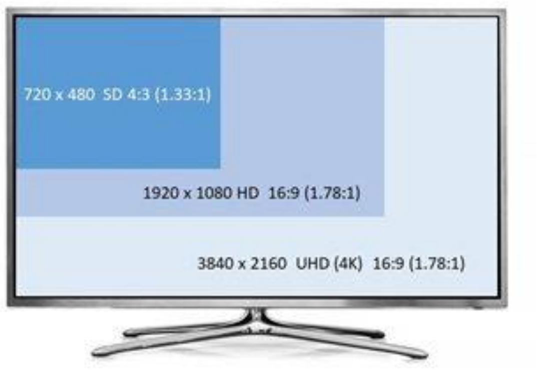
Introduction to Video Quality in AWS Elemental MediaLive

The article explains the AWS Elemental MediaLive settings that influence final video quality (VQ). It provides recommendations on how to configure the settings and/or explains how a particular setting influences the resulting video stream characteristics (latency, VQ, compatibility, stability, delivery cost).

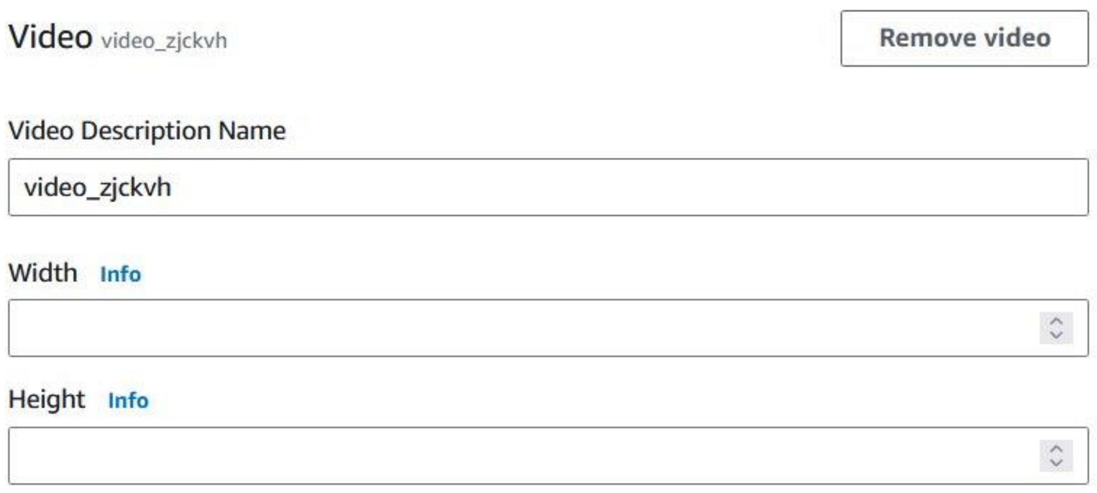
Disclaimers:
- Names and functionality sets might change over time
- Different codecs have different sets of configurations available

1. Video resolution

Resolution is the total number of pixels in a frame of video.


720 x 480 SD 4:3 (1.33:1)
1920 x 1080 HD 16:9 (1.78:1)
3840 x 2160 UHD (4K) 16:9 (1.78:1)

You define resolution as Width and Height parameters in MediaLive. MediaLive will accept any even number in these fields up to Ultra High Definition (UHD): 3840 width x 2160 height pixels. Setting these to a multiple of 16 is recommended as macroblocks are 16x16 in size.


Video video_zjckvh

Remove video

Video Description Name

video_zjckvh

Width Info

Height Info

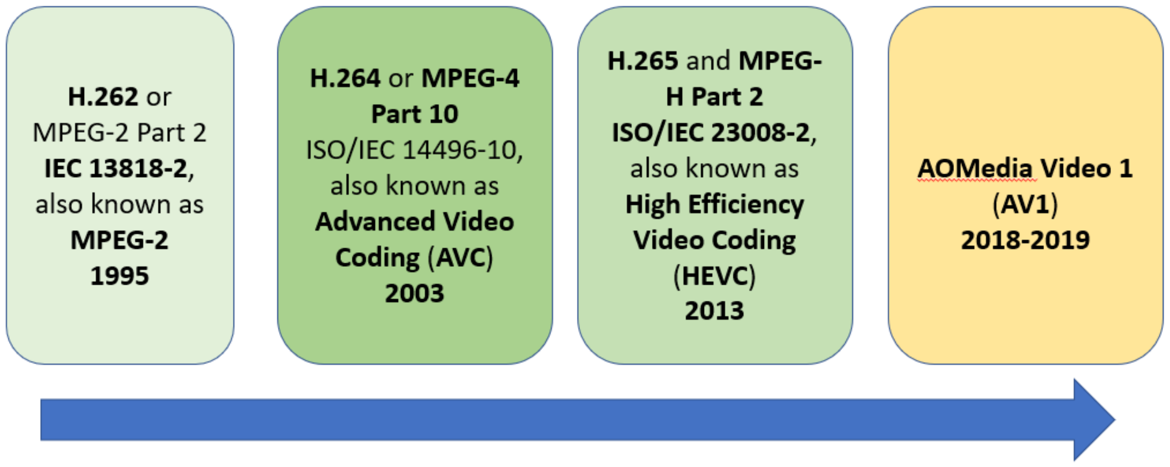
One interesting feature of the Width and Height fields is that you may leave them blank. If you leave both fields blank, then the encoder will use the same resolution as the source. If you leave either one of them blank, the encoder will automatically calculate the other based on the original video aspect ratio or the specified output aspect ratio (more details below in pixel aspect ratio section 3).

In general, higher resolutions such as 1080p (1920x1080) or 2160p (3840x2160) provide a more detailed and immersive viewing experience, but they also require higher bitrates and more bandwidth for delivery, and as a result a higher cost. Lower resolutions like 480p or 576p may be more suitable for content that needs to be accessible on a wider range of devices or for viewers with limited bandwidth.

2. Codec

A codec is a technology or algorithm used to encode and decode digital media, such as audio and video files/streams. Codecs are essential for the efficient storage, transmission, and playback of multimedia content (you can read more about codecs here).

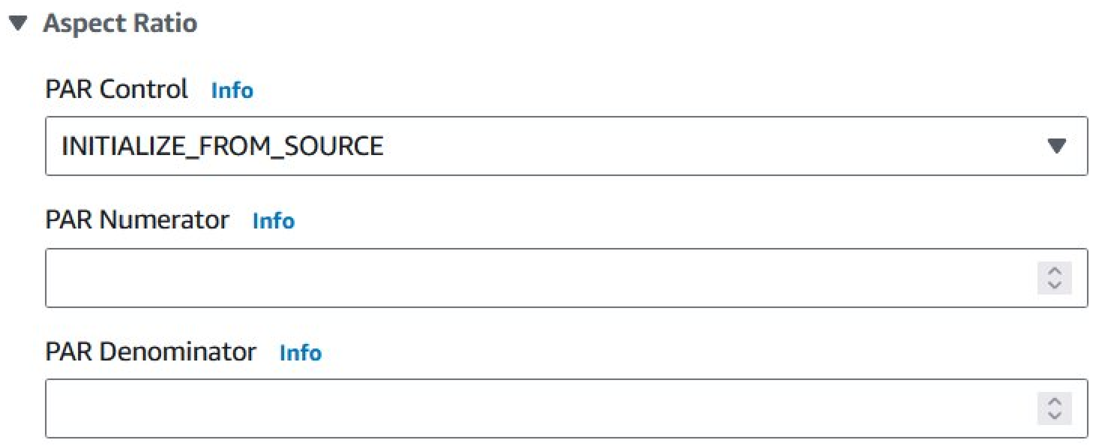
The choice of **video codec** depends on factors such as the target platform, device compatibility, video quality requirements, and file size/bandwidth constraints. MediaLive provides the flexibility to configure the appropriate codec settings to optimize the balance between video quality and resource utilization. MediaLive supports MPEG2 (H.262), AVC (H.264), HEVC (H.265), and AV1 video codecs. A list of supported codecs per output type can be found here.


H.262 or MPEG-2 Part 2 **IEC 13818-2**, also known as **MPEG-2** **1995**
H.264 or **MPEG-4 Part 10** ISO/IEC 14496-10, also known as **Advanced Video Coding** (**AVC**) **2003**
H.265 and **MPEG-H Part 2** **ISO/IEC 23008-2**, also known as **High Efficiency Video Coding** (**HEVC**) **2013**
AOMedia Video 1 (**AV1**) **2018-2019**

In summary, HEVC, when compared with other codecs like MPEG2 and AVC for example, offers superior compression efficiency and video quality, making it a more efficient choice for high-resolution and bandwidth-constrained video delivery. However, AVC is a more widely adopted and supported codec today with broad device compatibility. The choice between the two ultimately depends on the specific requirements of your video streaming application, such as target audience, device capabilities, and bandwidth constraints. We have not considered MPEG 2 in this article as it is a legacy codec with limited usage that has been replaced by AVC. AV1 is the latest codec supported in MediaLive. AV1 has potential for the future due to better compression efficiency than HEVC, but wider adaption is limited because of device support.

3. Pixel aspect ratio (PAR)

Pixel Aspect ratio (PAR) is a number that describes how the width of a pixel in a digital image/frame compares to the height of that pixel. PAR influences the final aspect ratio that the video is displayed on the client device. This option is sent to the client player as a flag in the video stream itself. You can use the same values as the input (initialize from source, available for AVC codec only) or you can define your own (specified):


▼ Aspect Ratio

PAR Control Info

INITIALIZE_FROM_SOURCE

PAR Numerator Info

PAR Denominator Info

- PAR Numerator: This is the first number in the aspect ratio, which represents the width of the video frame.
- DAR Denominator: This is the second number in the aspect ratio, which represents the height of the video frame.

For example, in a 16:9 aspect ratio the PAR Numerator is 16 and the PAR Denominator is 9.

There are three typical use cases for setting a different PAR. First, you may need to correct the aspect ratio of the source video without changing its resolution. Second, you may want to save bits by decreasing the number of pixels being encoded. This technique can improve picture quality in very limited bitrate profiles without affecting the output aspect ratio. For example, you could compress a 1920x1080 video at 1440x1080 and apply a PAR of 4:3; the resulting video is displayed at the same aspect ratio as 1920x1080, but is compressed with fewer pixels in the horizontal resolution (so the bits per pixel ratio increases). Finally, you may change PAR if you are doing an up or down resolution scaling operation, for example high definition (HD) to standard definition (SD) or SD to HD resolution.

To explain how this works we need to introduce more video parameters: Display Aspect Ratio (DAR) and Storage/Sample Aspect Ratio (SAR) in addition to PAR:

- DAR is the display aspect ratio. This is a width (W) to height (H) ratio of viewer targeted screen. For TV, DAR was traditionally 4:3 (a.k.a. full screen), with 16:9 (a.k.a. widescreen) now the standard for HDTV.
- SAR is the storage/sample aspect ratio. This is a ratio of resulting resolution that will be sent to the viewer (also known as output resolution or actual number of pixels). For TV, SAR was traditionally 720x576 (a.k.a. full screen), with 1920x1080 (a.k.a. widescreen) now the standard for HDTV.

This formula shows the dependencies between these parameters: PAR = DAR/SAR.

To better understand why this is important, we will demonstrate several real use case examples.

1. High-definition TV (HDTV) use case:

   DAR = targeting screens are predominantly 16:9
   SAR = output resolution = 1920x1080
   PAR = DAR/SAR = (16:9)/(1920:1080) = 1:1

2. Standard definition TV use case:

   DAR for targeting screens for this case are predominantly 4:3 in the past and there was a lot of content created to support it. Then HDTV came and 16:9 screens became predominant. This complicated things when maintaining proper proportion for SDTV channels where content is a mix of old (4x3) and new (16x9) formats.

   DAR (for old content) = 4:3
   DAR (for new content) = 16:9
   SAR = output resolution = 720x576

   PAR (for old content) = DAR/SAR = (4:3)/(720:576) = 48:45 = 16:15
   PAR (for new content) = DAR/SAR = (16:9)/(720:576) = 64:45

   Additional examples are shown in table below:

DAR	**SAR**	**PAR**	**PAR (decimal)**	
4:3	704x576/11:9	12:11	1.09	PAL*
4:3	720x576/5:4	16:15	1.06	
16:9	704x576/11:9	16:11	1.45	
16:9	720x576/5:4	64:45	1.42	
4:3	740x480/22:15	10:11	0.9	NTSC*
16:9	704x480/22:15	40:33	1.21	
16:9	1440x1080/4:3	4:3	1.(3)	
16:9	1920x1080/16:9	16:9	1.(3)	HD TV

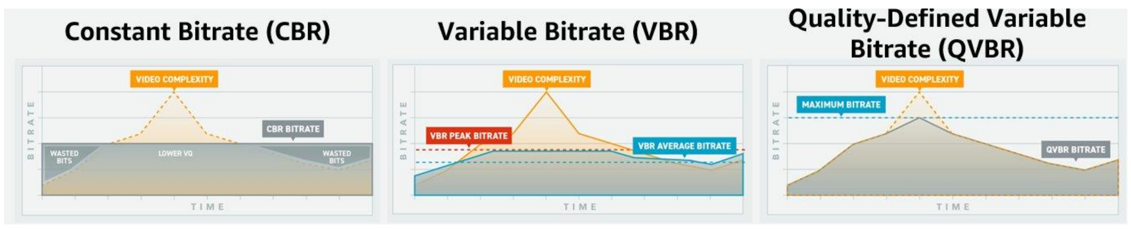
* PAL and NTSC are legacy analog video formats.

4. Rate control

This option controls how the encoder uses available bits set by the bitrate, or commonly known as how encoder varies the bitrate. Regardless of the option you choose you will always get an

Average Bitrate, which is essentially calculated by dividing the size in bits of the entire video by its duration in seconds.

MediaLive supports the following rate control modes for video encoding:



CBR (Constant Bitrate): You to choose a bitrate and the video is encoded at that bitrate, regardless of the complexity of the content. The resulting Average Bitrate is the bitrate you tell the encoder to use. This option is commonly used to maintain compatibility with players that don't support the varying bitrate options below.

VBR (Variable Bitrate - not available for the HEVC codec in MediaLive): With VBR, the encoder applies more bits to complex scenes while saving bits on easier to compress scenes. This option was recommended for mixed content in the past and used in traditional broadcast environments.

QVBR (Quality-Defined Variable Bitrate): QVBR is a more efficient evolution of VBR. QVBR allows the encoder to vary the bitrate while trying to achieve a desired picture quality. QVBR uses the frame complexity, max bitrate, and quality level to decide how many bits to assign to frames. The biggest difference between VBR and QVBR is that the encoder will not attempt to reach a specified average bitrate with QVBR.

Let’s look at an example: imagine a sequence of 10 frames where 8 of them are very complex. With VBR, the encoder will remove bits from the 2 easy frames and then distribute the bits among the 8 the remaining. With QVBR, you provide a “Max Bitrate”, so the 8 frames would get the maximum number of bits they can for being hard to compress. In VBR, the encoder is limited by the average bitrate, so the more complex frames you have, the less bits each one of them gets. In QVBR, the encoder is limited by a Max Bitrate, so the encoder will stay in the “max bitrate” range for as long as it needs to. For this reason, resulting QVBR average bitrates can vary.

QVBR mode requires you to configure both **Quality Level** and **Max Bitrate**. If you don’t specify a target quality level, MediaLive operates in QVBR Auto Quality mode and determines the target quality automatically.

Max Bitrate: The encoder respects the maximum bitrate you provide in order to keep the video within an acceptable size.

QVBR Quality Level: Recommended settings are 6-10. Smaller screens from a human perception perspective are less sensitive to encoding artifacts. Larger screens more clearly show any encoding artifacts, so if you don't want your viewers to notice a quality drop, the encoder would have to use higher bitrates. So lower Quality Level allows the encoder to decrease quality for smaller screens where the human eye will not see the difference and save more bits on delivery and storage.

QVBR Auto Quality Level: With this option, the encoder will select the quality level automatically and can also vary the quality level on a scene-by-scene basis. The advantage of selecting this option is that the encoder can produce more consistent quality across the entire video than forcing a fixed target Quality Level. To enable this mode, leave the field empty.

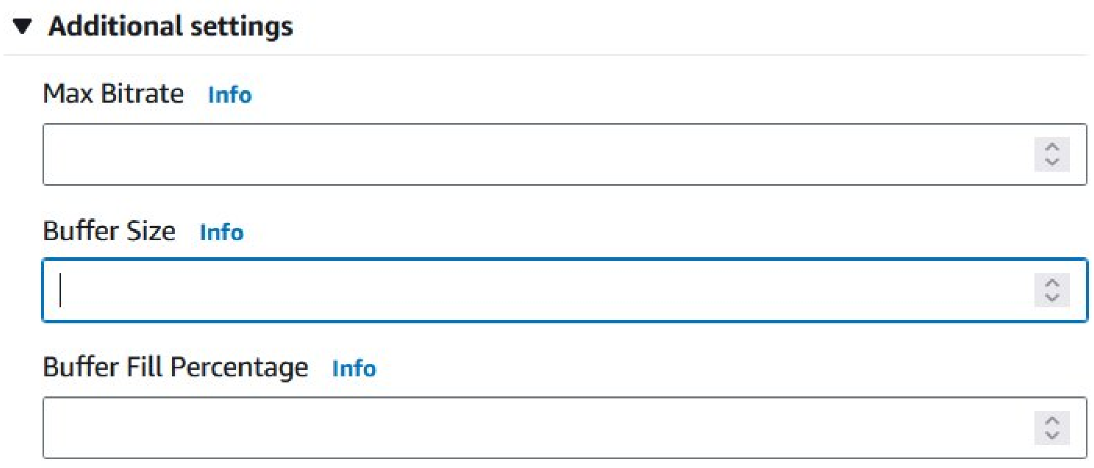
For reference, the **Multiplex** rate control mode is only supported (and is required) when the video is being delivered to a MediaLive Multiplex. In this case, the rate control configuration is controlled by the properties within the Multiplex Program.

The choice of rate control mode in MediaLive depends on your specific requirements, such as the target video quality, the available bandwidth, and the importance of maintaining a consistent bitrate (usually for compatibility) or achieving optimal cost-effective content delivery.

In general, QVBR is considered the most versatile and recommended option, as it can provide a good balance between video quality and efficient bandwidth usage. However, the other rate control modes may be more suitable for certain use cases or scenarios (legacy devices as an example).

5. Bitrate

Bitrate and Resolution are the two most important settings that affect video quality. The bitrate determines how many bits each frame will get; the more bits, the better the video quality is. As starting point for over the top (OTT) delivery you can review the recommendations for HLS streaming protocol here.


▼ Additional settings

Max Bitrate Info

Buffer Size Info

Buffer Fill Percentage Info

6. Buffer Size and Buffer Fill Percentage

Buffer Size in MediaLive refers to the reference hypothetical decoder (HRD) buffer size, which is used to model the behavior of a real video decoder. This buffer size setting is particularly relevant when working with variable bitrate (VBR) video encoding.

The HRD buffer helps ensure the video stream can be properly decoded and played back without underflow or overflow of the decoder's buffer. It models the buffer requirements of the video decoder and helps maintain the timing and synchronization of the video and audio components. The HRD buffer size setting helps ensure the video stream complies with the specified bitrate constraints and avoids exceeding the decoder's buffer capacity.

In MediaLive, you can configure the HRD buffer size in bits, with possible typical values ranging from 1,000 bits to 100,000,000 bits. The recommended HRD buffer size often depends on factors like the video codec, resolution, and overall bitrate being used.

Higher bitrates generally require larger HRD buffer sizes to accommodate the increased data flow and avoid buffer underflows or overflows. So, if the bitrate is very high (such as higher than 5 Mbps), we recommend that the buffer size be at least twice as large as the bitrate. If the bitrate is lower, you can experiment with 2 to 3 times the bitrate to see if you have any quality gains. Keep in mind two important caveats: the player (decoder) would have to support the buffer size you desire, and filling the buffer takes time which increases the delay. For this reason, we typically recommend a 90% Initial Buffer Fill (not available for HEVC), but you may need to reduce that percentage for latency sensitive content such as sports. This delay is important at the very beginning of the stream only when the operator presses “Start” on the channel. The Buffer Fill has no impact on viewers that connect to the stream after the stream has already started.

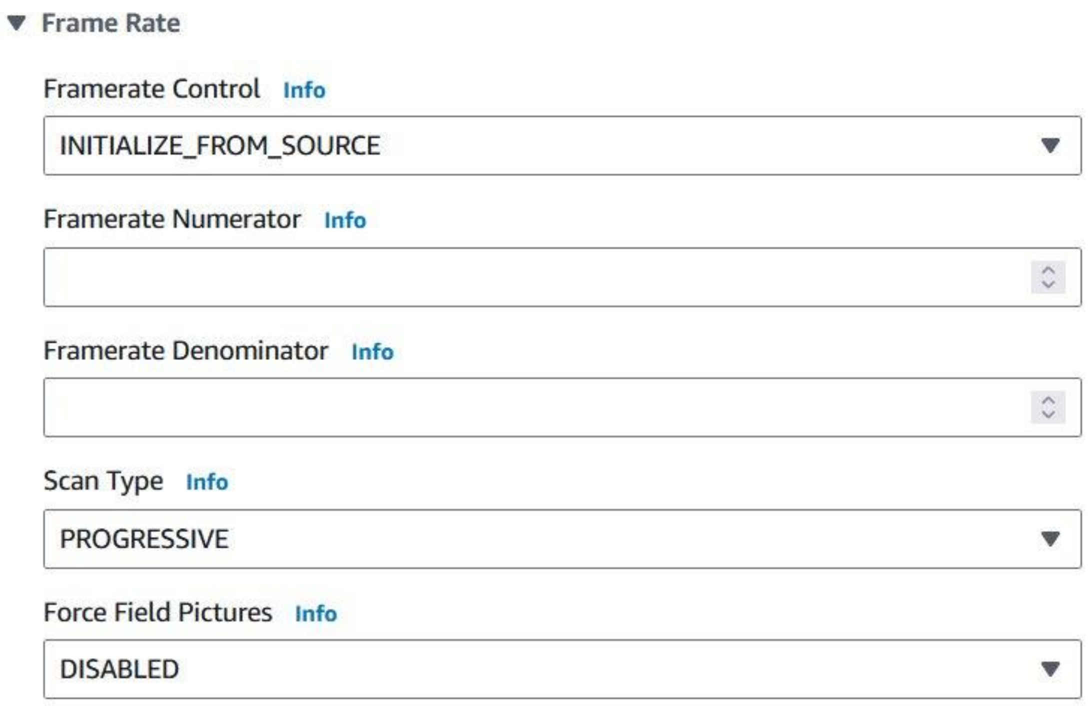
The HRD buffer size also impacts the overall latency of the video stream, as a larger buffer size can introduce additional delay in the decoding and playback process.

7. Frame Rate

Frame rate is sometimes referred as "temporal resolution", which is the information over time that creates the illusion that objects on a screen are moving when playing a video. In general, the higher the frame rate, the smoother the motion. This means that a ball on a tennis court, for instance, will "appear" in more places across the screen until it reaches the other side on a 50 or 60 frame per seconds (fps) video compared to 25 or 30 fps videos.

Before we continue our discussion, it is important to note how AWS Elemental represents the frame rate. The AWS Elemental ecosystem always shows the **frames** per second number, regardless of the frame type: interlaced or progressive. We will then denote the frame type with an "i" or a "p". For example, 29.97p is almost 30 frames per second progressive, which means that almost 30 full frames will be displayed every second. 29.97i is almost 30 frames per second interlaced, however, others may refer to this as "60 fields per second". Keep in mind that MediaLive always shows frames per second, while third party tool/solutions might show fields per second for interlaced use case.

This information is important when you choose the option "Initialize from Source" on your frame rate and enable the deinterlacer by selecting Scan type progressive. What Initialize from Source with Deinterlacer does is transform either 25i into 25p or 29.97i into 29.97p. Notice that the frame rate is the same as the source, but we deinterlaced thus going from "i" to "p".



▼ Frame Rate

Framerate Control Info

INITIALIZE_FROM_SOURCE

Framerate Numerator Info

Framerate Denominator Info

Scan Type Info

PROGRESSIVE

Force Field Pictures Info

DISABLED

Framerate Numerator: This is the first number in the frame rate ratio, which represents the number of frames per second.

Framerate Denominator: This is the second number in the frame rate ratio, which represents the time scale (usually 1 second).

For example, if the frame rate is set to 30 frames per second, the settings would be:

- Framerate Numerator = 30
- Framerate Denominator = 1

This means the video is playing at a rate of 30 frames per second.

Some other common frame rate ratios include:

- 24 frames per second: Numerator = 24, Denominator = 1
- 25 frames per second: Numerator = 25, Denominator = 1
- 29.97 frames per second: Numerator = 30000, Denominator = 1001

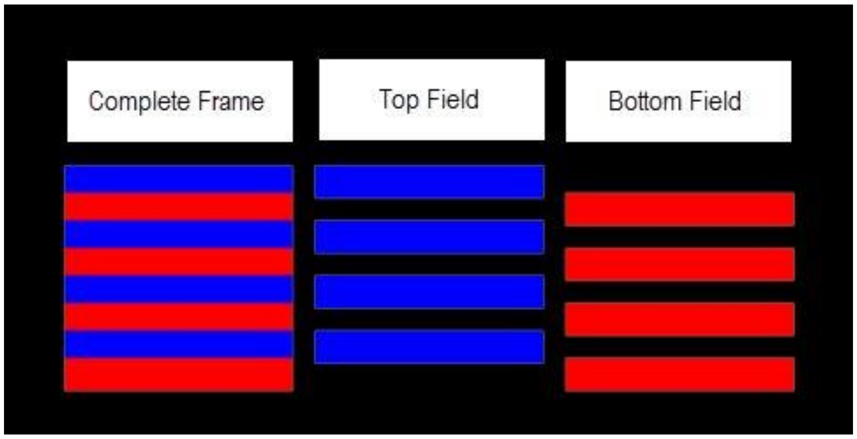
Higher frame rates generally require more bandwidth on delivery and storage space, as each additional frame adds to the overall data required to represent the video.

If "InitializeFromSource" is selected then the output video frame rate will be set equal to the input video frame rate of the first input.

In order to achieve a smoother temporal resolution (or same motion as watching an interlaced video on a traditional TV) you can specify the frame rate as either 59.94p or 50p manually. Note that a higher framerate will require higher bitrate as the data to process will be increased.

8. Scan type

You have option to select what output scan mode do you need to use by selecting one of two options: interlaced or progressive:



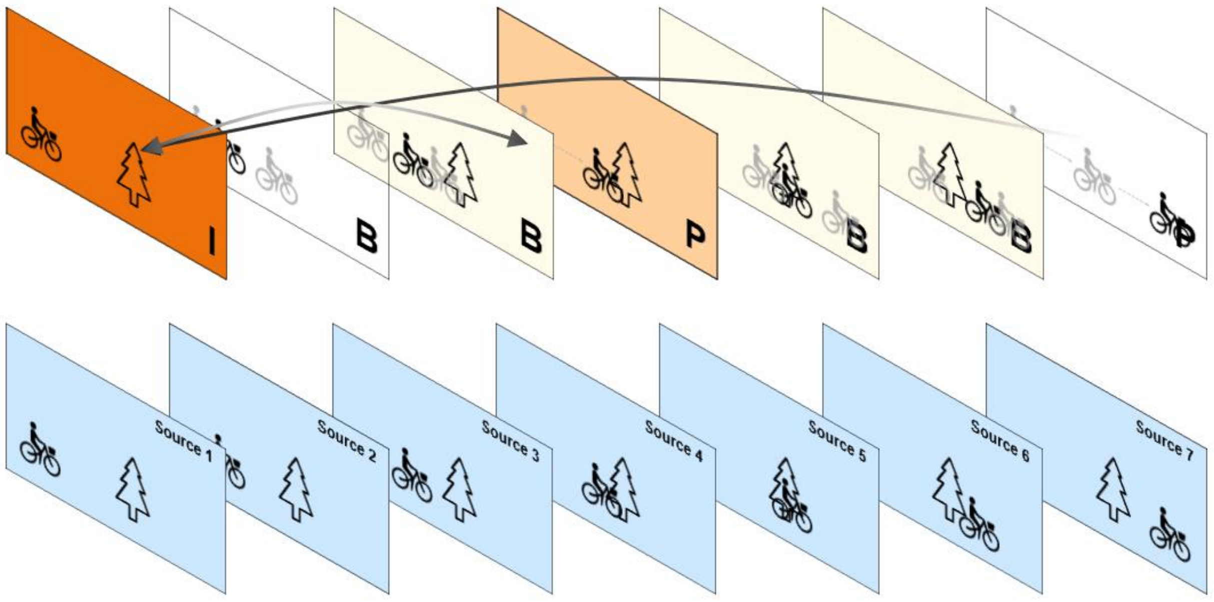
This option tells the encoder how to process the output video. If you set this to Progressive and the source is Interlaced, the de-interlacing filter will be automatically applied. Sometimes input source has incorrect or corrupted metadata and might not trigger the de-interlacing process. Not performing a de-interlacing step in the signal can cause the output video to display unwanted artifacts such as "combed" lines or ghosting in motion (when an object is currently occupying a new space in the frame area but you can still see a faint image of where it was in previous frames).

When you enable Force Field Pictures (available for AVC codec only), MediaLive codes on a field basis, and odd and even sets of fields are coded separately or together (on a frame basis using Picture-adaptive frame-field (PAFF) coding), depending on what is most appropriate for the content. When the video is progressive, the coding is always performed on a frame basis.

In summary, progressive scan is more prevalent video format today, providing a higher-quality, more stable, and more compatible video experience compared to the legacy interlaced scan format.

Interlacing was a common practice introduced by analog TV broadcasting. The first digitally encoded videos supported interlacing in order to be compatible with the TVs that existed at the time. Most TV displays are actually progressive today and most OTT specs don't allow interlaced content.

9. GOP Structure and GOP Size



A GOP (Group of Pictures) is a collection of frames between two I-Frames (ignoring scene change I-Frames). GOP can be defined in frames or in seconds, and can either be "open" or "closed" (see Closed GOP Cadence below). Open GOPs may have frames that reference a frame from a different GOP, while in Closed GOPs the frames reference other frames within the GOP itself. Because the frames in this group share data, certain types of content can benefit from larger GOPs.

Let's take an interview, for example. In this type of content, the camera spends some time filming the interviewer while he/she asks the question. Then the camera cuts to the interviewee that will also take a certain amount of time to answer the question. If the GOP is sufficiently large, the encoder can tell the frames either on the interviewer or in the interviewee to share data and it wouldn't have to refresh everything on each cut with a large (and redundant) I-Frame so often. Another example is a football match. The camera often pans from one side of the field to the other for a good amount of time. This means all the information on the field and the crowd in the background could be shared between the frames inside this GOP and the encoder wouldn't have to re-create them on an I-Frame so frequently. This strategy helps the encoder save bits, thus improving video quality or decreasing the bitrate necessary to encode (saving bandwidth). To learn more about GOP structure, please read this blog.

It is also worth mentioning that GOP size has an effect on latency, which can be undesirable for latency sensitive content like sports matches. In the OTT world, the segment sizes have to be integer multiples of the GOP size (excluding "parts" concept for Low Latency HLS specification). That means longer GOPs create longer segment sizes and, therefore, longer delays. So, for sports matches specifically, customers typically require a smaller GOP and so we need to focus our quality efforts elsewhere. In the traditional TV world, such as cable TV, the GOP sizes influence how fast a set top box (STB) can tune into the channel. The STB can only begin displaying a frame on TV when it finds a new GOP. If the GOP is too long and the viewer tuned into the channel right after a new GOP had already begun, the STB would have to wait until a new GOP arrives to display something. So, for traditional TV workflows, GOPs are usually small for all channels.

To achieve seamless transition for certain use cases without introducing artifacts through audio gaps or video gaps, it is important that audio and video I-frames are aligned at the beginning and end of segments. It is also important to have compatible video and audio segment durations so that the segment ends are aligned time wise.

To calculate this duration, it is important to know that an AAC-LC audio frame (taken as an example most common delivery audio codec for now) consists of 1024 samples. This means that one frame of AAC audio with a sample rate of 48 Kz is 1024/48000 seconds long. One frame of video duration is simply 1 divided by the frame rate (1/25 seconds for 25 frame per seconds example).

We need to find a common segment duration to carry integral number of audio 1024/48000 and video 1/25 frames. To do this, we need solve this formula:

```
x * 1/25 = y * 1024/48000

x/y = 8/15

8 * 1/25 = 15 * 1024/4800
```

Min duration will be 0.32 seconds. But in practice the recommended GOP duration for video encoding is 2 seconds.

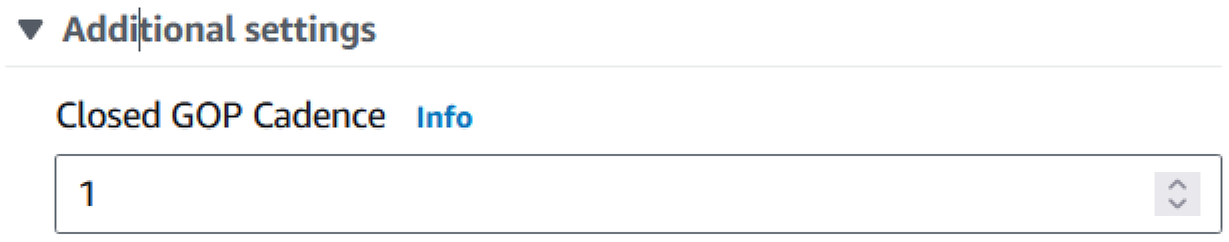
```
2 sec /0.32 sec = 6.25
```

Because we need integral number, the closest one is 6. Which finally, bring us to the magic number 1.92 seconds (6*0.32= 1.92 sec) or 48 video frames or 90 audio frames. More examples below:

Segment duration in sec	Number of Video Frames	Number of Audio Frames	Video Frame Rate, Hz	Number of audio frames per packet size in AAC -LC audio codec
1.92	48	90	25	1024
3.84	96	180	25	1024
1.6	48	76	30	1024
4.8	144	225	30	1024

10. Closed GOP Cadence

A closed GOP is a group of frames where the first frame (the I-frame) is self-contained and does not rely on any previous frames for decoding. This is in contrast to open GOPs, where the first frame may depend on frames from the previous GOP.

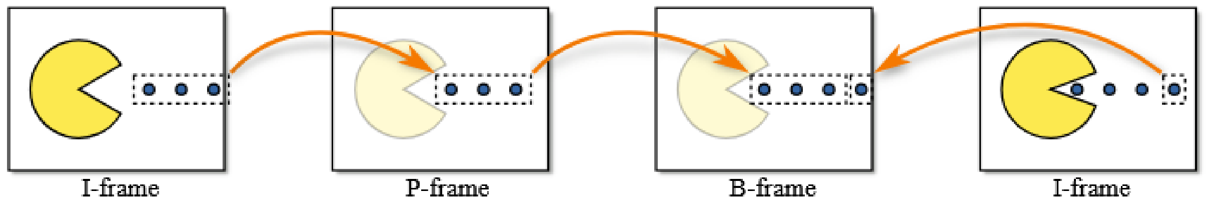


Closed GOP Cadence Info
1

OTT delivery requires a closed GOP, that is better suited for adaptive bitrate streaming, as they allow the player to switch between different bitrate versions of the video without causing visual artifacts or disruptions. Because of this, the Closed GOP cadence number must be an integer multiple of the GOP and segment length. So, if GOP = 2 seconds and segment = 4 seconds, Closed GOP cadence can be 1 or 2 (theoretically this could improve quality on certain kinds of content). If GOP = 2 seconds and segment = 2 seconds, the Closed GOP cadence has to be = 1. We recommend following these rules even for non-OTT delivery.

11. B-Frames

When encoding a video, three different frame types are used. **I-frames** are independently decodable and do not reference other frames (like a JPEG image). **P-frames** only reference past frames. **B-frames** are coded out of order and reference both past and future frames. That "bidirectional" reference helps the encoder save data by requiring less I-Frames to be inserted, or lowering the size of the "non-I-Frames" in your GOP.



Let's take the example of Pac-Man. The first frame is an I-frame that brings the whole picture of the Pac-Man, its contours, color, and the dots he is about to eat. In the next frame, Pac-Man and the dots are still there, but the dots moved a little bit towards Pac-Man. For this frame, the encoder still doesn't need future frame data since everything it needs was already described in the previous I-frame. Therefore, it creates a P-frame to simply describe how the dots have moved. In the following frame, the dots have moved enough to reveal a new dot (or information) is coming. So, in this frame the encoder could save bits if it references a future frame - which would contain this new information - by creating a B-frame. This B-frame references both past and future reference frames since it is an intermediate step between them. For that reason, B-frames are coded out of order. For a player to decode a B-frame, it must have already received all the references this B-frame needs (including the "future" one). This strategy benefits contents with lots of camera pans such as sports, where the camera usually follows athletes back and forth.

B-Frames help for most types of content and they should be more than 0 if you are using Main or High profiles. We typically recommend 1 or 2 B-Frames if GOP Reference B-Frames is disabled and 3 or 5 if GOP Reference B-Frames is enabled.

12. Dynamic Sub-GOP

While B-Frames help in most cases, there are cases when you would get better quality by lowering the number of B-Frames, such as in very complex motion sequences. Dynamic Sub-GOP allows the encoder to vary the number of B-Frames specified in an attempt to preserve quality when it detects such a case. We have been constantly improving the algorithms in our video codec, so it is safe to assume that enabling Dynamic Sub-GOP improves picture quality, so you should use Dynamic mode for most use cases. You may want to select Fixed mode for compatibility issues with legacy decoders.

13. Reference Frames

The number of reference frames tells the encoder how many other frames a given frame is capable of referencing. So, if you set Reference Frames to 3, for example, any given P or B-Frame will be able to reference up to 3 other frames. The number of frames you set on this field influences the strategies the encoder uses to choose a reference.

The typical value for this field that works for almost all content is 3. Setting this to anything higher provides no visible quality gains but could have an impact on processing speeds.

14. GOP Reference B-Frames

This option simply allows the encoder to use a B-Frame as a reference frame as well. When this option is enabled, we recommend setting the number of B-frames to 3 or 5. Note that this might not be compliant with legacy codec implementations and requires end device testing.

15. Scene Change Detection

Scene Change Detection allows the encoder to insert an I-Frame mid-GOP, without breaking its cadence and without having to wait for a new GOP to insert this type of frame. In the codecs that support multiple types of frames, not all I-Frames describe the beginning of a GOP. GOPs are marked by a specific type of I-Frame called IDR (Instantaneous Decoder Refresh). IDRs tell the decoder that no frame that comes after it will be referencing any frames before it which is important when a decoder is adapting between encodes in an ABR ladder. Simple I-Frames don't force the decoder to start a new GOP and thus provide an additional flexibility for the encoder when there is a substantial change in the "scene" mid-GOP.

We recommend that you set this option to enable, so the encoder can search further into the differences and come up with a better strategy for placing I-Frames mid-GOP.

16. Min I-Interval

This setting tightly links to the previous scene change detection setting. When you specify a minimum I-interval, the encoder will drop the next GOP cadence I-Frame if it occurs within the number of frames specified after a Scene Change Detection I-Frame. This avoids two I-Frames being placed close to one another. If a Scene Change occurred, the I-Frame placed in the beginning of that new scene will probably already carry the information we need to continue encoding the next GOP, so another I-Frame in the vicinity would be redundant and a waste of bits.

Increasing the Min I-Interval can potentially improve the overall video quality, as more P-Frames and B-Frames can be used which are generally more efficient in terms of bitrate. However, this may come at the cost of slightly higher bitrates to maintain the same visual quality.

For live streaming and low-latency scenarios, a Min I-Interval of 1 or 2 seconds is commonly recommended.

Codec Details

17. Profile and Level

Both Profile and Level specify the tools the encoder is allowed to use when compressing video and they should be set according to target player capabilities. If the encoder uses a certain tool in its compressing strategy that a player can't use, it’s likely the stream won't play. Most players also check for the profile and level tags in the video stream and decide whether they can play it or not.

The common **MPEG profiles** that are used for AVC compression are:

a. Baseline Profile: Designed for low-complexity applications and devices with limited processing power. Baseline supports features like variable-length coding and prediction.
b. Main Profile: The most widely used profile, offering a balance of compression efficiency and decoder complexity. Main adds support for features like B-frames, CABAC and interlaced coding.
c. High Profile: Provides enhanced compression efficiency and supports additional coding tools, and is often used for high-definition and professional video applications.
d. High 10 Profile: Adds support for 10-bit color depth, providing better color fidelity and dynamic range.
e. High 422 Profile: Adds support for 4:2:2 Chroma Subsampling, providing better color fidelity and dynamic range.
f. High 422 10bit Profile: Combination of High 10 and High 422 Profile capabilities.

For HEVC there is similar set of profiles, but 4:2:2 chroma subsampling is not supported on MediaLive at this time for HEVC and Baseline profile is not part of the HEVC specification.

Levels: Levels in MPEG compression define the constraints on various parameters, such as the maximum video resolution, frame rate, and bitrate. These constraints help ensure the video stream can be properly decoded and played back on a target device or platform.

The combination of a specific profile and level defines the overall capabilities and constraints of an MPEG video stream, ensuring compatibility and appropriate resource utilization on the target devices.

When working with MPEG video in the context of MediaLive, it's important to consider the profile and level requirements of your target audience and devices to ensure the video content is properly encoded and can be played back seamlessly. To improve video quality we recommend using High profile (if compatibility with end devices is there). For level configuration, we recommend using the auto mode where MediaLive will select the right level automatically based on encoding settings.

18. Entropy Encoding

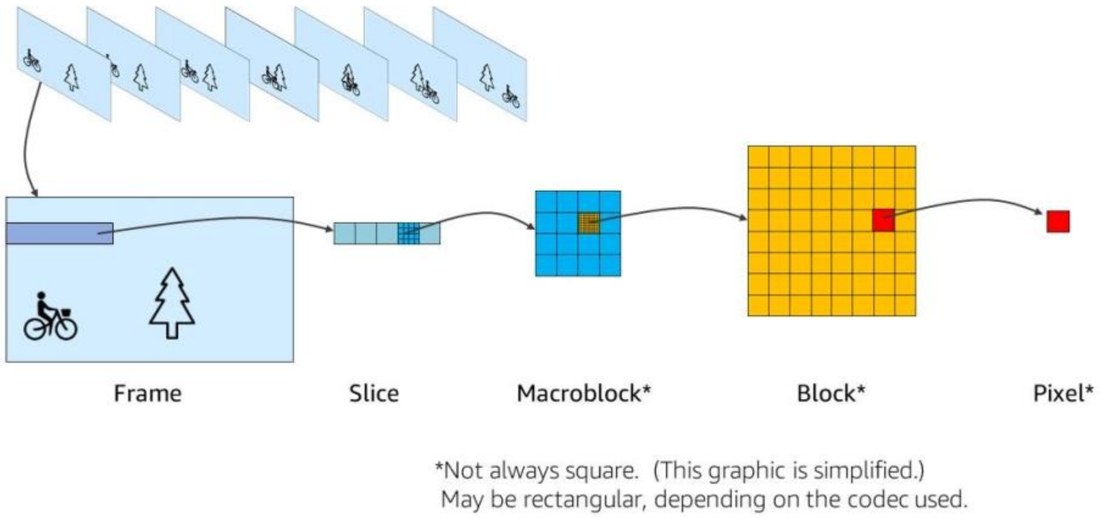
Context-adaptive binary arithmetic coding *(***CABAC)** and **Context-adaptive variable-length coding (CAVLC)** is the two entropy encoding options available for AVC. **CABAC** requires much more processing on the decoder side, so it was left out of the Baseline profile in AVC. On the other hand, it is the only entropy available in all HEVC profiles.

Entropy Encoding is a complex compression technique on which the encoder assigns a variable length codeword for symbols on the input data based on the probability that such symbols occur. The more probable it is for the symbol to occur, the smaller the codeword it receives. Conversely the less probable it is to occur, the bigger the codeword. The symbols are then replaced by their codewords on the output; since the most repeating symbols got the smaller codes, the resulting data is smaller in bits than the input. In layman's terms, the entropy algorithm tries to identify reoccurring patterns in order to represent them with codes that are smaller than the patterns themselves.

For example, the word "video" appears many times throughout this document. If we were to replace it with "$", for example, the text would become much shorter than it currently is because "$" takes less bits than "video".

CABAC should be enabled every time it is available.

19. Slices


Frame
Slice
Macroblock*
Block*
Pixel*
*Not always square. (This graphic is simplified.)
May be rectangular, depending on the codec used.

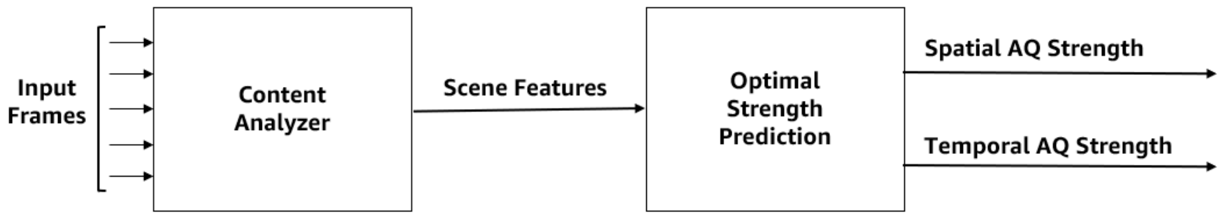
A slice is a spatially distinct region of a frame that is encoded separately from any other region in the same frame. The number of slices per picture must be less than or equal to the number of macroblock rows for progressive pictures, and less than or equal to half the number of macroblock rows for interlaced pictures. This field is optional; when no value is specified the encoder will choose the number of slices based on encode resolution. Our recommendation is to leave it blank.

20. Adaptive Quantization

Adaptive Quantization (AQ) can be thought of as rate control inside a picture.

The Adaptive Quantization parameter selects the strength applied to the different quantization modes explained below.

In Auto Mode, we use our own internal AI models to set the strength for each of the modes independently based on each frame's content.



21. Spatial Adaptive Quantization

Spatial AQ tries to move bits from areas of the frame (blocks) that are classified as visually less important to more important areas. The idea is that certain blocks can sustain more distortion with no noticeable visual degradation (e.g, complex textured blocks) while on other blocks, any small distortion will be noticeable (e.g, smooth textured blocks). Keep in mind that Spatial AQ isn't content aware. Therefore, the blocks the encoder chooses to apply less bits to could belong to an object that was drawing the viewer's attention at that moment. In other words, removing bits from them at that specific scene may not be ideal.

In general, you should choose a Low AQ for any homogenous content, such as gaming and cartoon and choose High or Higher for content with a wider variety of textures.

22. Temporal Adaptive Quantization

Temporal AQ tries to spare bits for areas within the frame that are complex in nature and are affected by motion. For example, scrolling text tickers on news cast and scoreboards on sports matches. The text characters themselves are highly complex with sharp edges and they move across the screen. Temporal AQ will try to improve the readability by detecting motion of such complex entities, and remove possible trailing artifacts.

Temporal AQ should almost always be enabled, but keep in mind that, like Spatial AQ, it might remove bits from relevant areas of the frame. For example, if enabling Temporal AQ on sports matches would make the athletes faces and movements less clear, it could possibly improve things by being turned off. On the other hand, for a news program not only is the news presenter moving less and the video has less complexity, but also the viewers might be reading the text scrolling at the bottom of the screen, so Temporal AQ would help here.

23. Flicker Adaptive Quantization

The reason why flicker (or I-Frame pop) occurs in encoded video is because the encoder copies certain macroblocks from one frame to other with the intention to save bits. When a new I-Frame is decoded and displayed, it refreshes the whole frame so the macroblocks are updated with new information and the abrupt change is caught by our eyes. Even though the image under these macroblocks changed very little from one frame to the other, they have, in fact, changed. When Flicker AQ is enabled, the encoder forces these macroblocks that were being copied many times over to be refreshed a short number of frames before the next I-Frame comes. This means these macroblocks are refreshed just enough to fool our eyes, while the encoder can still copy them many times over to save bits.

We recommend enabling this feature only when I-Frame pulsing is noticeable. Otherwise, you should let the encoder run its macroblock optimizations freely.

24.Softness

The "Softness" setting in AWS Elemental MediaLive refers to a video preprocessing filter that can be applied to the input video before encoding. It can be used to reduce the sharpness or edge definition of the input video. It can be useful for reducing noise or aliasing artifacts that may be present in the input video. Softening the video can also create a more aesthetically pleasing look, depending on the content and the desired visual style. We don’t recommend to use it for pristine sources.

Softness Setting Values:

The Softness setting in MediaLive can be adjusted on a scale from 0 to 100. A value of 0 means no softening is applied, and the video will retain its original sharpness. Higher values (up to 100) increase the amount of softening applied to the video.

Use Cases for the Softness Setting:

1. Reducing noise or graininess in low-quality input video sources.
2. Softening the appearance of high-definition video to match a specific artistic style or visual aesthetic.
3. Improving the quality of video captured from sources with poor optics or lens distortion.

Considerations for Softness:
Applying too much softness can lead to a blurry or overly smooth appearance, which may not be desirable. The optimal Softness setting will depend on the specific input video characteristics and the desired output quality. Overall, we recommend you experiment with different Softness values and evaluate the results to find the best balance for your use case.

In summary, the Softness setting in AWS Elemental MediaLive allows you to apply a video preprocessing filter to reduce the sharpness and edge definition of the input video, which can be useful for improving visual quality or achieving a specific aesthetic.

25. Look Ahead Rate Control

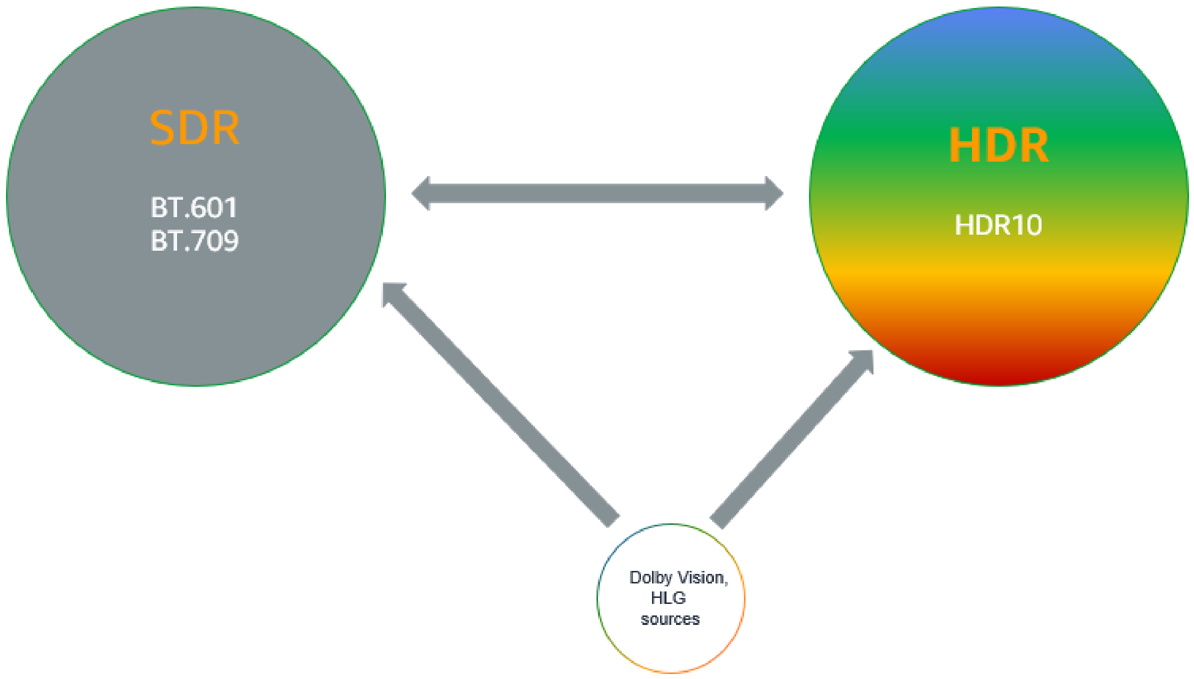
Look Ahead Rate Control analyzes a few seconds of upcoming frames in the video stream. Based on this analysis, it can predict the complexity of the upcoming frames and adjust the encoding budget per frame to maintain the target video quality. This predictive approach helps prevent quality fluctuations and ensures a more stable and consistent viewing experience.

Look Ahead Rate Control value of Low can decrease latency, while High can produce better quality for certain content at the expense of increased latency.

Color Space

26. Color Space Conversion

As the name indicates, this option allows you to output video in a color space that is different from the input. This is commonly used to translate from one High Dynamic Range (HDR) standard to another or to convert HDR to SDR and vice-versa. For example, you could feed the encoder with an HLG video and configure two outputs: in one of them you leave color correction off (pass through), so the output color space is the same as the input and in the other output you enable color correction and set it to HDR10 or SDR. In this way, with a single HDR input, you can obtain streams compatible with other HDR standards. HDR conversion is available for the HEVC codec only.



In addition, you can configure a channel to use a **3D** lookup table (**3D LUT**) file for conversion. You provide a list of 3D LUT files and each file contains color mapping information for a specific source/output combination.

3D LUTs are typically used in professional video production and post-production workflows, where precise color grading and control are essential. The default color conversion in MediaLive is suitable for more general video processing and streaming scenarios, where advanced color manipulation may not be as critical.

In summary, the use of a 3D LUT in MediaLive allows for more sophisticated and tailored color transformations at the cost of increased complexity and setup requirements. The default color conversion in MediaLive is a simpler, more generic approach, which may be sufficient for many video streaming use cases.

Additional Encoding Settings

27. Quality Level

You should leave this set to STANDARD_QUALITY, or choose a different value which might result in additional costs to run the channel. ENHANCED_QUALITY produces slightly better video quality without an increase in the bitrate, but affects video only when the Rate control mode is QVBR or CBR. If this channel is in a MediaLive multiplex, the value must be ENHANCED_QUALITY. STANDARD_QUALITY is valid for any Rate control mode.

28. Filter settings

The **temporal** filter uses frequency and contrast masking with additional motion masking to improve encoding efficiency. Also known as MCTF (Motion Compensated Temporal Filter), this filter is well stablished and widely used to reduce imperceptible details and help the encoder lower the bitrates. When the strength is set in the range from 1 to 4, the encoder will perform a lossless frequency filtering that should be imperceptible to the human eye. When the strength is set to higher values, the encoder performs in lossy mode where loss of some details and sharpness may occur. You should use the higher strength levels only in extreme cases such as for creating extremely low bitrate versions of content.

Considerations:

- Applying too much temporal filtering can lead to a "smeared" or overly blurred appearance, especially for fast-moving content.
- The optimal Temporal Filter settings depend on the specific characteristics of the input video and the desired output quality.
- We recommended you experiment with different Temporal Filter settings and evaluate the results to find the best balance for your use case.

In summary, the Temporal Filter in MediaLive is a powerful tool for reducing video noise and artifacts over time, helping to improve the overall quality and stability of the output video stream.

Bandwidth Reduction Filter (BRF) reduces encoded video bitrates by an average of 8-10%, depending on content, while maintaining quality and saving on storage and CDN costs. At its core, the BRF leverages human vision system models to reduce imperceptible signals and temporal noise from video input, which generally originates from camera sensors.

The BRF is a custom filter designed to run inside the video encoder loop (in-loop), allowing access to privileged data about each frame. This data includes the frame type (I, B, or P frame), detailed information about skip blocks and intra blocks, and the quantization parameter (QP) of the frame.

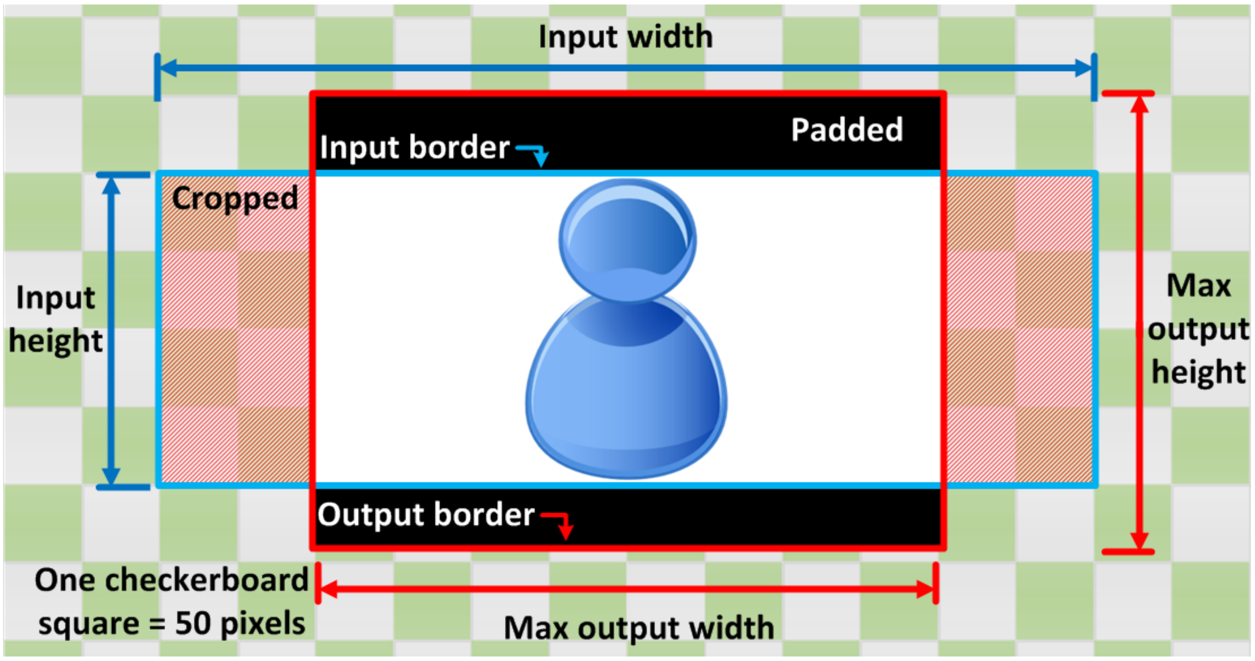
The in-loop filter provides efficiencies as it processes frames based on specific complexities and requirements. It uses internal compression statistics to dynamically adjust strength which is different from a pre-filter, which processes all frames uniformly before they enter the encoder. This functionality allows BRF to be perceptually motivated as a non-destructive filter.

The recommendation is to enable BRF by default (available for HEVC and Enhanced AVC).

Scaling Settings

29. Scaling Behavior

With the **Default** configuration, the encoder will keep the original video's aspect ratio, even if the specified aspect ratio in the Resolution field is different. To maintain the aspect ratio, the encoder inserts black bars at the top and bottom (letterbox) or at the sides (pillarbox). With **Stretch To Output**, the encoder will use all the pixels specified in the resolution field (the whole image area, so no added black bars) even if it has to distort the original picture.



30. Scaling Sharpness

This sharpness option is used exclusively when scaling into different resolutions, with a value of 0 for the softest setting and 100 for the sharpest. If your output resolution is the same as the input, this option does not do anything. Typically, when scaling down it is safe to set the value to 100 for most content in order to preserve as much contour detail as possible. When scaling up, higher values can cause aliasing, so we recommend a value of 50 or 25. Keep in mind that this filter is applied by the scaler before any pre-processor, so any Noise Reduction filter is applied after this one and adds to the effect.

A setting of 50 is recommended for most content.

Input settings

31. Input Video Filter

The **Denoise** filter tries to remove ringing artifacts (usually around edges; and when they are in motion), or what some would call “mosquito noise”. This effect is most noticeable if a source is already heavily compressed. The encoder uses heuristics to search for edges and then it filters around them. Depending on the quantization of the source frame (or the quantization that the previous encoder used for that frame) we apply more or less filtering.

The **Deblocking** filter also use heuristics to find possible blocking artifacts, which are “squares” in a frame that stand out from the rest.

Mosquito noise and blocking artifacts are both typical compression issues and these filters are meant to help reduce them in heavily compressed sources. If you are using a high-quality source, there is no need to use these input filters. If your source is heavily compressed, then the strength of these filters will depend on how sensitive you are to losing detail (creating a softer image) or how sensible they are to reduce artifacts. There is no optimal filter strength to recommend since these depend on personal taste.

Conclusion:

We hope this document provides insights into the encoding parameters that can be configured for AWS Elemental MediaLive, and how altering them impacts the output video.

For more information, please refer to the MediaLive documentation.