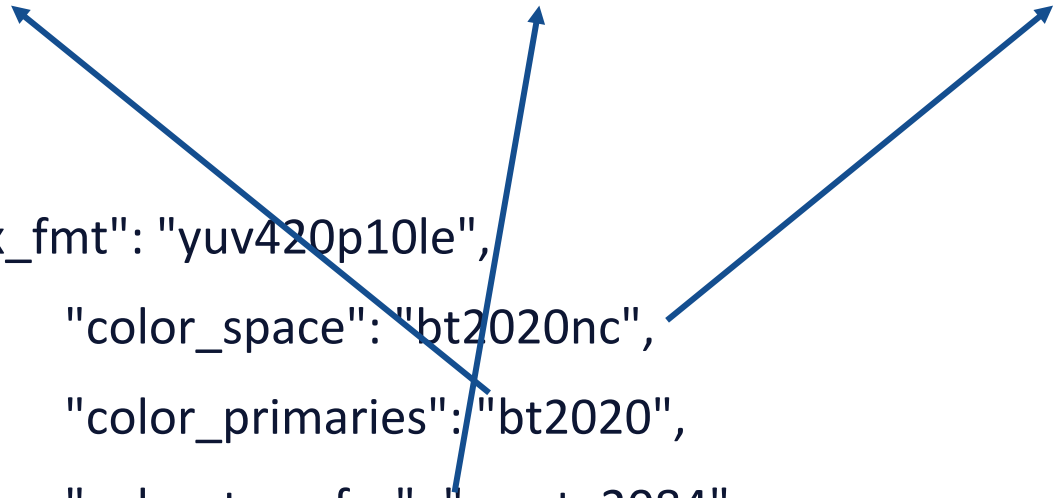
JSON File Populates FFmpeg Script – 3 Data Sources


```
"color_space": "bt2020nc",
"color_primaries": "bt2020",
"color_transfer": "smpte2084",
"side_data_list": [

"side_data_type": "Content light level metadata",
"max_content": 0,
"max_average": 0
```

```
{
"side_data_type": "Mastering display metadata",
"red_x": "34000/50000",
"red_y": "16000/50000",
"green_x": "13250/50000",
"green_y": "34500/50000",
"blue_x": "7500/50000",
"blue_y": "3000/50000",
"white_point_x": "15635/50000",
"white_point_y": "16450/50000",
"min_luminance": "1/10000",
"max_luminance": "10000000/10000"
```

HDR10: Processing the Data – Into Command String

```
colorprim=bt2020:transfer=smpte2084:colormatrix=bt2020nc:
```



- "pix_fmt": "yuv420p10le",
- "color_space": "bt2020nc",
- "color_primaries": "bt2020",
- "color_transfer": "smpte2084",

HDR10: Processing the Data

Process into x/y data

G(13250,34500)B(7500,3000)R(34000,16000)WP(15635,16450)L(10000000,1)


```
{
    "side_data_type": "Mastering display metadata",
    "red_x": "34000/50000",
    "red_y": "16000/50000",
    "green_x": "13250/50000",
    "green_y": "34500/50000",
    "blue_x": "7500/50000",
    "blue_y": "3000/50000",
    "white_point_x": "15635/50000",
    "white_point_y": "16450/50000",
    "min_luminance": "1/10000",
    "max_luminance": "10000000/10000"
```

HDR 10: Integrate Into Command String – Two Pass

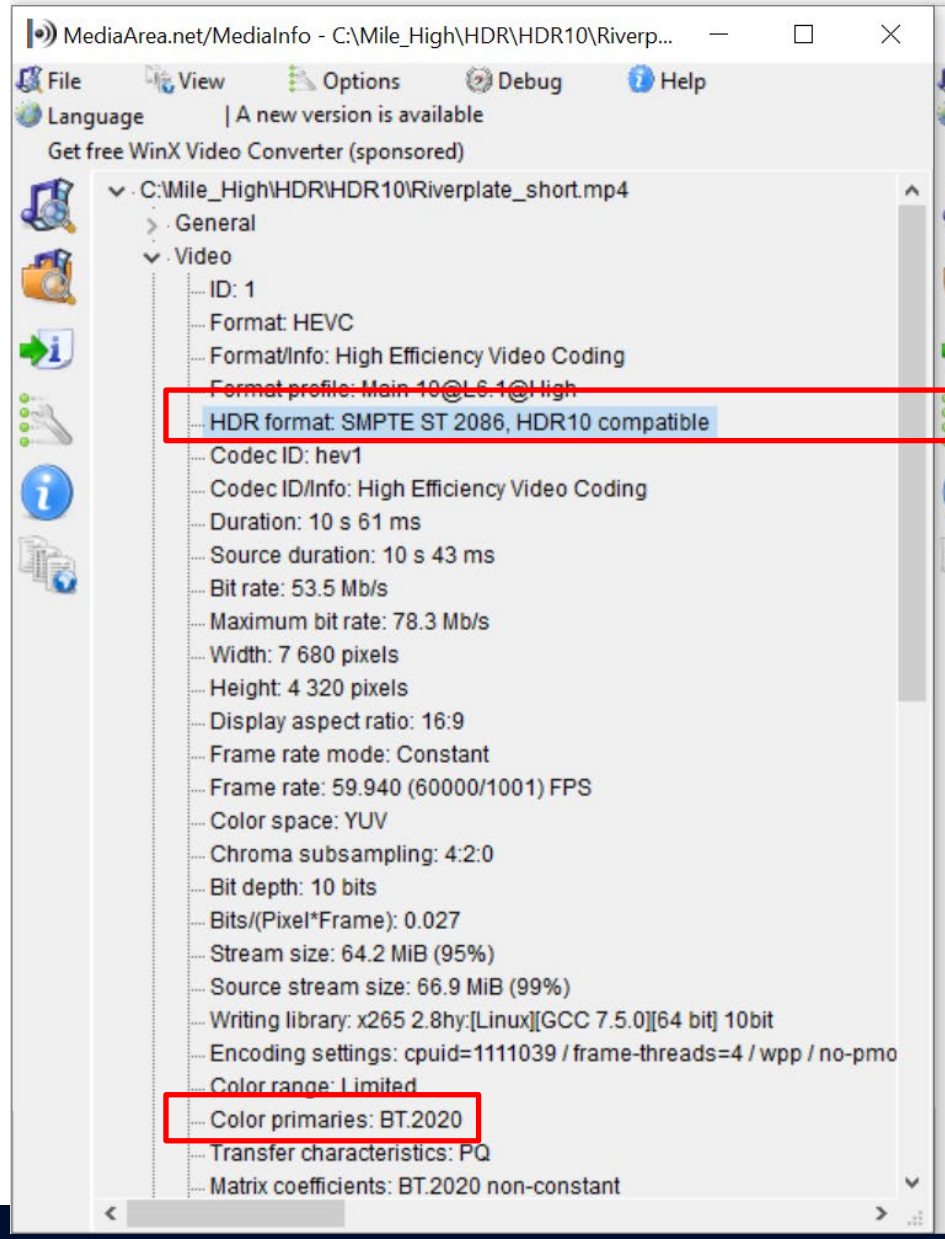
```
ffmpeg -y -i Riverplate_short.mp4 -c:v libx265 -an -preset veryslow -x265-params repeat-
headers=1:colorprim=bt2020:transfer=smpte2084:colormatrix=bt2020nc:master-
display=G(13250,34500)B(7500,3000)R(34000,16000)WP(15635,16450)L(10000000,1):max-
cll=0,0:keyint=60:min-keyint=60:scenecut=0:bitrate=15000:open-gop=0:pass=1 -f mp4 NUL && \

ffmpeg -y -i Riverplate_short.mp4 -c:v libx265 -an -preset veryslow -x265-params repeat-
headers=1:colorprim=bt2020:transfer=smpte2084:colormatrix=bt2020nc:master-
display=G(13250,34500)B(7500,3000)R(34000,16000)WP(15635,16450)L(10000000,1):max-
cll=0,0:keyint=60:min-keyint=60:scenecut=0:bitrate=15000:vbv-maxrate=30000:vbv-
bufsize=30000:open-gop=0:pass=2 Riverplate_short.mkv
```

Key Commands

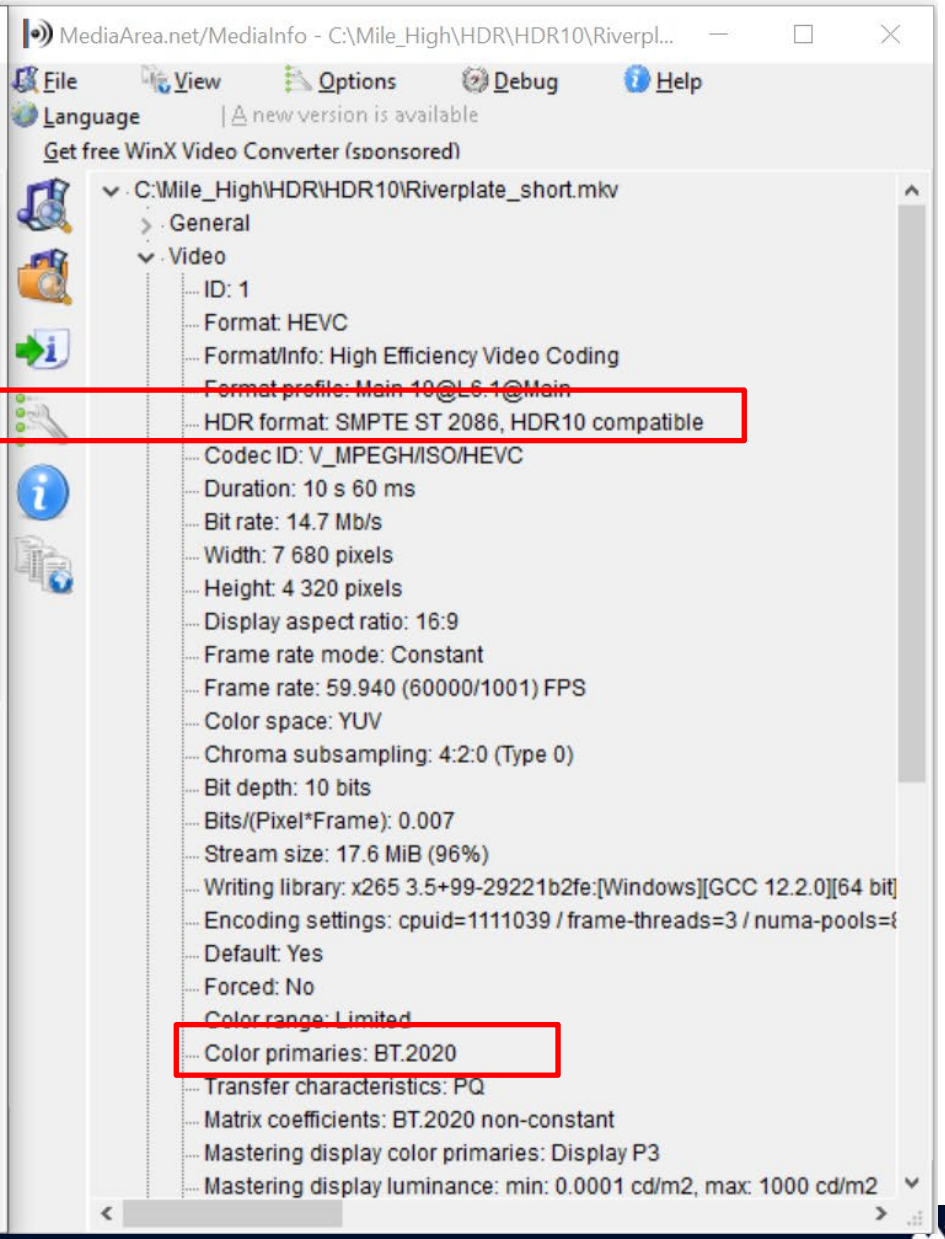
- `repeat-headers=1` – Insert headers on every frame as required
- `colorprim, transfer and colormatrix` - info from ffprobe
- `master-display` - from color string above
- `max-cll` - Content light level data from FFprobe

HDR10: Verify File Data

Source



Output



HDR10+: Step 1: Duplicate for Static Metadata

- FFprobe
- Process data

Extracting HDR10+ Metadata

- HDR10plus_tool – free/open-source (will use to extract metadata)
  - Download here - bit.ly/hdrp_tool
- Operation – a bit different than Code Calamity
  - Documentation here - bit.ly/hdr10plus_tool_doc

HDR10+: Extract Dynamic Metadata with Free hdr10plus_tool

```
ffmpeg -i hdr10+test_lake.mp4 -map 0:v:0 -c copy -vbsf
hevc_mp4toannexb -f hevc - | hdr10plus_tool extract -o
metadata.json -
```

- Simple syntax:
  - `ffmpeg -` Call FFmpeg
  - `-map 0:v:0 -c copy -vbsf hevc_mp4toannexb -f hevc -` Extracts the HEVC bitstream from the source
  - `- | hdr10plus_tool -` Pipes it into the hdr10plus_tool
  - `extract -o metadata.json -` Extracts it to metadata.json

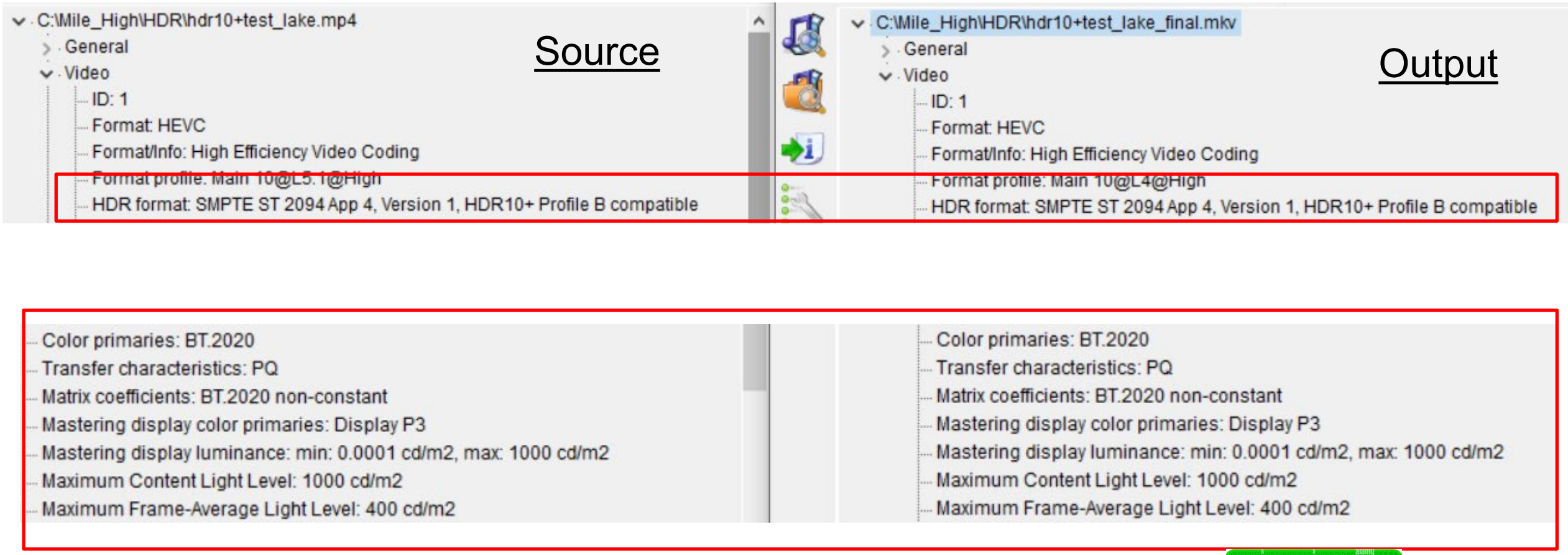
HDR 10+: Integrate Into Command String – Two Pass

```
ffmpeg -y -i hdr10+test_lake.mp4 -c:v libx265 -an -preset veryslow -x265-params repeat-
headers=1:colorprim=bt2020:transfer=smpte2084:colormatrix=bt2020nc:master-
display=G(13250,34500)B(7500,3000)R(34000,16000)WP(15635,16450)L(10000000,1):max-
cll=1000,400:keyint=60:min-keyint=60:scenecut=0:bitrate=15000:open-gop=0:pass=1:dhdr10-
info=metadata.json -f mp4 NUL && \

ffmpeg -y -i hdr10+test_lake.mp4 -c:v libx265 -an -preset veryslow -x265-params repeat-
headers=1:colorprim=bt2020:transfer=smpte2084:colormatrix=bt2020nc:master-
display=G(13250,34500)B(7500,3000)R(34000,16000)WP(15635,16450)L(10000000,1):max-
cll=1000,400:keyint=60:min-keyint=60:scenecut=0:bitrate=15000:vbv-maxrate=30000:vbv-
bufsize=30000:open-gop=0:pass=2:dhdr10-info=metadata.json hdr10+test_lake_final.mkv
```

Key Commands:

- `HDR10 commands` – as before
- `dhdr10-info=metadata.json` – inject metadata file

HDR10: Verify File Data



Questions?