INTERNATIONAL STANDARD

ISO/IEC 23001-11

First edition
2015-07-15

Information technology — MPEG systems technologies —

Part 11: Energy-efficient media consumption (green metadata)

Technologies de l'information — Technologies des systèmes MPEG —

Partie 11: Consommation des supports éconergétiques (métadonnées vertes)

Reference number
ISO/IEC 23001-11:2015(E)

ISO copyright office
Ch. de Blandonnet 8 • CP 401
CH-1214 Vernier, Geneva, Switzerland
Tel. +41 22 749 01 11
Fax +41 22 749 09 47
copyright@iso.org
www.iso.org

Contents

Page

Introduction

This part of ISO/IEC 23001 specifies the metadata (Green Metadata) that facilitates reduction of energy usage during media consumption as follows:

— the format of the metadata that enables reduced decoder power consumption;

— the format of the metadata that enables reduced display power consumption;

— the format of the metadata that enables media selection for joint decoder and display power reduction;

— the format of the metadata that enables quality recovery after low-power encoding.

This metadata facilitates reduced energy usage during media consumption without any degradation in the Quality of Experience (QoE). However, it is also possible to use this metadata to get larger energy savings, but at the expense of some QoE degradation.

Information technology — MPEG systems technologies —

Part 11:
Energy-efficient media consumption (green metadata)

1 Scope

This part of ISO/IEC 23001 specifies metadata for energy-efficient decoding, encoding, presentation, and selection of media.

The metadata for energy-efficient decoding specifies two sets of information: Complexity Metrics (CM) metadata and Decoding Operation Reduction Request (DOR-Req) metadata. A decoder uses CM metadata to vary operating frequency and thus reduce decoder power consumption. In a point-to-point video conferencing application, the remote encoder uses the DOR-Req metadata to modify the decoding complexity of the bitstream and thus reduce local decoder power consumption.

The metadata for energy-efficient encoding specifies a quality metric that is used by a decoder to reduce the quality loss from low-power encoding.

The metadata for energy-efficient presentation specifies RGB-component statistics and quality levels. A presentation subsystem uses this metadata to reduce power by adjusting display parameters, based on the statistics, to provide a desired quality level from those provided in the metadata.

The metadata for energy-efficient media selection specifies Decoder Operation Reduction Ratios (DOR-Ratios), RGB-component statistics and quality levels. The client in an adaptive streaming session uses this metadata to determine decoder and display power-saving characteristics of available video Representations and to select the Representation with the optimal quality for a given power-saving.

2 Normative references

The following documents, in whole or in part, are normatively referenced in this document and are indispensable for its application. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 13818-1:2013, *Information technology — Generic coding of moving pictures and associated audio information — Part 1: Systems*

ISO/IEC 14496-10, *Information technology — Coding of audio-visual objects — Part 10: Advanced Video Coding*

ISO/IEC 14496-12, *Information technology — Coding of audio-visual objects — Part 12: ISO base media file format*

ISO/IEC 23001-10, *Information technology — MPEG systems technologies — Part 10: Carriage of Timed Metadata Metrics of Media in ISO Base Media File*

ISO/IEC 23009-1, *Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 1: Media presentation description and segment formats*

ISO/IEC 23009-1:2015/Amd 2:2015, *Spatial relationship description, generalized URL parameters and other extensions*

ISO/IEC/TR 23009-3:2015, *Information technology — Dynamic adaptive streaming over HTTP (DASH) — Part 3: Implementation guidelines*

3 Terms, definitions, symbols, abbreviated terms and conventions

For the purposes of this document, the following terms and definitions apply.

3.1 Terms and definitions

3.1.1
Adaptation Set
using the terms and definitions in ISO/IEC 23009-1, a set of interchangeable encoded versions of one or several media content components

3.1.2
Alpha-Point Deblocking Instance
APDI
using the notation, terms, and definitions in ISO/IEC 14496-10, a single filtering operation that produces either a single, filtered output p'_0 or a single, filtered output q'_0, where p'_0 and q'_0 are filtered samples across a 4x4 *block* edge

3.1.3
bitstream
using the terms and definitions in ISO/IEC 14496-10, a sequence of bits that forms the representation of coded pictures and associated data forming one or more coded video sequences

3.1.4
block
using the terms and definitions in ISO/IEC 14496-10, an MxN (M-column by N-row) array of samples or an MxN array of transform coefficients

3.1.5
byte
using the terms and definitions in ISO/IEC 14496-10, a sequence of 8 bits, written and read with the most significant bit on the left and the least significant bit on the right

3.1.6
chroma
using the terms and definitions in ISO/IEC 14496-10, an adjective specifying that a sample array or single sample is representing one of the two colour difference signals relating to the primary colours

3.1.7
chroma_format_idc
using the notation, terms and definitions in ISO/IEC 14496-10, specifies the chroma sampling relative to the luma sampling

3.1.8
decoded picture
using the terms and definitions in ISO/IEC 14496-10, a picture derived by decoding a coded picture

3.1.9
decoder
using the terms and definitions in ISO/IEC 14496-10, an embodiment of a decoding process

3.1.10
display process
using the terms and definitions in ISO/IEC 14496-10, a process that takes, as its input, the cropped decoded pictures that are the output of the decoding process

3.1.11
encoder
using the terms and definitions in ISO/IEC 14496-10, an embodiment of an encoding process

3.1.12
frame
using the terms and definitions in ISO/IEC 14496-10, an array of luma samples in monochrome format or an array of luma samples and two corresponding arrays of chroma samples in 4:2:0, 4:2:2 , and 4:4:4 colour format

3.1.13
informative
term used to refer to content provided in this Recommendation | International Standard that is not an integral part of this Recommendation | International Standard

3.1.14
intra coding
using the terms and definitions in ISO/IEC 14496-10, coding of a block, macroblock, slice or picture that uses intra prediction

3.1.15
luma
using the terms and definitions in ISO/IEC 14496-10, an adjective specifying that a sample array or single sample is representing the monochrome signal relating to the primary colours

3.1.16
macroblock
using the terms and definitions in ISO/IEC 14496-10, a 16x16 block of luma samples and two corresponding blocks of chroma samples of a picture that has three sample arrays, or a 16x16 block of samples of a monochrome picture or a picture that is coded using three separate colour planes

3.1.17
Media Presentation Description
MPD
using the terms and definitions in ISO/IEC 23009-1, a formalized description for a Media Presentation for the purpose of providing a streaming service

3.1.18
No-Quality-Loss Operating Point
NQLOP
metadata-enabled operating point associated with the largest display-power reduction that can be achieved without any quality loss (infinite PSNR)

3.1.19
non-zero macroblock
macroblock (3.1.16) containing at least one non-zero sample

3.1.20
note
term that is used to prefix *informative* (3.1.13) remarks (used exclusively in an informative context)

3.1.21
peak signal
maximum permissible *RGB component* (3.1.31) in a reconstructed frame

Note 1 to entry: For 8-bit video, peak signal is 255.

3.1.22
period
interval over which complexity-metrics metadata are applicable

3.1.23
Period
using the terms and definitions in ISO/IEC 23009-1, an interval of the Media Presentation, where a contiguous sequence of all Periods constitutes the Media Presentation

3.1.24
PicSizeInMbs
using the notation, terms and definitions in ISO/IEC 14496-10, a variable that is derived as the product of PicWidthInMbs and PicHeightInMbs

3.1.25
picture
using the terms and definitions in ISO/IEC 14496-10, a collective term for a field or a frame

3.1.26
pixel
smallest addressable element in an all-points addressable display device

3.1.27
prediction
using the terms and definitions in ISO/IEC 14496-10, an embodiment of the prediction process

3.1.28
reconstructed frames
frames obtained after applying RGB colour-space conversion and cropping to the specific *decoded picture* (3.1.8) or *pictures* (3.1.25) for which display power-reduction metadata are applicable

3.1.29
Representation
using the terms and definitions in ISO/IEC 23009-1, a collection and encapsulation of one or more media streams in a delivery format and associated with descriptive metadata

3.1.30
RGB colour space
colour space based on the red, green, and blue primaries

3.1.31
RGB component
single sample representing one of the three primary colours of the *RGB colour space* (3.1.30)

3.1.32
Segment
using the terms and definitions in ISO/IEC 23009-1, a unit of data associated with an HTTP-URL and optionally a byte range that are specified by an MPD

3.1.33
separate_colour_plane_flag
using the notation, terms, and definitions in ISO/IEC 14496-10, a flag that, when set, specifies that the three colour components of the 4:4:4 chroma format are coded separately

3.1.34
shall
term used to express mandatory requirements for conformance to this Recommendation | International Standard

3.1.35
should
term used to refer to behaviour of an implementation that is encouraged to be followed under anticipated ordinary circumstances, but is not a mandatory requirement for conformance to this Recommendation | International Standard

3.1.36
Six-Tap Filtering
STF
indicates a single application of the 6-tap filter, defined in ISO/IEC 14496-10, to generate a single filtered sample

3.1.37
source
using the terms and definitions in ISO/IEC 14496-10, a term used to describe some of the video material or some of its attributes before encoding

3.2 Symbols and abbreviated terms

For the purpose of this document, the symbols and abbreviated terms given in the following apply:

APDI	Alpha-Point Deblocking Instance
ASIC	Application Specific Integrated Circuit
AVC	Advanced Video Coding
CM	Complexity Metric
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central processing Unit
DASH	Dynamic Adaptive Streaming over HTTP
DOR-Ratio	Decoding Operation Reduction Ratio
DOR-Req	Decoding Operation Reduction Request
DVFS	Dynamic Voltage Frequency Scaling
FS	Fresh Start
GP	Good Picture
MPD	Media Presentation Description
MSD	Mean Square Difference
MV	Motion Vector
NQLOP	No-Quality-Loss Operating Point
PSNR	Peak Signal to Noise Ratio
QoE	Quality of Experience
RBLL	Remaining Battery Life Level
RGB	Red, Green, Blue
SEI	Supplemental Enhancement Information
SP	Start Picture
STF	Six-Tap Filtering
XSD	Cross-Segment Decoding

3.3 Conventions

3.3.1 Arithmetic operators

+	Addition

-	Subtraction (as a two-argument operator) or negation (as a unary prefix operator)
*	Multiplication
x^y	Exponentiation
x/y	Division where no truncation or rounding is intended
$\frac{x}{y}$	Division where no truncation or rounding is intended
$\sum_{i=x}^{y} f(i)$	Summation of $f(i)$ with i taking all integer values from x up to and including y

3.3.2 Mathematical functions

Mathematical functions in this Technical Specification are defined as follows:

$$\mathrm{Abs}(x) = \begin{cases} -x, & x < 0 \\ x, & x \geq 0 \end{cases} \quad (3\text{-}1)$$

$$\mathrm{Clip}(x) = \begin{cases} x, & x < 256 \\ 255, & \text{otherwise} \end{cases} \quad (3\text{-}2)$$

Floor(x) is the greatest integer less than or equal to x (3-3)

Log10(x) returns the base-10 logarithm of x (3-4)

Round(x) = Sign(x) * Floor(Abs(x) + 0.5) (3-5)

$$\mathrm{Sign}(x) = \begin{cases} -1, & x < 0 \\ 1, & x \leq 0 \end{cases} \quad (3\text{-}6)$$

4 Functional architecture (Informative)

This clause is informative and placed here to provide context.

4.1 Description of the functional architecture

Figure 1 shows the functional architecture utilizing Green Metadata in this Technical Specification. The media pre-processor is applied to analyse and to filter the content source and a video encoder is used to encode the content to a bitstream for delivery. The bitstream is delivered to the receiver and decoded by a video decoder with the output rendered on a presentation subsystem that implements a display process.

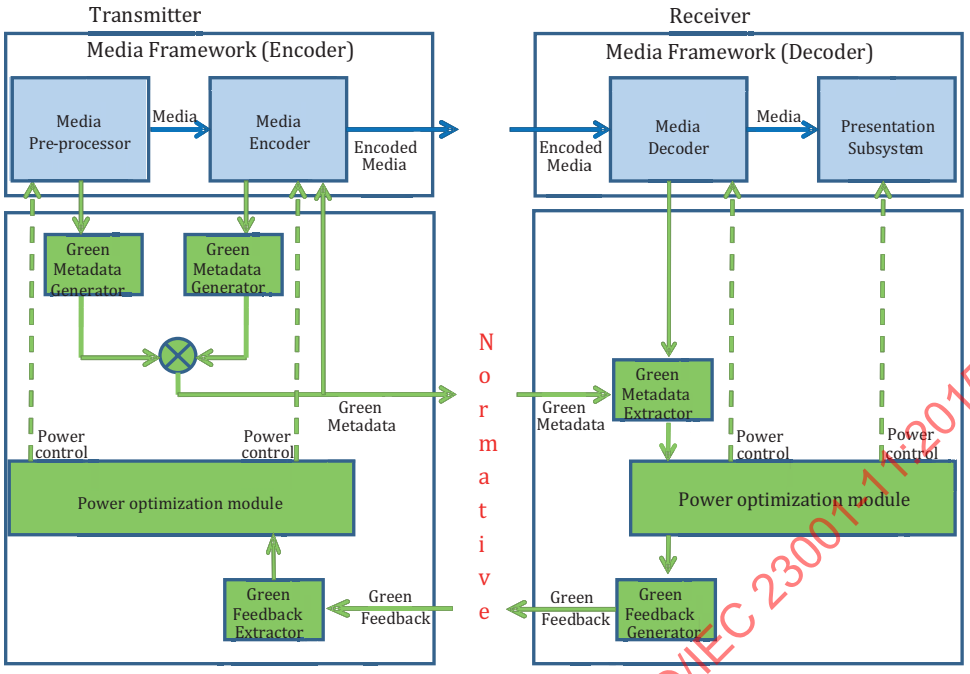

Figure 1 — Functional architecture

The Green Metadata is extracted from either the media encoder or the media pre-processor. In both cases, the Green Metadata is multiplexed or encapsulated in the conformant bitstream. Such Green Metadata is used at the receiver to reduce the power consumption for video decoding and presentation. The bitstream will be packetized and delivered to the receiver for decoding and presentation. At the receiver, the metadata extractor processes the packets and sends the Green Metadata to a power optimization module for efficient power control. For instance, the power optimization module interprets the Green Metadata and then applies appropriate operations to reduce the video decoder's power consumption when decoding the video and also to reduce the presentation subsystem's power consumption when rendering the video. In addition, the power-optimization module could collect receiver information, such as remaining battery capacity, and send it to the transmitter as green feedback to adapt the encoder operations for power-consumption reduction.

The normative aspect of this document is limited to the Green Metadata and Green Feedback in Figure 1.

4.2 Definition of components in the functional architecture

Green Metadata generator

— Generates metadata from either the video encoder or the content pre-processor.

Green Metadata extractor

— Interprets the bitstream syntax information and sends it to the power optimization module in the receiver.

Green feedback generator

— Generates feedback information for the transmitter.

— Communicates with the transmitter through a feedback channel, if available, for energy-efficient processing.

Green feedback extractor

— Receives the feedback from the receiver and sends it to the power optimization module in the transmitter.

Power optimization module in the transmitter

— Collects platform statistics such as the remaining battery capacity of the device in which the transmitter resides.

— Controls the operation of the Green Metadata generator, video encoder and content pre-processor.

— Processes green feedback.

Power optimization module in the receiver

— Processes the green-metadata information and applies appropriate operations for power-consumption control.

— Collects platform statistics such as remaining battery capacity of the device in which the receiver resides.

— Sends requests to Green feedback generator.

5 Decoder power reduction

5.1 General

Energy-efficient decoding is achieved with two types of metadata: Complexity Metrics (CMs) metadata and Decoding Operation Reduction Request (DOR-Req) metadata. A decoder may use CMs metadata to vary operating frequency and thus reduce decoder power consumption. In a point-to-point video conferencing application, the remote encoder may use the DOR-Req metadata to modify the decoding complexity of the bitstream and thus reduce local decoder power consumption.

5.2 Complexity metrics for decoder-power reduction

5.2.1 General

With respect to the functional architecture in Figure 1, the green-metadata generator provides CMs that indicate the picture-decoding complexity of an AVC bitstream to the decoder.

5.2.2 Syntax

The syntax for the CMs is as follows:

	Size (bits)	Descriptor
period_type	8	unsigned integer
if (period_type = = 2) {		
num_seconds	16	unsigned integer
}		
else if (period_type = = 3) {		
num_pictures	16	unsigned integer
}		
percent_non_zero_macroblocks	8	unsigned integer
percent_intra_coded_macroblocks	8	unsigned integer
percent_six_tap_filterings	8	unsigned integer
percent_alpha_point_deblocking_instances	8	unsigned integer

5.2.3 Signalling

SEI messages can be used to signal Green Metadata in an AVC stream. The Green Metadata SEI message payload type is specified in ISO/IEC 14496-10:2014/Amd 2. The complete syntax of the Green Metadata SEI message payload is specified in Annex A.

The message containing the CMs is transmitted at the start of an upcoming period. The next message containing CMs will be transmitted at the start of the next upcoming period. Therefore, when the upcoming period is a picture or the interval up to the next I-slice, a message will be transmitted for each picture or interval, respectively. However, when the upcoming period is a specified time interval or a specified number of pictures, the associated message will be transmitted with the first picture in the time interval or with the first picture in the specified number of pictures.

5.2.4 Semantics

The semantics of various terms are defined below.

period_type – specifies the type of upcoming period over which the four complexity metrics are applicable and is defined in the following table.

Value	Description
0x00	complexity metrics are applicable to a single picture
0x01	complexity metrics are applicable to all pictures in decoding order, up to (but not including) the picture containing the next I slice
0x02	complexity metrics are applicable over a specified time interval in seconds
0x03	complexity metrics are applicable over a specified number of pictures counted in decoding order
0x04–0xFF	user-defined

num_seconds – when period_type is 2, num_seconds indicates the number of seconds over which the complexity metrics are applicable.

num_pictures – when period_type is 3, num_pictures specifies the number of pictures, counted in decoding order, over which the complexity metrics are applicable.

num_pics_in_period – specifies the number of pictures in the specified period. When period_type is 0, then num_pics_in_period is 1. When period_type is 1, then num_pics_in_period is determined by counting the pictures in decoding order up to (but not including) the one containing the next I slice. When period_type is 2, then num_pics_in_period is determined from the frame rate. When period_type is 3, then num_pics_in_period is equal to num_pictures.

total_num_macroblocks_pic(i) – set to the value of the AVC variable picSizeInMbs for the i^{th} picture within the specified period, where 1 <= i <= num_pics_in_period.

total_num_macroblocks_in_period – indicates the total number of macroblocks that are coded in the specified period. Determined by the following computation:

$$\sum_{i=1}^{\text{num\_pics\_in\_period}} \text{total\_num\_macroblocks\_pic}(i) \tag{5-1}$$

num_intra_coded_macroblocks – indicates the number of intra-coded macroblocks in the specified period.

percent_intra_coded_macroblocks – indicates the percentage of intra-coded macroblocks in the specified period and is defined as follows:

$$\text{percent\_intra\_coded\_macroblocks}=\text{Floor}\left(\frac{\text{num\_intra\_coded\_macroblocks}}{\text{total\_num\_macroblocks\_in\_period}}*255\right) \tag{5-2}$$

num_non_zero_macroblocks – indicates the number of non-zero macroblocks in the specified period.

percent_non_zero_macroblocks – indicates the percentage of non-zero macroblocks in the specified period and is defined as follows:

$$\text{percent\_non\_zero\_macroblokcs}=\text{Floor}\left(\frac{\text{num\_non\_zero\_macroblocks}}{\text{total\_num\_macroblocks\_in\_period}}*255\right) \tag{5-3}$$

num_six_tap_filterings – indicates the number of Six-Tap Filterings (STFs), as defined in ISO/IEC 14496-10, within the specified period.

max_num_six_tap_filterings_pic(i) – indicates the maximum number of STFs that could occur in the i^{th} picture within the specified period, where 1 <= i <= num_pics_in_period. Set to the value (1664 * picSizeInMbs), where picSizeInMbs is the value of the corresponding AVC variable for the i^{th} picture.

max_num_six_tap_filterings_in_period – indicates the maximum number of STFs that could occur within the specified period. Determined by the following computation:

$$\sum_{i=1}^{\text{num\_pics\_in\_period}} \text{max\_num\_six\_tap\_filterings\_pic}(i) \tag{5-4}$$

percent_six_tap_filterings – indicates the percentage of STFs in the specified period and is defined as follows:

$$\text{percent\_six\_tap\_filterings}=\text{Floor}\left(\frac{\text{num\_six\_tap\_filterings}}{\text{max\_num\_six\_tap\_filterings\_in\_period}}*255\right) \tag{5-5}$$

num_alpha_point_deblocking_instances – indicates the number of Alpha-Point Deblocking Instances (APDIs) in the specified period. Using the notation in ISO/IEC 14496-10, this is equivalent to the total number of filtering operations applied to produce filtered samples of the type p'_0 or q'_0, in the specified period.

max_num_alpha_point_deblocking_instances_pic(i) – indicates the maximum number of APDIs that could occur in the i^{th} picture within the specified period, where 1 <= i <= num_pics_in_period. Set as follows:

$$\text{max\_num\_alpha\_point\_deblocking\_instances\_pic(i)} = 128 * \text{chroma\_format\_multiplier} * \text{PicSizeInMbs} \quad (5\text{-}6)$$

where chroma_format_multiplier depends on the AVC variables separate_colour_plane_flag and chroma_format_idc as shown in the following table.

chroma_format_multiplier	separate_colour_plane_flag	chroma_format_idc	Comment
1	1	any value	separate colour plane
1	0	0	monochrome
1.5	0	1	4:2:0 sampling
2	0	2	4:2:2 sampling
3	0	3	4:4:4 sampling

max_num_alpha_point_deblocking_instances_in_period – indicates the maximum number of APDIs that could occur within the specified period. Determined by the following computation:

$$\sum_{i=1}^{\text{num\_pics\_in\_period}} \text{max\_num\_alpha\_point\_deblocking\_instances\_pic}(i) \quad (5\text{-}7)$$

percent_alpha_point_deblocking_instances – indicates the percentage of APDIs in the specified period and is defined as follows:

$$\text{percent\_alpha\_point\_deblocking\_instances} = \text{Floor}\left(\frac{\text{num\_alpha\_point\_deblocking\_instances}}{\text{max\_num\_alpha\_point\_deblocking\_instances}} * 255\right) \quad (5\text{-}8)$$

5.3 Interactive signalling for remote decoder-power reduction

5.3.1 General

For point-to-point video conferencing, each device contains a transmitter and a receiver. A local device sends metadata that instructs the remote device to modify the decoding complexity of the bitstream and thus reduce local decoder-power consumption.

5.3.2 Syntax

The syntax is as follows:

	Size (bits)	Descriptor
dec_ops_reduction_req	8	signed integer

5.3.3 Signalling

The transmitter in each device sends a dec_ops_reduction_req (DOR-Req) message to the attention of the remote encoder. This message requests the remote encoder to adjust its encoding parameters so that ideally, when the local decoder decodes the bitstream, the power saving of the local decoder will match the power saving implied by the DOR-Req message.

5.3.4 Semantics

dec_ops_reduction_req – the requested percentage reduction of local decoding operations relative to the local decoding operations since the last dec_ops_reduction_req was sent to the transmitter, or since the start of the video session, if no earlier dec_ops_reduction_req was sent. The percentage will be expressed as a signed integer. A negative percentage means an increase of decoding operations. dec_ops_reduction_req is an integer in the interval [-100, 100].

6 Display power reduction using display adaptation

6.1 General

With respect to the functional architecture, Display Adaptation (DA) provides Green Metadata comprised of RGB-component statistics and quality indicators. The statistics are used to set display controls in the presentation subsystem so that desired quality levels and corresponding display power reductions are attained.

6.2 Syntax

6.2.1 Systems without a signalling mechanism from the receiver to the transmitter

The following message format is used to send metadata from the transmitter to the receiver:

	Size (bits)	Descriptor
num_constant_backlight_voltage_time_intervals	2	unsigned integer
num_max_variations	2	unsigned integer
num_quality_levels	4	unsigned integer
for (j = 0; j < num_max_variations; j++) {		
max_variation[j]	8	unsigned integer
}		
for (k = 0; k < num_constant_backlight_voltage_time_intervals;k++) {		
constant_backlight_voltage_time_interval[k]	16	unsigned integer
for (j = 0; j < num_max_variations; j++) {		
lower_bound[k][j]	8	unsigned integer
if **(lower_bound[k][j]** > 0) {		
upper_bound[k][j]	8	unsigned integer
}		
rgb_component_for_infinite_psnr[k][j]	8	unsigned integer
for (i = 1; i < = num_quality_levels; i++) {		
max_rgb_component[k][j][i]	8	unsigned integer
scaled_psnr_rgb[k][j][i]	8	unsigned integer
}		
}		
}		

6.2.2 Systems with a signalling mechanism from the receiver to the transmitter

The receiver first uses the following message format to signal information to the transmitter:

	Size (bits)	Descriptor
constant_backlight_voltage_time_interval	16	unsigned integer
max_variation	8	unsigned integer

The transmitter then uses the message format shown below to then signal metadata to the receiver:

	Size (bits)	Descriptor
num_quality_levels	4	unsigned integer
rgb_component_for_infinite_psnr	8	unsigned integer
for (i = 1; i < = num_quality_levels; i++) {		
max_rgb_component[i]	8	unsigned integer
scaled_psnr_rgb[i]	8	unsigned integer
}		

6.3 Signalling

6.3.1 Systems without a signalling mechanism from the receiver to the transmitter

Green Metadata can be carried as specified in ISO/IEC 13818-1:2013-Amd 3 or it can be carried in metadata tracks within the ISO Base Media File Format (ISO/IEC 14496-12), as specified in ISO/IEC 23001-10. Using the format in 6.2.1, the transmitter sends a message to the receiver. The DA metadata is applicable to the presentation subsystem until the next message containing DA metadata arrives.

6.3.2 Systems with a signalling mechanism from the receiver to the transmitter

Using the first message format described in 6.2.2, the receiver first signals constant_backlight_voltage_time_interval and max_variation to the transmitter. The transmitter then uses the second message format in 6.2.2 to send a message to the receiver. The DA metadata is applicable to the presentation subsystem until the next message containing DA metadata arrives.

6.4 Semantics

num_constant_backlight_voltage_time_intervals – the number of constant backlight/voltage time intervals for which metadata is provided in the bitstream.

constant_backlight_voltage_time_interval[k] – the minimum time interval, in milliseconds, that must elapse before the backlight can be updated after the last backlight update. This is the k^{th} minimum time interval for which metadata is provided in the bitstream, where 0 <= k < num_constant_backlight_voltage_time_intervals.

num_max_variations – the number of maximum variations for which metadata is provided in the bitstream.

max_variation[j] – the maximal change between backlight values of two successive frames relative to the backlight value of the earlier frame. The backlight value for a frame is the value of backlight_scaling_factor[k][j][i] for that frame. max_variation is in the range [0,001, 0,1] and is normalized to one byte by rounding after multiplying by 2 048. This is the j^{th} maximal backlight change for which metadata is provided in the bitstream, where 0 <= j < num_max_variations.

num_quality_levels – the number of quality levels that are enabled by the metadata, excluding the NQLOP.

max_rgb_component[k][j][i] – for the k[th] constant_backlight_voltage_time_interval, j[th] max_variation and i[th] quality level, the maximum RGB component (as defined in 3.1) that will be retained in the frames, where 1 <= i <= num_quality_levels. Note that max_rgb_component[k][j][0] = rgb_component_for_infinite_psnr[k][j].

scaled_frames[k][j][i] – for the k[th] constant_backlight_voltage_time_interval, j[th] max_variation and i[th] quality level, the frames obtained from the reconstructed frames by saturating to max_rgb_component[k][j][i] all RGB components that are greater than max_rgb_component[k][j][i], where 0 <= i <= num_quality_levels.

rgb_component_for_infinite_psnr[k][j] – for the k[th] constant_backlight_voltage_time_interval and j[th] max_variation, the largest RGB component (as defined in 3.1) in the reconstructed frames. Therefore, scaled_frames[k][j][0] are identical to the reconstructed frames. The rgb_component_for_infinite_psnr[k][j] defines a No-Quality-Loss Operating Point (NQLOP) and consequently scaled_frames[k][j][0] will have a PSNR of infinity relative to the reconstructed frames.

scaled_psnr_rgb[k][j][i] – the PSNR of scaled_frames[k][j][i] relative to the reconstructed frames. This PSNR is defined as follows:

scaled_psnr_rgb[k][j][i] =

$$\mathrm{Clip}\left(\mathrm{Round}\left(10\mathrm{Log}_{10}\left(\frac{\text{peak signal}^2 * \text{width} * \text{height} * N_{\text{colour}} * N_{\text{frames}}}{\sum_{n=1}^{N_{\text{frames}}}\sum_{c=1}^{N_{\text{colour}}}\sum_{l=X_S+1}^{\text{peaksignal}} N_{c,n}(l) * (l-X_S)^2}\right)\right)\right) \quad (6\text{-}1)$$

for 0 < I <= num_quality_levels,

where

width is the width of a video frame;

height is the height of a video frame;

N_{colour} is the number of colour channels. For RGB colourspace, $N_{\text{colour}} = 3$;

N_{frames} is the number of frames in the reconstructed frames;

$N_{c,n}(l)$ is the number of RGB components that are set to l in the n^{th} frame of colour-channel c in reconstructed frames;

X_S is max_rgb_component[k][j][i].

Note that scaled_psnr_rgb[k][j][0] is associated with the NQLOP. It is not transmitted, but understood to be mathematically infinite.

backlight_scaling_factor[k][j][i] – max_rgb_component[k][j][i]/peak signal, for the k[th] constant_backlight_voltage_time_interval, j[th] max_variation and i[th] quality level.

lower_bound[k][j] – if lower_bound[k][j] is greater than zero, then metadata for contrast enhancement is available at the lowest quality level, for the k[th] constant_backlight_voltage_time_interval and j[th] max_variation. If lower_bound[k][j] = 0, then contrast-enhancement metadata is unavailable.

upper_bound[k][j] – for the k[th] constant_backlight_voltage_time_interval and j[th] max_variation, if lower_bound[k][j] is greater than zero, then contrast enhancement is performed as follows: All RGB components of reconstructed frames that are less than or equal to lower_bound[k][j] are set to zero and all RGB components that are greater than or equal to upper_bound[k][j] are saturated to peak signal. The RGB components in the range (lower_bound[k][j], upper_bound[k][j]) are mapped linearly onto the range (0, peak signal).

7 Energy-efficient media selection

7.1 General

The Green Metadata specified in this clause can enable a client in an adaptive streaming session, such as DASH, to determine decoder and display power-saving characteristics of available video Representations and to select the Representation with the optimal quality for a given power-saving.

Two types of Green Metadata are defined as follows:

— decoder-power indication metadata gives the potential decoder power saving of each available Representation of a video Segment;

— display-power indication metadata gives the maximum potential display power saving of a video Segment for a specified number of quality levels. This metadata is computed without any constraint on the maximal backlight change between two successive frames and with no practical restriction on the minimum time interval between backlight updates. Therefore, using the semantics of 6.4, the metadata is produced with the assumptions that max_variation is mathematically infinite and that constant_backlight_voltage_time_interval is less than or equal to the interval between two successive frames.

7.2 Syntax

The decoder-power indication metadata is a pair of decoder operations reduction ratios:

	Size (bits)	Descriptor
dec_ops_reduction_ratio_from_max	8	unsigned integer
dec_ops_reduction_ratio_from_prev	16	signed integer

The display-power indication metadata contains a list of ms_num_quality_levels pairs, as shown below:

	Size (bits)	Descriptor
ms_num_quality_levels	4	unsigned integer
ms_rgb_component_for_infinite_psnr	8	unsigned integer
for (i = 1; i < = ms_num_quality_levels; i++) {		
ms_max_rgb_component[i]	8	unsigned integer
ms_scaled_psnr_rgb[i]	8	unsigned integer
}		

7.3 Signalling

Green Metadata can be carried in metadata tracks within the ISO Base Media File Format (ISO/IEC 14496-12). Such carriage is specified in ISO/IEC 23001-10.

In the context of DASH delivery, a specific Adaptation Set within the MPD can define the available Green Metadata Representations and their association to the available media Representations, using the signalling mechanisms specified in ISO/IEC 23009-1:2014/Amd 2 and ISO/IEC 23009-3:2014/Amd 1 and illustrated in Annex B.

7.4 Semantics

7.4.1 Decoder-power indication metadata semantics

num_dec_ops(i) – the estimated number of decoding operations required for the i^{th} Representation of the current video Segment.

num_prev_dec_ops(i) – the estimated number of decoding operations required for the i^{th} Representation of the previous video Segment in a given Period. If the current video Segment is the first segment of a Period, then num_prev_dec_ops(i) = num_dec_ops(i).

max_num_dec_ops – the estimated number of decoding operations required for the most demanding Representation of the current video Segment.

dec_ops_reduction_ratio_from_max(i) - the percentage by which decoding operations are reduced in the i^{th} Representation compared to the most demanding Representation of the current video Segment:

$$\text{dec\_ops\_reduction\_ratio\_from\_prev(i)}=\text{Floor}\left(\frac{\text{max\_num\_dec\_ops-num\_dec\_ops(i)}}{\text{max\_num\_dec\_ops}}*100\right) \quad (7\text{-}1)$$

dec_ops_reduction_ratio_from_prev(i) - the percentage by which decoding operations are reduced in the current video Segment compared to the previous video Segment for the i^{th} Representation in a given Period. A negative value means an increase in decoding operations:

$$\text{dec\_ops\_reduction\_ratio\_from\_prev(i)}=\text{Floor}\left(\frac{\text{num\_prev\_dec\_ops(i)-num\_dec\_ops(i)}}{\text{num\_prev\_dec\_ops(i)}}*100\right) \quad (7\text{-}2)$$

If the current video Segment is the first Segment of a Period, then dec_ops_reduction_ratio_from_prev(i) = 0.

7.4.2 Display-power indication metadata semantics

ms_num_quality_levels – the number of quality levels that are enabled by the metadata.

ms_rgb_component_for_infinite_psnr – the average, over the *N* reconstructed frames of the video Segment, of the largest RGB component (as defined in 3.1) in each of the reconstructed frames.

ms_max_rgb_component[i] – for the i^{th} quality level (1 <= i <= num_quality_levels), the average, over the N reconstructed frames of the video Segment, of the maximum RGB component that will be retained in each of the reconstructed frames. Note that ms_max_rgb_component[0] = ms_rgb_component_for_infinite_psnr.

ms_scaled_psnr_rgb[i] – for the i^{th} quality level (1 <= i <= num_quality_levels), the average, over the N reconstructed frames in the video segment, of scaled_psnr_rgb[i] computed for each frame as defined in 6.4, with $N_{frames} = 1$. Note that ms_scaled_psnr_rgb[0] is associated with the NQLOP. It is not transmitted, but understood to be mathematically infinite.

8 Metrics for quality recovery after low-power encoding

8.1 General

An encoder can achieve power reduction by encoding alternating high-quality and low-quality Segments, in a segmented delivery mechanism such as DASH. The power reduction occurs because low-complexity encoding mechanisms are used to produce the low-quality Segments. A metric describing the quality of the last picture of each Segment is delivered as metadata to the decoder. The metric is utilized, by the decoder, in conjunction with the last frame of the prior high-quality Segment to enhance the quality of the low-quality Segment and, thereby, ameliorate any negative visual impact. Annex B describes in detail how cross-segment decoding may be used to improve the quality of the low-quality Segments.

8.2 Syntax

The encoder embeds the following message in the last picture of each Segment using the following syntax:

	Size (bits)	Descriptor
xsd_metric_type	8	unsigned integer
xsd_metric_value	16	unsigned integer

8.3 Signalling

SEI messages can be used to signal Green Metadata in an AVC stream. The Green Metadata SEI message payload type for AVC is specified in ISO/IEC 14496-10:2014/Amd. 2.

The SEI message for Green Metadata can be used to signal the preceding message as explained in Annex A.

8.4 Semantics

xsd_metric_type - indicates the type of the objective quality metric as shown in the table below. PSNR, as defined in ISO/IEC 23001-10, is the only type currently supported.

Value	Description
0x00	PSNR
0x01–0xFF	User-defined

xsd_metric_value - contains the metric value of the last picture of the Segment. When xsd_metric_type is 0, then the stored 16-bit unsigned integer xsd_metric_value, is interpreted as a floating-point PSNR value (in dB) as follows:

$$\mathrm{PSNR} = \frac{\mathrm{xsd\_metric\_value}}{100} \tag{8.1}$$

Annex A
(normative)

Supplemental Enhancement Information (SEI) syntax

A.1 SEI payload syntax in AVC

sei_payload(payloadType, payloadSize)	C	Descriptor
If(payloadType = = 0)		
buffering_period(payloadSize)	5	
…		
Else if(payloadType = = xx)		
green_metadata(payloadSize)	5	
Else		
reserved_sei_message(payloadSize)	5	
…		
}		

A.2 Green Metadata SEI message syntax and semantics in AVC

A.2.1 Syntax

	C	Descriptor
green_metadata(payload_size)		
green_metadata_type	5 5	u(8)
switch (**green_metadata_type**) {		
case 0:		
period_type	5	u(8)
if (period_type = = 2) {		
num_seconds	5	u(16)
}		
else if (period_type = = 3) {	5	
num_pictures	5	u(16)
}		
percent_non_zero_macroblocks	5	u(8)
percent_intra_coded_macroblocks	5	u(8)
percent_six_tap_filterings	5	u(8)
percent_alpha_point_deblocking_instances	5	u(8)
break;		
case 1:		
xsd_metric_type	5	u(8)

	C	Descriptor
xsd_metric_value	5	u(16)
break;		
default:		
}		

A.2.2 Semantics

green_metadata_type – specifies the type of metadata that is present in the SEI message. If green_metadata_type is 0, then complexity metrics are present. Otherwise, if green_metadata_type is 1, then metadata enabling quality recovery after low-power encoding is present.

Annex B
(normative)

Implementation guidelines for the usage of Green Metadata

B.1 Codec dynamic voltage frequency scaling for decoder-power reduction

B.1.1 General

Codec Dynamic Voltage Frequency Scaling (C-DVFS) uses the DVFS technique to scale the voltage and operating frequency of the CPU to achieve power savings while decoding a bitstream. Typically the dynamic power consumption of a CMOS circuit increases monotonically with the operating frequency. The power-optimization module at the receiver extracts the Complexity Metrics (CMs) metadata that indicates picture-decoding complexity. It uses these CMs to determine and set the optimum operating voltage and frequency of the CPU so that video pictures are correctly decoded with minimal power consumption. By embedding these CMs as metadata into the bitstream at the encoder, C-DVFS enabled receivers will achieve power reduction.

B.1.2 Derivation of the complexity metrics

5.2.2 specifies these four CMs: percent_non_zero_macroblocks, percent_intra_coded_macroblocks, percent_six_tap_filterings and percent_alpha_point_deblocking_instances. The computation of the first two CMs, as explained in 5.2.4, is straightforward. However, computation of percent_six_tap_filterings and percent_alpha_point_deblocking_instances is more involved. To provide a better understanding of these two CMs, the next two subclauses describe how max_num_six_tap_filterings_pic(i) and max_num_alpha_point_deblocking_instances_pic(i) are derived.

B.1.2.1 Deriving the worst-case, largest value for max_num_six_tap_filterings_pic(i)

To determine max_num_six_tap_filterings_pic(i), the following terms, as defined in ISO/IEC 14496-10, are referenced: motion vector, PicSizeInMbs, reference picture list. At the decoder, the worst-case, largest number of Six-Tap Filterings (STFs) occurs in a picture when all partitions consist of 4x4 blocks that will be interpolated. The 4x4 blocks produce the largest number of STFs because the overhead from interpolating samples that are outside the block is larger for 4x4 blocks than for 8x8 blocks as explained below.

In Figure B.1, upper-case letters represent integer samples and lower-case letters represent fractional sample positions. Subscripts are used to indicate the integer sample that is associated with a fractional sample position. The subsequent analysis is for the worst-case largest number of STFs for the interpolation of the 4x4-block consisting of samples G, H, I, J, M, N, P, Q, R, S, V, W, T, U, X, Y. This interpolation must be performed when a motion vector (MV) points to one of the following fractional-sample positions: a_G, b_G, c_G, d_G, e_G, f_G, g_G, h_G, i_G, j_G, k_G, n_G, p_G, q_G, r_G. If the MV points to a_G, then the decoder must compute a_G and the 15 points (a_H, a_I, ...) that have the same respective relative locations to H, I, J, M, N, P, Q, R, S, V, W, T, U, X, Y that a_G has to G. Similarly, the decoder would need to compute 16 points for each of the other fractional-sample positions (b_G, c_G, ..., r_G) that the MV could point to. To determine the worst-case largest number of STFs for the interpolation of the 4x4 block, here is a count of the STFs required for each fractional-sample position that the MV could point to.

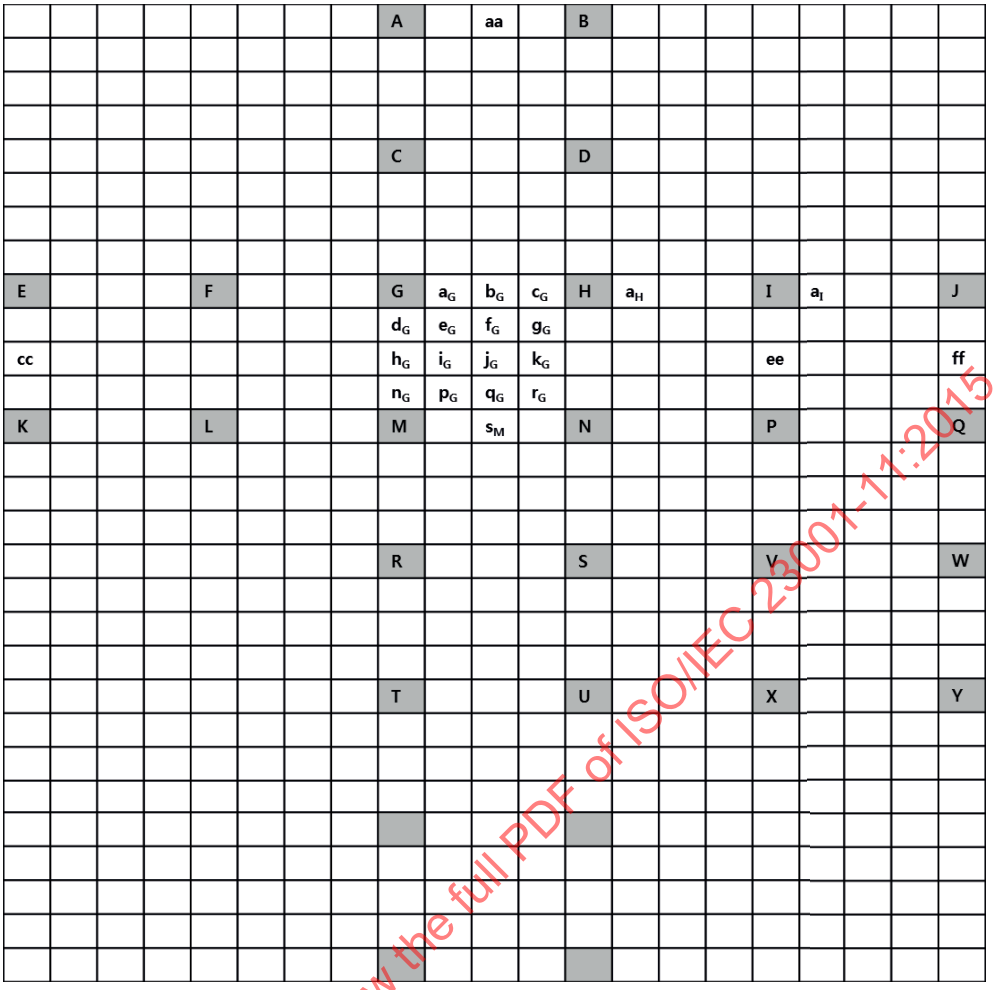

Figure B.1 — Quarter-sample interpolation of the 4x4-block consisting of samples G, H, I, J, M, N, P, Q, R, S, V, W, T, U, X, Y

1 If the MV points to b_G, then to interpolate b_G, the decoder must apply 1 STF to E, F, G, H, I, J which are already available as integer samples. So 16 STFs are needed to compute b_G, ..., b_Y for the 4x4 block.

2 If the MV points to h_G, then to interpolate h_G, the decoder must apply 1 STF to A, C, G, M, R, T which are already available as integer samples. So 16 STFs are needed to compute h_G, ..., h_Y for the 4x4 block.

3 If the MV points to j_G, then to interpolate j_G, the decoder must apply 6 STFs to compute aa, bb, b_G, s_M, gg, hh because these are unavailable. Next, 1 STF is needed to compute j_G from aa, bb, b_G, s_M, gg, hh. So 7 STFs are required for j_G.

   a To get j_M, the decoder needs bb, b_G, s_M, gg, hh, ii. Only ii is unavailable. So 2 STFs are needed for j_M (one for ii and one for j_M).

   b To get j_R, the decoder needs 2 STFs (one for jj and one for j_R).

   c To get j_T, the decoder needs 2 STFs (one for kk and one for j_T).

d Therefore, for j_G, j_M, j_R and j_T, the decoder needs 7 + 2 + 2 + 2 = 13 STFs. Since the computation is identical for each of the four columns GMRT, HNSU, IPVX and JQWY, the decoder needs 13 * 4 = 52 STFs to compute j_G, ... j_Y for the 4x4 block.

4 If the MV points to a_G, then to interpolate a_G, the decoder needs 1 STF to get b_G (from (1)) and therefore 16 STFs to compute a_G, ..., a_Y for the 4x4 block.

5 If the MV points to c_G, then to interpolate c_G, the decoder needs 1 STF to get b_G (from (1)) and therefore 16 STFs to compute c_G, ..., c_Y for the 4x4 block.

6 If the MV points to d_G, then to interpolate d_G, the decoder needs 1 STF to get h_G (from (2)) and therefore 16 STFs to compute d_G, ..., d_Y for the 4x4 block.

7 If the MV points to n_G, then to interpolate n_G, the decoder needs 1 STF to get h_G (from (2)) and therefore 16 STFs to compute n_G, ..., n_Y for the 4x4 block.

8 If the MV points to f_G, then to interpolate f_G, the decoder needs 7 STFs to get j_G (from (3)). Note that b_G is included in these 7 STFs. Therefore, from (3), 52 STFs are required to compute f_G, ... f_Y for the 4x4 block.

9 If the MV points to i_G, then to interpolate i_G, the decoder needs 7 STFs to get j_G. Note that h_G is computed by one of these 7 STFs. Therefore, 52 STFs are required to compute i_G, ... i_Y for the 4x4 block. For this analysis, the row j_G, j_H, j_I, j_J is computed first (to obtain h_G) and then this process is repeated for the other 3 rows (MNPQ, RSVW, TUXY) in the 4x4 block. Previously, in (3), Column GMRT was analysed first and the analysis was then repeated for the other 3 columns (HNSU, IPVX, JQWY).

10 If the MV points to k_G, then to interpolate k_G, the decoder needs 7 STFs to get j_G. Note that m_G is computed by one of these 7 STFs. Therefore, 52 STFs are required to compute k_G, ... k_Y for the 4x4 block.

11 If the MV points to q_G, then to interpolate q_G, the decoder needs 7 STFs to get j_G. Note that s_G is computed by one of these 7 STFs. Therefore, 52 STFs are required to compute q_G, ... q_Y for the 4x4 block.

12 If the MV points to e_G, then to interpolate e_G, the decoder needs 2 STFs to get b_G and h_G (from (1), (2)). Therefore 32 STFs are needed to compute e_G, ..., e_Y for the 4x4 block.

13 If the MV points to g_G, then to interpolate g_G, the decoder needs 2 STFs to get b_G and m_H. Therefore, 32 STFs are needed to compute g_G, ..., g_Y for the 4x4 block.

14 If the MV points to p_G, then to interpolate p_G, the decoder needs 2 STFs to get h_G and s_G. Therefore, 32 STFs are needed to compute p_G, ..., p_Y for the 4x4 block.

15 If the MV points to r_G, then to interpolate r_G, the decoder needs 2 STFs to get m_G and s_G. Therefore, 32 STFs are needed to compute r_G, ..., r_Y for the 4x4 block.

From (1),...,(15), the worst-case, largest number of STFs is 52, when the MV points to j_G, f_G, i_G, k_G or q_G. Since the overhead of filtering samples outside the block is smaller for larger block sizes, the worst case STFs is when all partitions are 4x4 blocks and two MVs are used for each block (one from each reference picture list). In this case, the worst-case, largest number of STFs in a picture is

max_num_six_tap_filterings_pic(i) = (worst-case number of STFs in a 4x4 block) *

(worst-case number of reference picture lists) *

(PicSizeInMbs) *

(number of 4x4 luma blocks in a macroblock)

$$= 52 * 2 * \text{PicSizeInMbs} *16$$

$$= 1664 * \text{PicSizeInMbs} \quad \text{(B-1)}$$

B.1.2.2 Deriving the worst-case, largest value for max_num_alpha_point_deblocking_instances_pic(i)

To determine max_num_alpha_point_deblocking_instances_pic(i), the following analysis determines the worst-case, largest number of Alpha-Point Deblocking Instances (APDIs) that can occur when deblocking a picture at the decoder. The following terms, as defined in ISO/IEC 14496-10, are referenced: raster scan, PicSizeInMbs.

Consider a macroblock containing a 16x16 luma block in which the samples have been numbered in raster-scan order as shown in Figure B.2. Upper-case roman numerals are used to reference columns of samples and lower-case roman numerals are used to reference rows of samples. For example, Column IV refers to the column of Samples 4, 20, ... 244 and Row xiii refers to the row of samples 193, 194, ..., 208. Edges are indicated by an ordered pair that specifies the columns or rows on either side of the edge. For example, Edge (IV, V) refers to the vertical edge between Columns IV and V. Similarly, Edge (xii, xiii) indicates the horizontal edge between Rows xii and xiii. Note that the leftmost vertical edge and the topmost horizontal edge are denoted by (0, I) and (0, i) respectively.

The maximum number of APDIs occurs when the 4x4 transform is used on each block and a single APDI occurs in every set of eight samples across a 4x4 block horizontal or vertical edge denoted as p_i and q_i with i = 0,...,3 as shown in Figure 8-11 of ISO/IEC 14496-10.

For the macroblock in Figure B.2, the Vertical Edges (0, I), (IV, V), (VIII, IX) and (XII, XIII) are filtered first. Then the Horizontal Edges (0,i), (iv, v), (viii, ix) and (xii, xiii) are filtered. Now, when Vertical Edge (0, I) is filtered, in the worst-case, an APDI will occur on each row of the edge because the q_0 Samples 1, 17, ... 241 will all be APDIs. Therefore, 16 APDIs will occur in Vertical Edge (0, I). Similarly, when Vertical Edge (IV, V) is filtered, there will also be 16 APDIs corresponding to the 16 (p_0, q_0) sample pairs (20, 21), (36, 37), ... (244, 245). Thus, there will be 16*4 = 64 APDIs from vertical-edge filtering. After horizontal-edge filtering, there will be an additional 64 APDIs because each horizontal edge will contribute 16 APDIs. For example, Horizontal Edge (viii, ix) will contribute the 16 APDIs corresponding to the (p_0, q_0) sample pairs (113, 129), (114, 130), ..., (128, 144). Hence, in the worst-case, deblocking the luma block in a macroblock produces 128 APDIs.

Next, consider the two chroma blocks corresponding to the luma block in the macroblock. The worst-case number of APDIs is determined by the chroma sampling relative to the luma sampling.

	I	II	III	IV	V	VI	VII	VIII	IX	X	XI	XII	XIII	XIV	XV	XVI
i	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
ii	17			20	21			24	25			28	29			
iii	33			36	37			40	41			44	45			
iv	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64
v	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80
vi	81			84	85			88	89			92	93			
vii	97			100	101			104	105			108	109			
viii	113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128
ix	129	130	131	132	133	134	135	136	137	138	139	140	141	142	143	144
x	145			148	149			152	153			156	157			
xi	161			164	165			168	169			172	173			
xii	177	178	179	180	181	182	183	184	185	186	187	188	189	190	191	192
xiii	193	194	195	196	197	198	199	200	201	202	203	204	205	206	207	208
xiv	209			212	213			216	217			220	221			
xv	225			228	229			232	233			236	237			
xvi	241			244	245			248	249			252	253			256

Figure B.2 — 16x16 luma block. Upper-case roman numerals reference columns of samples and lower-case roman numerals reference rows of samples.

1 For each chroma block in 4:2:0 format, two vertical edges and two horizontal edges are filtered. Each edge contributes 8 APDIs, in the worst-case. So, 8*4*2 = 64 APDIs are produced by worst-case deblocking of the two chroma blocks.

2 For 4:2:2 format, two vertical edges and four horizontal edges are filtered. Each vertical edge contributes 16 APDIs and each horizontal edge contributes 8 APDIs. So, 2*(2*16 + 4*8) = 128 APDIs are produced by worst-case deblocking of the two chroma blocks.

3 For 4:4:4 format, the worst-case analysis for each chroma block is identical to that of the 16x16 luma block. Therefore, 256 APDIs are produced by worst-case deblocking of the two chroma blocks.

4 Finally, for separate colour planes, the worst-case analysis of a 16x16 block is identical to that a 16x16 luma block.

To conclude, since each picture has PicSizeInMbs macroblocks, the worst-case number of APDIs per picture, is as follows:

max_num_alpha_point_deblocking_instances_pic(i)

= PicSizeInMbs * (128 + 64) = 192 * PicSizeInMbs, for 4:2:0,

= PicSizeInMbs * (128 + 128) = 256 * PicSizeInMbs, for 4:2:2,

= PicSizeInMbs * (128 + 256) = 384 * PicSizeInMbs, for 4:4:4,

= 128 * PicSizeInMbs, for a single colour plane. (B-2)

B.1.3 Example usage of C-DVFS metadata

C-DVFS metadata may be signalled at a picture, group of pictures, or scene level and can therefore be adapted to application requirements. Signalling may be done with SEI messages. With SEI-message signalling, each time the SEI message is encountered by the decoder, a new upcoming period begins. The value period_type indicates whether the new upcoming period is a single picture, a single group of pictures, or a time interval (specified in seconds or number of pictures). Figure 1 shows an example process for metadata extraction, complexity prediction, DVFS control-parameter determination and decoding under DVFS control. As an example, assume that the upcoming period is a single picture. Then, the SEI message is parsed to obtain percent_non_zero_macroblocks, percent_intra_coded_macroblocks, percent_six_tap_filterings and percent_num_alpha_point_deblocking_instances. From these percentage values and the corresponding worst-case instances the four CMs are derived: num_non_zero_macroblocks (n_{nz}), num_intra_coded_macroblocks (n_{intra}), num_six_tap_filterings (n_{six}), and num_alpha_point_deblocking_instances (n_{α}). Once the complexity parameters are derived, the total picture complexity ($C_{picture}$) is estimated or predicted according to Formula B-3:

$$C_{picture} = K_{init} * n_{MB} + k_{bit} * n_{bit} + k_{nz} * n_{nz} + k_{intra} * n_{intra} + k_{six} * n_{six} + k_{\alpha} * n_{\alpha} \quad \text{(B-3)}$$

where $C_{picture}$ is the total picture complexity. The total number of macroblocks per picture (n_{MB}) and the number of bits per picture (n_{bit}) can be easily obtained after de-packetizing the encapsulated packets and parsing the sequence parameter set. Constants k_{init}, k_{bit}, k_{nz}, k_{intra}, k_{six}, and k_{α} are unit-complexity constants for performing macroblock initialization (including parsed data filling and prefetching), single-bit parsing, non-zero block transform and quantization, intra-block prediction, inter-block six-tap filtering, and deblocking alpha-points filtering, respectively. Note that k_{nz}, k_{intra}, and k_{six} are fixed constants for a typical platform, while k_{init}, k_{bit}, and k_{α} can be accurately estimated using a linear predictor from a previous decoded picture.

Once the picture complexity is determined, the decoder applies DVFS to determine a suitable clock frequency and supply voltage for the decoder. Then, the decoder can decode the video picture at the appropriate clock frequency and supply voltage.

The DVFS-enabling SEI message can be inserted into the bitstream on a picture-by-picture, scene-by-scene, or even time-interval-by-time-interval basis, depending on the underlying application. Therefore, the SEI message can be inserted once at the start of each picture, scene, or time interval. A scene-interval or time-interval inserted message requires less overhead than a picture-level inserted message. For processors that don't support high-frequency DVFS (e.g. adapting at 33 ms for 30Hz video playback), setting period_type to an interval is preferable to setting period_type to a picture. Once all complexity metrics are obtained from the SEI message, the decoder estimates the complexity for the next picture, group of pictures, or time interval as indicated by period_type. This complexity is then used to adjust the voltage and frequency for the upcoming period.

In a hardware (ASIC) implementation, instead of deriving decoding complexity and using it directly to control a single clock frequency in a DVFS scheme, the ASIC can be designed so that it includes several distinct clock domains, each of which corresponds to one of the terms in Formula B-3. Greater power reduction can be obtained by using such a flexible ASIC with distinct clock domains. For example, six clock domains in the ASIC can control the following six sections of the ASIC: macroblock initialization, bit parsing, transform and quantization, intra-block prediction, interpolation, and deblocking. To achieve fine-grained DVFS adjustments, the clock frequencies in each domain may be varied in proportion to the

corresponding term in Formula B-3. Accordingly, the preceding clock domains can have instantaneous clock frequencies that are respectively proportional to the following terms: $k_{init} * n_{MB}$, $k_{bit} * n_{bit}$, $k_{nz} * n_{nz}$, $k_{intra} * n_{intra}$, $k_{six} * n_{six}$, and $k_{\alpha} * n_{\alpha}$.

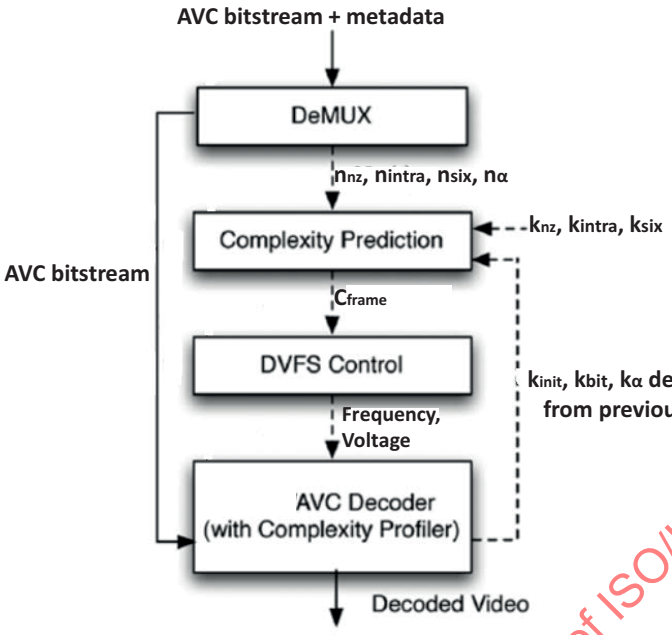

Figure B.3 — Example of parsing, complexity prediction, and DVFS control

B.2 Display adaptation

B.2.1 General

Display Adaptation (DA) achieves power savings by scaling up the RGB components in the reconstructed frames while reducing the backlight or voltage proportionally. The decreased backlight or voltage reduces display power consumption while still producing the same perceived display. The metadata in 6.2.1 can be stored using the file format specified in ISO/IEC 23001-10 or the metadata can be carried by MPEG-2 systems as specified in ISO/IEC 13818-1:2013/Amd. 3:2014.

B.2.2 Example usage of display-adaptation metadata

The metadata scaled_psnr_rgb[i] indicates the PSNR for the i^{th} quality level. At the transmitter, reconstructed frames are available within the encoder and scaled_frames[i] is estimated by saturating all RGB components of reconstructed frames to max_rgb_component[i]. The scaled_frames[i] thus obtained are what would be perceived at the display after the receiver scales the RGB components of reconstructed frames by (peak signal / max_rgb_component[i]) and then applies the backlight scaling factor, b = (max_rgb_component[i] / peak signal) to the LCD backlight. scaled_psnr_rgb[i] is computed at the transmitter using peak signal and by assuming that the noise is the difference between scaled_frames[i] and reconstructed frames accumulated over R, G and B components, as explained in 6.4.

The receiver examines the (num_quality_levels + 1) pairs of metadata and selects the pair (max_rgb_component[iSelected], scaled_psnr_rgb[iSelected]) for which scaled_psnr_rgb[iSelected] is an acceptable quality level. Then, the receiver derives DA scaling factors from max_rgb_component[iSelected]. Finally, the display scales the RGB components of reconstructed frames by peak signal / max_rgb_component[iSelected] and it scales the backlight or voltage level by max_rgb_component[iSelected] / peak signal. After backlight scaling, the displayed pixels are perceived as scaled_frames[iSelected]. The metadata clearly enables a trade-off between quality (PSNR) and power reduction (backlight scaling factor).

The following power-saving protocol can be implemented in a mobile device. The user specifies a list of N acceptable PSNR quality levels Q[1], …, Q[N], where Q[1] > Q[2] > …> Q[N] and a list of Remaining Battery Life Levels (RBLLs) RBLL[1], …, RBLL[N] so that RBLL[1] > RBLL[2] > … > RBLL[N]. For example, consider N = 3 and Q[1] = 40, Q[2] = 35, Q[3] = 25 with RBLL[1] = 70%, RBLL[2] = 40% and RBLL[3] = 0%. When the user watches a video, the device monitors the actual RBLL, denoted RBLLactual, of the device and selects RBLL[iSelected] so that RBLL[iSelected-1] > RBLLactual > RBLL[iSelected], where RBLL[0] = 100%. For each frame to be displayed, the device examines the display-adaptation metadata and selects the pair indexed by jSelected for which Q[iSelected-1] > scaled_psnr_rgb[jSelected] > Q[iSelected], where Q[0] = infinity. The metadata max_rgb_component[jSelected] is then used to determine display-adaptation scaling parameters. Thus, the device will implement a protocol that strikes a balance between perceived quality and power-saving. The balance is tilted toward quality when the RBLL is high but shifts toward power saving as the battery is depleted.

B.2.2.1 Example usage of display-adaptation metadata for contrast enhancement

At low quality levels, contrast enhancement significantly improves perceived visual quality, especially for bright content. To enhance contrast at the lowest quality level associated with the backlight scaling factor b = (max_rgb_component[num_quality_levels] / peak signal) the receiver first examines lower_bound. If it is greater than zero, then contrast enhancement metadata is available and the receiver stores upper_bound. The presentation subsystem performs contrast enhancement by setting the backlight scaling factor to b = (max_rgb_component[num_quality_levels] / peak signal), and for each RGB component, x, of reconstructed frames, the following scaling to S(x) is performed:

S(x) = 0, for x in [0, lower_bound],

= peak signal * (x-lower_bound) / (upper_bound – lower_bound),

for x in (lower_bound, upper_bound),

= peak signal, for x in [upper_bound, peak signal]

Observe that the interval (lower_bound, upper_bound) is mapped to the interval (0, peak signal). Then, after applying the backlight scaling factor, b, to the display, the interval (lower_bound, upper_bound) is perceived visually as the interval (0, b * peak signal). Therefore, for RGB components within the interval (lower_bound, upper_bound), the perceived contrast enhancement is proportional to b * peak signal / (upper_bound – lower_bound). This expression simplifies to b / (upper_bound – lower_bound), because peak signal is a constant. For RGB components within the intervals [0, lower_bound] and [upper_bound, peak signal], all contrast is lost because these intervals are mapped to 0 and peak signal, respectively.

From the preceding observation, it is clear that the contrast is maximized by determining lower_bound and upper_bound so that the majority of RGB components lie within the interval (lower_bound, upper_bound). Therefore, the optimal contrast-enhancement metadata is computed by the following process, at the transmitter. First, determine the backlight_scaling_factor corresponding to the lowest quality level

as b = max_rgb_component[num_quality_levels] / peak signal. Then, invoke the following pseudocode function get_contrast_metadata() to determine lower_bound and upper_bound.

```
// Given RGB components, x, of reconstructed frames with
// cumulative distribution function, C(x), the function get_contrast_metadata() returns
// lower_bound and upper_bound.
[lower_bound, upper_bound] = get_contrast_metadata(C(x)) {
// C(x): Cumulative distribution function of RGB components of reconstructed frames.
max_enhancement = 0;
for (lower_bound = 0; lower_bound < peak signal; lower_bound++){
  for (upper_bound = lower_bound; upper_bound < peak signal; upper_bound++){
    enhancement = (C(upper_bound) – C(lower_bound)) / (upper_bound – lower_bound)
    if (enhancement > max_enhancement) {
      max_enhancement = enhancement;
      best_lower_bound = lower_bound;
      best_upper_bound = upper_bound;
    }
  }
}
return (best_lower_bound, best_upper_bound);
}
```

Although the metadata computed by get_contrast_metadata() is optimal for each frame, flicker artefacts may occur when the video is viewed due to large differences between lower_bound (or upper_bound) settings on successive video frames. To avoid such flicker, the lower_bound and upper_bound metadata should be smoothed temporally using the pseudo-code function smooth_contrast_metadata() shown below.

```
// Given a video sequence with frameNum in [1,...,N], first smooth the lower bounds by
// applying the function recursively to all frames by issuing
// smooth_contrast_metadata(LowerBounds,1),
// ...
// smooth_contrast_metadata(LowerBounds,N)
// Then smooth the upper bounds by issuing
// smooth_contrast_metadata(UpperBounds,1),
// ...
// smooth_contrast_metadata(UpperBounds,N)
// where
// LowerBounds: vector of lower_bound metadata for the N frames
// UpperBounds: vector of upper_bound metadata for the N frames
void smooth_contrast_metadata(Vector, frameNum) {
// Vector: vector of metadata to be smoothed
// frameNum: current frame number
cur = Vector[frameNum]
prev = Vector[frameNum - 1]
if Abs((cur - prev) / prev) > Threshold { // Check whether the metadata variation between
                                          // successive frames exceeds the threshold.
 if (cur < prev) { // if the current frame's metadata are lower than the previous frame's metadata,
                   // then increase the current frame's metadata so that it reaches the acceptable
                   // threshold.
   Vector[frameNum] = prev * (1 - Threshold)
} else { // increase the previous frame's metadata so that it reaches the acceptable
         // threshold. Then adjust the metadata for all preceding frames.
   Vector[frameNum - 1] = cur / (1 + Threshold)
   smooth_contrast_metadata(Vector, frameNum - 1)
 }
}
```

The value of Threshold is display independent and can be set to 0,015, which corresponds to a 1,5% metadata variation between successive frames.

B.2.2.2 Preventing flicker arising from control latency

If DA metadata were unavailable, then to implement DA, the display would have to estimate max_rgb_component[i] and immediately adjust the backlight (or voltage). This is impossible in most practical implementations because there is a significant latency of D milliseconds between the instant when the backlight scaling control is applied and the instant when the backlight actually changes, in response to the control. If D is sufficiently large, then the backlight values will not be synchronized with the displayed frames and flickering is visible. Fortunately, DA metadata eliminates this flickering. Because the receiver obtains the metadata in advance, the backlight scaling factor can be applied D milliseconds ahead of the video frame with which that scaling factor is associated. Therefore, by transmitting metadata, the latency issue is solved and the backlight scaling factor will be set appropriately for each frame. This avoids flicker from backlight changes during video display.

B.2.2.3 Metadata for DA on displays with control-frequency limitations

Besides eliminating flicker arising from backlight-control latency, DA metadata can also enable DA to be applied to displays in which the backlight (or voltage) cannot be changed frequently. For such displays, once the backlight has been updated it must retain its value for a time interval that spans the duration of some number of successive frames. After the time interval has elapsed, the backlight may be updated again. DA metadata allows the backlight to be set appropriately for the specified time interval so that maximal power reduction and minimal RGB-component saturation occurs. This appropriate backlight value is determined by aggregating the RGB component histograms in all successive frames in each time interval over which the backlight must remain constant. The aggregated histograms are then used to derive DA metadata, as explained in preceding subclauses. To enable this mode of operation, the receiver must signal to the transmitter, constant_backlight_voltage_time_interval, the time interval over which the backlight (or voltage) must remain constant. Alternatively, the transmitter may assume a reasonable value for constant backlight voltage time interval.

On currently available displays, setting constant_backlight_voltage_time_interval to 100 milliseconds is sufficient to prevent flicker. Therefore, setting num_constant_backlight_voltage_time_intervals = 1 and constant_backlight_voltage_time_interval[0] = 100 is sufficient to prevent flicker arising from control-frequency limitations. However, in the future, a new display technology with constant_backlight_voltage_time_interval significantly different from 100 milliseconds may be invented. During the transition period from the current display technology to the new display technology, two types of displays will be widely used and it will be necessary to set num_constant_backlight_voltage_time_intervals = 2, to support both display types. The preceding mode of operation assumes that a signalling mechanism from the receiver to the transmitter does not exist.

However, if such a signalling mechanism does exist, then the receiver can explicitly signal constant_backlight_voltage_time_interval to the transmitter as explained in 6.2.2 and 6.3.2. If the transmitter is additionally capable of re-computing the display adaptation metadata to be consistent with the signalled constant_backlight_voltage_time_interval, then the re-computed metadata can subsequently be provided to the receiver.

B.2.2.4 DA metadata to prevent flicker from large variations

On some platforms, besides the flicker that arises from control latency and control-frequency limitations, flicker can also occur due to a large difference between the backlight (or voltage) settings (defined as backlight_scaling_factor in 6.4) of successive video frames. To avoid such flicker, a transmitter can use the function in the table below to adjust the backlight setting of each frame. Specifically, if the relative backlight variation between a frame and its predecessor is larger than a threshold, then the backlight values of all preceding frames must be adjusted. This adjustment is done at the transmitter after metadata has been computed using one of the methods described in the preceding subclauses.

For example, for a targeted quality level, the transmitter would estimate max_rgb_component and the corresponding backlight_scaling_factor for each of N frames. Given max_variation (normalized to 255), the transmitter applies adjust_backlight() with the specified max_variation threshold computed as the floating-point number (max_variation/2048). This function adjusts the vector of backlight_scaling_factor values for the *N* frames so that the relative backlight variation between successive frames is

less than max_variation. After the backlight values have been adjusted, the DA metadata is modified, if necessary, to be consistent with the adjusted backlight values.

```
// Given a video sequence with frameNum in [1,...,N], apply the function
// recursively to all frames by issuing
// adjust_backlight(Backlights,1,max_variation),
// ...
// adjust_backlight(Backlights,N,max_variation)
void adjust_backlight(Backlights, frameNum, max_variation) {
// Backlights: vector of backlight_scaling_factor values
// frameNum: current frame number
// max_variation: maximum permissible backlight variation between two
// consecutive backlight values
cur = Backlights[frameNum]
prev = Backlights[frameNum - 1]
if Abs((cur - prev) / prev) > max_variation { // Check whether the backlight variation between
                                              // successive frames exceeds the threshold.
if (cur < prev) { // if the current frame's backlight is lower than the previous frame's backlight,
                  // then increase the current frame's backlight so that it reaches the acceptable threshold.
  backlights[frameNum] = prev * (1 - max_variation)
}else { // increase the previous frame's backlight so that it reaches the acceptable
        // threshold.Then adjust the backlights for all preceding frames.
  backlights[frameNum - 1] = cur / (1 + max_variation)
  adjust_backlight(backlights, frameNum - 1, max_variation)
}
}
```

For a given display, large values of max_variation will induce more flicker but also save more power. Therefore, the selected value of max_variation is a compromise between flicker reduction and power saving. The max_variation metadata guarantees that the receiver will not experience flicker because the backlights are adjusted specifically for the receiver's display.

On currently available displays, setting max_variation = 0,015*2 048 is sufficient to prevent flicker. Therefore, setting num_max_variations = 1 and max_variation = 0,015*2 048 is sufficient to prevent flicker arising from control-frequency limitations. However, in the future, a new display technology with max_variation significantly different from 0,015*2 048 may be invented. During the transition period from the current display technology to the new display technology, two types of displays will be widely used and it will be necessary to set num_max_variations = 2, to support both display types. The preceding mode of operation assumes that a signalling mechanism from the receiver to the transmitter does not exist.

However, if such a signalling mechanism does exist, then the receiver can explicitly signal max_variation to the transmitter as explained in 6.3.2. If the transmitter is additionally capable of re-computing the display adaptation metadata to be consistent with the signalled max_variation, then the re-computed metadata can subsequently be provided to the receiver.

B.3 Energy-efficient media selection in adaptive streaming

B.3.1 General

This clause explains how the Green Metadata for adaptive streaming can be computed at the server and how such metadata can be used at the client.

B.3.2 Green Metadata production and transmission at the server side

Given *N* video Representations, the decoder-power indication metadata dec_ops_reduction_ratio_from_max(i) (DOR-Ratio-Max(i)) and dec_ops_reduction_ratio_from_prev(i) (DOR-Ratio-Prev(i)) are computed by the encoding system and provided by the server for i = 0 to N-1, as shown in Figure B.4. The display-power indication metadata is computed from one Representation.

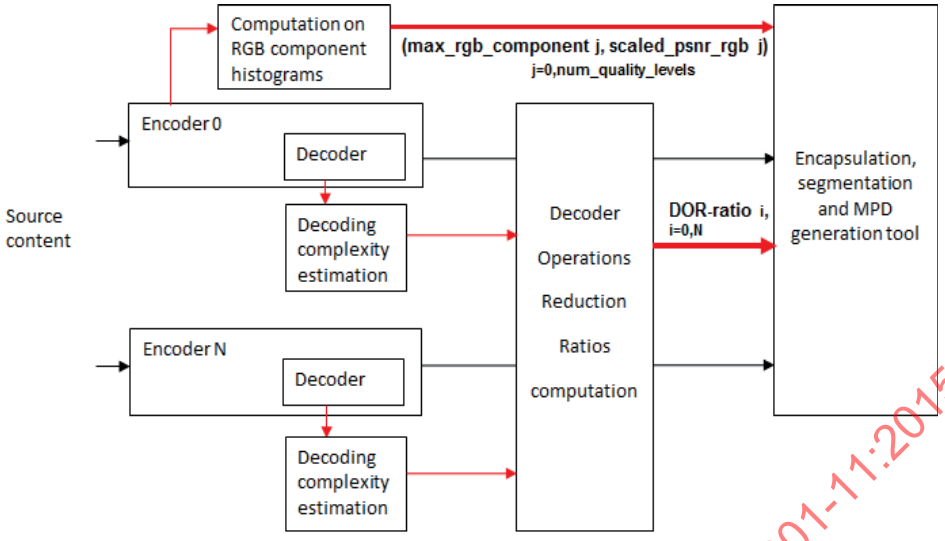

Figure B.4 — Green Metadata computation and insertion

The DOR-Ratio-Max(i) associated with each video Representation i of a Segment is computed as the power-saving ratio from the most demanding video Representation produced for the Segment, as defined in 7.4.1.

The DOR-Ratio-Prev(i) associated with each video Representation i of a Segment is computed as the power-saving ratio from the previous Segment of the same Representation, as defined in 7.4.1.

To produce the normative Green Metadata DOR-Ratio-Max(i) and DOR-Ratio-Prev(i) for a given Segment, the encoding system needs to estimate the decoding complexity of each video Representation, as a number of processing cycles.

Each sample which contains the DOR-Ratio values is then stored in a specific metadata file "id/$Time$.mp4m" (one for each Segment) using the format specified in ISO/IEC 23001-10. In the DASH context, the metadata files created for one or multiple video Representations will be considered as metadata Representations. The available metadata Representations will be signalled in a specific Adaptation Set within the MPD. The association of a metadata Representation with a media Representation is signalled in the MPD through the @associationId and @associationType attributes. A metadata Segment and its associated media Segment(s) are time aligned on Segment boundaries.

The decoder-power indication metadata Representation is associated with a single media Representation as shown in Figure B.5.

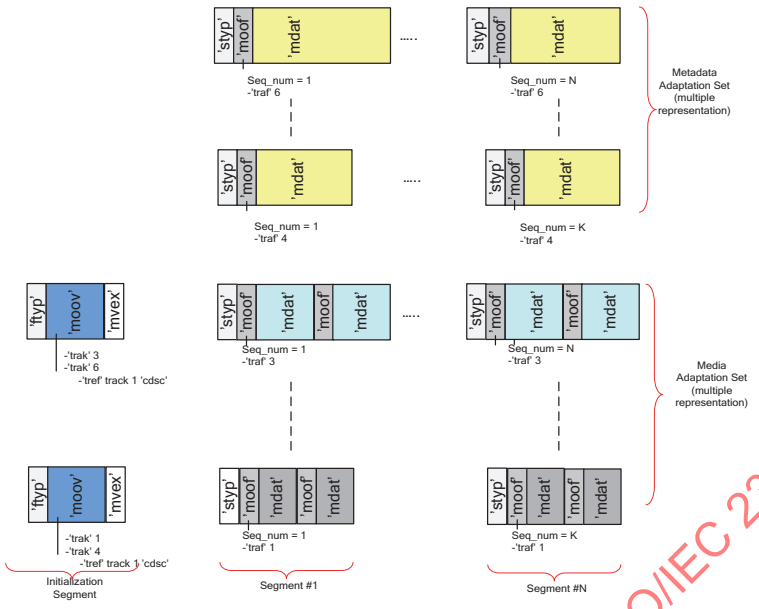

Figure B.5 — One metadata Representation for one media Representation