INTERNATIONAL STANDARD

ISO/IEC 23001-11

Third edition
2023-02

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 23001-11:2023(E)

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Page

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see https://patents.iec.ch).

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 29, *Coding of audio, picture, multimedia and hypermedia information*.

This third edition cancels and replaces the second edition (ISO/IEC 23001-11:2019), which has been technically revised.

The main changes are as follows:

— 6.2 related to complexity metrics for decoder-power reduction is amended by the specification of a new VVC SEI message carrying complexity metrics for decoder-power reduction.

— Clause 9 related to metrics for quality recovery after low-power encoding is amended by the specification of additional metrics for quality recovery after low-power encoding in the newly added VVC SEI message.

— 6.3 related to interactive signalling for remote decoder-power reduction is amended by adding new syntax elements allowing a finer control by decoder of the encoding operations.

A list of all parts in the ISO/IEC 23001 series can be found on the ISO and IEC websites.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

Introduction

This document 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.

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 quality metrics that are 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 DOR-Req parameters, 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.

Information technology — MPEG systems technologies —

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

1 Scope

This document specifies metadata for energy-efficient decoding, encoding, presentation, and selection of media.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

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

ISO/IEC 23008-2, *Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 2: High efficiency video coding*

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

ISO/IEC 23090-3, *Information technology — Coded representation of immersive media — Part 3: Versatile video coding*

3 Terms, definitions, symbols, abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 14496-10, ISO/IEC 23008-2, ISO/IEC 23009-1, ISO/IEC 23090-3 and the following apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp

— IEC Electropedia: available at https://www.electropedia.org/

3.1.1
alpha-point deblocking instance
APDI
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

3.1.2
deblocking filtering instance
single filtering operation that produces either a single, filtered output p' or a single, filtered output q', where p' and q' are filtered samples across an 8x8 and 4x4 block edge for HEVC and VVC, respectively

3.1.3
decoding process
process that reads a bitstream and derives decoded pictures from it

Note 1 to entry: The decoding process is specified in ISO/IEC 14496-10, ISO/IEC 23008-2 or ISO/IEC 23090-3.

3.1.4
encoding process
process that produces a bitstream

Note 1 to entry: The bitstream shall conform to ISO/IEC 14496-10, ISO/IEC 23008-2 or ISO/IEC 23090-3.

3.1.5
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)

3.1.6
non-zero block
block containing at least one non-zero transform coefficient

3.1.7
peak signal
maximum permissible RGB component in a reconstructed frame

Note 1 to entry: For *N*-bit video, peak signal is $(2^N - 1)$.

3.1.8
period
interval over which complexity-metrics metadata are applicable

3.1.9
pixel
smallest addressable element in an all-points addressable display device

3.1.10
reconstructed frames
frames obtained after applying RGB colour-space conversion and cropping to the specific decoded picture or pictures for which display power-reduction metadata are applicable

3.1.11
RGB component
single sample representing one of the three primary colours of the RGB colour space

3.1.13
six-tap filtering
STF
single application of the 6-tap filter to generate a single filtered sample for fractional positions using the samples at integer-sample positions

3.2 Symbols and abbreviated terms

3.2.1 Symbols

+	addition
-	subtraction (as a two-argument operator) or negation (as a unary prefix operator)
*	multiplication

/ integer division with truncation of the result toward zero. For example, 7 / 4 and −7 / −4 are truncated to 1 and −7 / 4 and 7 / −4 are truncated to −1

÷ division in mathematical equations where no truncation or rounding is intended

3.2.2 Abbreviations

APDI alpha-point deblocking instance

ASIC application specific integrated circuit

AVC advanced video coding – ISO/IEC 14496-10

BMFF base media file format

CM complexity metric

CMOS complementary metal oxide semiconductor

CMP cubemap projection format

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

ERP equi-rectangular projection format

Fps frames per second

FS fresh start

GP good picture

HEVC high efficiency video coding – ISO/IEC 23008-2

HCMP hemisphere cubemap projection format

Mbps mega bits per second

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
SSIM	structural similarity index measure
VVC	versatile video coding – ISO/IEC 23090-3
XSD	cross-segment decoding
wPSNR	weighted peak signal-to-noise ratio
WS-PSNR	weighted to spherically uniform peak signal-to-noise ratio

4 Conventions

4.1 Arithmetic operators

x^y	exponentiation
x/y	division where no truncation or rounding is intended
	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
$\sum_{p \text{ in } B}^{y} f(p)$	summation of $f(p)$ with p taking all integer location values in a block B in a picture
$x \,\%\, y$	Modulus. Remainder of x divided by y, defined only for integers x and y with $x >= 0$ and $y > 0$

4.2 Relational operators

>	greater than
>=	greater than or equal to
<	less than
<=	less than or equal to
= =	equal to
!=	not equal to

When a relational operator is applied to a syntax element or variable that has been assigned the value "na" (not applicable), the value "na" is treated as a distinct value for the syntax element or variable. The value "na" is considered not to be equal to any other value.

4.3 Bit-wise operators

x >> y	arithmetic right shift of a two's complement integer representation of x by y binary digits

This function is defined only for non-negative integer values of y. Bits shifted into the most significant bits (MSBs) as a result of the right shift have a value equal to the MSB of x prior to the shift operation.

x << y arithmetic left shift of a two's complement integer representation of x by y binary digits

This function is defined only for non-negative integer values of y. Bits shifted into the least significant bits (LSBs) as a result of the left shift have a value equal to 0.

4.4 Assignment operators

= assignment operator

++ increment, i.e., *x*++ is equivalent to *x* = *x* + 1; when used in an array index, evaluates to the value of the variable prior to the increment operation

– – decrement, i.e., *x*– – is equivalent to *x* = *x* – 1; when used in an array index, evaluates to the value of the variable prior to the decrement operation

+= increment by amount specified, i.e., *x* += 3 is equivalent to *x* = *x* + 3, and *x* += (–3) is equivalent to *x* = *x* + (–3)

–= decrement by amount specified, i.e., *x* –= 3 is equivalent to *x* = *x* – 3, and *x* –= (–3) is equivalent to *x* = *x* – (–3)

4.5 Range notation

x = *y*..*z* *x* takes on integer values starting from *y* to *z*, inclusive, with *x*, *y*, and *z* being integer numbers and *z* being greater than or equal to *y*

4.6 Mathematical functions

Mathematical functions are defined as follows:

$$\mathrm{Abs}(x)=\begin{cases}-x, & x<0\\ x, & x\geq 0\end{cases} \tag{4-1}$$

$$\mathrm{Clip}(x)=\begin{cases}x, & x<256\\ 255, & \text{otherwise}\end{cases} \tag{4-2}$$

$$\mathrm{Clip3}(x,y,z)=\begin{cases}x & ; \quad z<x\\ y & ; \quad z>y\\ z & ; \quad \text{otherwise}\end{cases} \tag{4-3}$$

Floor(*x*) is the greatest integer less than or equal to *x* (4-4)

Log10(*x*) returns the base-10 logarithm of *x* (4-5)

Round(*x*) = Sign(*x*) * Floor(Abs(*x*) + 0.5) (4-6)

$$\mathrm{Sign}(x)=\begin{cases}-1, & x<0\\ 1, & x\geq 0\end{cases} \tag{4-7}$$

x^y specifies *x* to the power of *y* (4-8)

power(*x*, *y*) specifies x to the power of *y* (4-9)

4.7 Specification of syntax functions and descriptors

The following function is used in the specification of the syntax:

read_bits(n) reads the next n bits from the bitstream and advances the bitstream pointer by n bit positions. When n is equal to 0, read_bits(n) is specified to return a value equal to 0 and to not advance the bitstream pointer.

The following descriptors specify the parsing process of each syntax element:

— u(n): unsigned integer using n bits. The parsing process for this descriptor is specified by the return value of the function read_bits(n) interpreted as a binary representation of an unsigned integer with most significant bit written first.

— s(n): signed integer using n bits. The parsing process for this descriptor is specified by the return value of the function read_bits(n) interpreted as a two's complement integer representation with most significant bit written first.

5 Functional architecture

5.1 Description of the functional architecture

Figure 1 shows the functional architecture utilizing green metadata. 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.

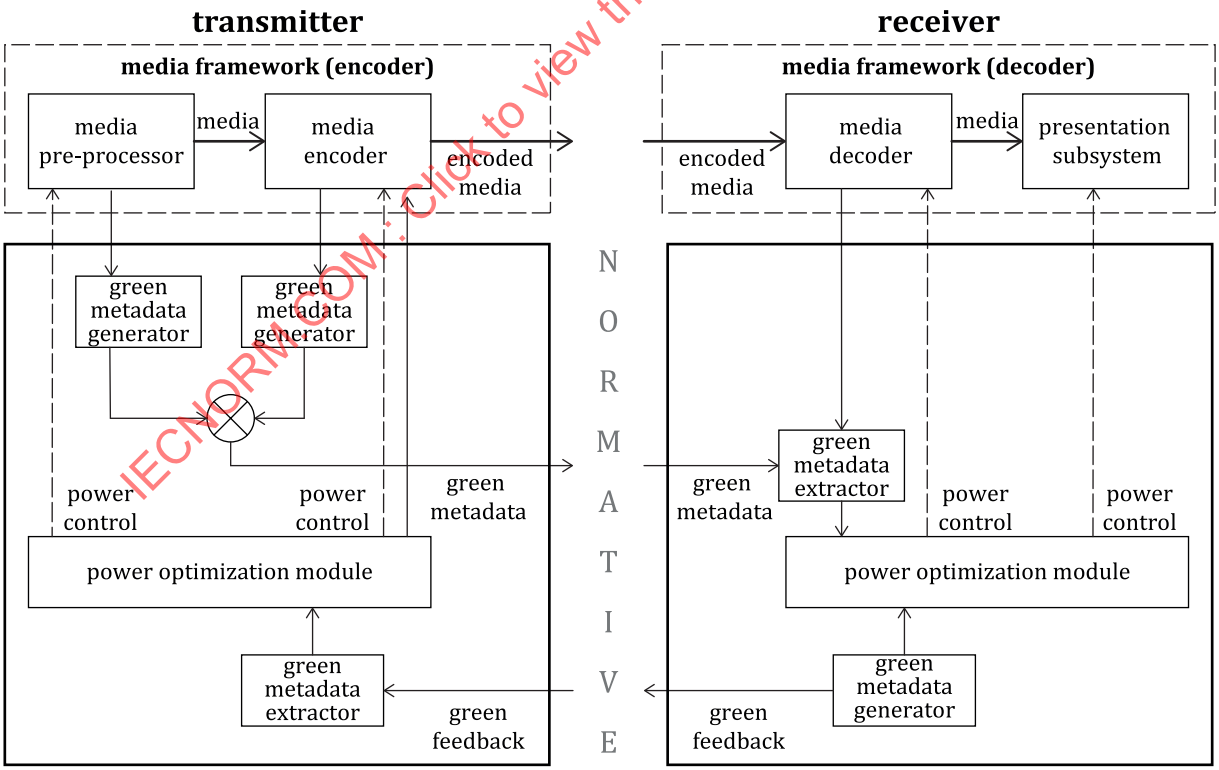

Figure 1 — Functional architecture

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 is 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 to reduce the presentation subsystem's power consumption when rendering the video. In addition, the power-optimization module can 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.

5.2 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.

6 Decoder power reduction

6.1 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.

6.2 Complexity metrics for decoder-power reduction

6.2.1 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, HEVC or VVC bitstream to the decoder.

6.2.2 Syntax

The syntax for the AVC CMs is described in Table 1.

Table 1 — Syntax for the AVC CMs

	Descriptor
period_type	u(8)
if ((period_type = = 2) || (period_type = = 7)) {	
num_seconds	u(16)
}	
else if ((period_type = = 3) || (period_type = = 8)) {	
num_pictures	u(16)
}	
if (period_type = = 8) {	
temporal_map	u(8)
for (t=0; t<8; t++) {	
if ((temporal_map>>t)&2 = = 1)	
num_pictures_in_temporal_layers[t]	u(16)
}	
}	
if (period_type <= 3) {	
portion_non_zero_8x8_blocks	u(8)
portion_intra_predicted_macroblocks	u(8)
portion_six_tap_filterings	u(8)
portion_alpha_point_deblocking_instances	u(8)
}	
else if (period_type = = 4) {	
for (i=0; i<= num_slice_groups_minus1; i++) {	
num_slices_minus1[i]	u(16)
}	
for (i=0; i<= num_slice_groups_minus1; i++) {	
for (j=0; j<=num_slices_minus1[i]; j++) {	
first_mb_in_slice[i][j]	u(16)
portion_non_zero_8x8_blocks[i][j]	u(8)

Table 1 *(continued)*

portion_intra_predicted_macroblocks[i][j]	u(8)
portion_six_tap_filterings[i][j]	u(8)
portion_alpha_point_deblocking_instances[i][j]	u(8)
}	
}	
}	
else if (period_type >= 5) && (period_type <= 8) {	
num_layers_minus1	u(16)
for (l=0; l<= num_layers_minus1; l++) {	
picture_parameter_set_id[l]	u(8)
priority_id[l]	u(6)
dependency_id[l]	u(3)
quality_id[l]	u(4)
temporal_id[l]	u(3)
portion_non_zero_8x8_blocks[l]	u(8)
portion_intra_predicted_macroblocks[l]	u(8)
portion_six_tap_filterings[l]	u(8)
portion_alpha_point_deblocking_instances[l]	u(8)
}	
}	

The syntax for the HEVC CMs is described in Table 2.

Table 2 — Syntax for the HEVC CMs

	Descriptor
period_type	u(8)
if (period_type = = 2) {	
num_seconds	u(16)
}	
else if (period_type = = 3) {	
num_pictures	u(16)
}	
if (period_type <= 3) {	
portion_non_zero_blocks_area	u(8)
if (portion_non_zero_blocks_area != 0) {	
portion_8x8_blocks_in_non_zero_area	u(8)
portion_16x16_blocks_in_non_zero_area	u(8)
portion_32x32_blocks_in_non_zero_area	u(8)
}	
portion_intra_predicted_blocks_area	u(8)
if (portion_intra_predicted_blocks_area = = 255) {	
portion_planar_blocks_in_intra_area	u(8)
portion_dc_blocks_in_intra_area	u(8)
portion_angular_hv_blocks_in_intra_area	u(8)
}	
else {	
portion_blocks_a_c_d_n_filterings	u(8)

Table 2 *(continued)*

portion_blocks_h_b_filterings	u(8)
portion_blocks_f_i_k_q_filterings	u(8)
portion_blocks_j_filterings	u(8)
portion_blocks_e_g_p_r_filterings	u(8)
}	
portion_deblocking_instances	u(8)
}	
else if (period_type = = 4) {	
max_num_slices_tiles_minus1	u(16)
for (t=0; t<=max_num_slices_tiles_minus1; t++) {	
first_ctb_in_slice_or_tile[t]	u(16)
portion_non_zero_blocks_area[t]	u(8)
if (portion_non_zero_blocks_area[t] != 0) {	
portion_8x8_blocks_in_non_zero_area[t]	u(8)
portion_16x16_blocks_in_non_zero_area[t]	u(8)
portion_32x32_blocks_in_non_zero_area[t]	u(8)
}	
portion_intra_predicted_blocks_area[t]	u(8)
if (portion_intra_predicted_blocks_area[t] = = 255) {	
portion_planar_blocks_in_intra_area[t]	u(8)
portion_dc_blocks_in_intra_area[t]	u(8)
portion_angular_hv_blocks_in_intra_area[t]	u(8)
}	
else {	
portion_blocks_a_c_d_n_filterings[t]	u(8)
portion_blocks_h_b_filterings[t]	u(8)
portion_blocks_f_i_k_q_filterings[t]	u(8)
portion_blocks_j_filterings[t]	u(8)
portion_blocks_e_g_p_r_filterings[t]	u(8)
}	
portion_deblocking_instances[t]	u(8)
}	
}	

The syntax for the VVC CMs is described in Table 3.

Table 3 — Syntax for the VVC CMs

	Descriptor
period_type	u(4)
granularity_type	u(3)
extended_representation_flag	u(1)
if (period_type = = 2) {	
num_seconds	u(16)
}	
else if (period_type = = 3) {	
num_pictures	u(16)
}	

Table 3 *(continued)*

if (granularity_type = = 0) {	
portion_non_zero_blocks_area	u(8)
portion_non_zero_transform_coefficients_area	u(8)
portion_intra_predicted_blocks_area	u(8)
portion_deblocking_instances	u(8)
portion_alf_filtered_blocks	u(8)
if (extended_representation_flag) {	
if (portion_non_zero_blocks_area != 0) {	
portion_non_zero_4_8_16_blocks_area	u(8)
portion_non_zero_32_64_128_blocks_area	u(8)
portion_non_zero_256_512_1024_blocks_area	u(8)
portion_non_zero_2048_4096_blocks_area	u(8)
}	
if (portion_intra_predicted_blocks_area < 255) {	
portion_bi_and_gpm_predicted_blocks_area	u(8)
portion_bdof_blocks_area	u(8)
}	
portion_sao_filtered_blocks	u(8)
}	
}	
else if (granularity_type <= 3) {	
max_num_segments_minus1	u(16)
for (t=0; t<=max_num_segments_minus1; t++) {	
segment_address[t]	u(16)
portion_non_zero_blocks_area[t]	u(8)
portion_non_zero_transform_coefficients_area[t]	u(8)
portion_intra_predicted_blocks_area[t]	u(8)
portion_deblocking_instances[t]	u(8)
portion_alf_filtered_blocks[t]	u(8)
if (extended_representation_flag) {	
if (portion_non_zero_blocks_area[t] != 0) {	
portion_non_zero_4_8_16_blocks_area[t]	u(8)
portion_non_zero_32_64_128_blocks_area[t]	u(8)
portion_non_zero_256_512_1024_blocks_area[t]	u(8)
portion_non_zero_2048_4096_blocks_area[t]	u(8)
}	
if (portion_intra_predicted_blocks_area[t] < 255) {	
portion_bi_and_gpm_predicted_blocks_area[t]	u(8)
portion_bdof_blocks_area[t]	u(8)
}	
portion_sao_filtered_blocks[t]	u(8)
}	
}	
}	

6.2.3 Signalling

SEI messages can be used to signal green metadata in an AVC, HEVC or VVC stream. The green metadata SEI message payload type is specified in ISO/IEC 14496-10, ISO/IEC 23008-2, and ISO/IEC 23090-3. 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 is 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 is 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 is transmitted with the first picture in the time interval or with the first picture in the specified number of pictures.

6.2.4 Semantics

6.2.4.1 AVC semantics

The semantics of various terms are defined below.

period_type specifies the type of upcoming period over which the complexity metrics are applicable and is defined in the Table 4.

Table 4 — specification of period_type for AVC

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	complexity metrics are applicable to a single picture with slice granularity
0x05	complexity metrics are applicable to a single picture with scalable layer granularity
0x06	complexity metrics are applicable to all pictures in decoding order, up to (but not including) the picture containing the next I slice in the base layer with scalable layer granularity
0x07	complexity metrics are applicable over a specified time interval in seconds with scalable layer granularity
0x08	complexity metrics are applicable over a specified number of pictures counted in decoding order with scalable layer granularity
0x09–0xFF	user-defined

num_seconds indicates the number of seconds over which the complexity metrics are applicable when period_type is 2 or 7.

num_pictures indicates the number of pictures, counted in decoding order, over which the complexity metrics are applicable when period_type is 3 or 8. When period_type is 8, this is a default number of pictures for each temporal layer, which can be overridden using temporal_map flags.

$N_{\text{picsInPeriod}}$ specifies the number of pictures in the specified period. When period_type is 0 or 4, then $N_{\text{picsInPeriod}}$ is 1. When period_type is 1, then $N_{\text{picsInPeriod}}$ 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 $N_{\text{picsInPeriod}}$ is determined from the frame rate. When period_type is 3, then $N_{\text{picsInPeriod}}$ is equal to num_pictures.

$N_{mbsInPeriod}$ specifies the total number of macroblocks that are coded in the specified period. It is determined by the following computation:

$$N_{mbsInPeriod} = \sum_{n=1}^{N_{picsInPeriod}} N_{mbsInPic}(n) \quad (6\text{-}1)$$

where $N_{mbsInPic}(n)$ is set to the value of the AVC variable PicSizeInMbs for the n^{th} picture within the specified period, where $1 <= n <= N_{picsInPeriod}$.

temporal_map indicates which temporal layer has a different number of pictures from num_pictures in the specified period, when period_type is 8.

num_pictures_in_temporal_layers[t] indicates the number of pictures in the specified period for the t^{th} temporal layer when period_type is 8. When not present, it is equal to num_pictures.

$N_{picsInPeriodForTempLayer}[t]$ specifies the number of pictures in the specified period for the t^{th} temporal layer. When period_type is 5 then $N_{picsInPeriodForTempLayer}[t]$ is 1. When period_type is 6, then $N_{picsInPeriodForTempLayer}[t]$ is determined by counting the pictures associated to the t^{th} temporal layer in decoding order up to (but not including) the one containing the next I slice. When period_type is 7, then $N_{picsInPeriodForTempLayer}[t]$ is determined from the frame rate associated to the t^{th} temporal layer. When period_type is 8, then $N_{picsInPeriodForTempLayer}[t]$ is equal to num_pictures_in_temporal_layers[t].

portion_non_zero_8x8_blocks indicates the portion of 8x8 blocks with non-zero transform coefficients values in the specified period and is set equal to $P_{nonZero8x8Blks}$ defined as follows:

$$P_{nonZero8x8Blks} = \text{Floor}\left(\frac{N_{nonZero8x8Blks}}{4 * N_{mbsInPeriod}} * 255 \right) \quad (6\text{-}2)$$

where $N_{nonZero8x8Blks}$ is the number of 8x8 blocks with non-zero transform coefficients values in the specified period. $N_{nonZero8x8Blks}$ is derived from portion_non_zero_8x8_blocks and $N_{mbsInPeriod}$ in the decoder.

portion_intra_predicted_macroblocks indicates the portion of intra-predicted macroblocks in the specified period and is set equal to $P_{intraMbs}$ defined as follows:

$$P_{intraMbs} = \text{Floor}\left(\frac{N_{intraMbs}}{N_{mbsInPeriod}} * 255 \right) \quad (6\text{-}3)$$

where $N_{intraMbs}$ is the number of intra-predicted macroblocks in the specified period. $N_{intraMbs}$ is derived from portion_intra_predicted_macroblocks and $N_{mbsInPeriod}$ in the decoder.

portion_six_tap_filterings indicates the portion of 6-tap filterings (STFs) in the specified period and is set equal to $P_{sixTapFilt}$ defined as follows:

$$P_{sixTapFilt} = \text{Floor}\left(\frac{N_{sixTapFilt}}{N_{maxSixTapFiltInPeriod}} * 255 \right) \quad (6\text{-}4)$$

where $N_{maxSixTapFiltInPeriod}$ is the maximum number of STFs that can occur within the specified period and is derived from $N_{mbsInPeriod}$ variable as

$$N_{maxSixTapFiltInPeriod} = (1664 * N_{mbsInPeriod}) \quad (6\text{-}5)$$

and $N_{sixTapFilt}$ is the number of 6-tap filterings (STFs) within the specified period. Guidance for the counting of $N_{sixTapFilt}$ can be found in Annex B. $N_{sixTapFilt}$ is derived from portion_six_tap_filterings and $N_{maxSixTapFiltInPeriod}$ in the decoder.

portion_alpha_point_deblocking_instances indicates the portion of alpha-point deblocking instances (APDIs) in the specified period and is set equal to $P_{alphaPtDbfs}$ defined as follows:

$$P_{alphaPtDbfs} = \text{Floor}\left(\frac{N_{alphaPtDbfs}}{N_{maxAlphaPtDbfsInPeriod}} * 255 \right) \quad (6\text{-}6)$$

$N_{maxAlphaPtDbfsInPeriod}$ is the maximum number of APDIs that can occur within the specified period and is derived from $N_{mbsInPeriod}$ and $S_{chrMult}$ variables as

$$N_{maxAlphaPtDbfsInPeriod} = 128 * S_{chrMult} * N_{mbsInPeriod} \quad (6\text{-}7)$$

$S_{chrMult}$ depends on the AVC variables separate_colour_plane_flag and chroma_format_idc as shown in the Table 5.

Table 5 — specification of $S_{chrMult}$ for AVC

$S_{chrMult}$	separate_colour_plane_flag	chroma_format_idc	Comment
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
3	1	any value	separate colour plane

$N_{alphaPtDbfs}$ is the number of 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. $N_{alphaPtDbfs}$ is derived from portion_alpha_point_deblocking_instances and $N_{maxAlphaPtDbfsInPeriod}$ in the decoder.

num_slices_minus1 plus 1 indicates the number of slices per slice_group in the picture.

first_mb_in_slice[i][j] indicates the first macroblock number in the slice[i][j].

$N_{mbsInSlice}$[i][j] is the total number of macroblocks that are coded in the slice[i][j] and is determined by the following computation:

— if num_slice_groups_minus1 is equal to 0, the following process described in pseudo-code applies.

```
if (j<num_slices_minus1[0])
  NmbsInSlice[0][j] = first_mb_in_slice[0][j+1] - first_mb_in_slice[0][j]
else
  NmbsInSlice[0][j] = PicSizeInMbs - first_mb_in_slice[0][j]
```
(6-8)

— otherwise (num_slice_groups_minus1 is not equal to 0), and after derivation of the macroblock to slice group map (MbToSliceGroupMap) as specified in ISO/IEC 14496-10:2022, 8.2.2.8, the following process described in pseudo-code applies:

```
k=0;
if (j<num_slices_minus1[i])
  for ( n=first_mb_in_slice[i][j]; n<first_mb_in_slice[i][j+1]; n++ )
    if ( MbToSliceGroupMap[first_mb_in_slice[i][j]] = = MbToSliceGroupMap[n] )
      k++;
  NmbsInSlice[i][j] = k;
else
  for ( n=first_mb_in_slice[i][j]; n<PicSizeInMbs; n++ )
    if ( MbToSliceGroupMap[first_mb_in_slice[i][j]] = = MbToSliceGroupMap[n] )
      k++;
```

$N_{mbsInSlice}$[i][j] = k; (6-9)

portion_non_zero_8x8_blocks[i][j] indicates the portion of 8x8 blocks with non-zero transform coefficients values in the slice[i][j] and is set equal to $P_{nonZero8x8Blks}$[i][j] defined as follows:

$$P_{nonZero8x8Blks}[\,i\,][\,j\,] = \mathrm{Floor}\left(\frac{N_{nonZero8x8Blks}[\,i\,][\,j\,]}{4 * N_{mbsInSlice}[\,i\,][\,j\,]} * 255\right) \quad (6\text{-}10)$$

where $N_{nonZero8x8Blks}$[i][j] is the number of 8x8 blocks with non-zero transform coefficients values in the slice[i][j]. $N_{nonZero8x8Blks}$[i][j] is derived from portion_non_zero_8x8_blocks[i][j] and $N_{mbsInSlice}$[i][j] in the decoder.

portion_intra_predicted_macroblocks[i][j] indicates the portion of macroblocks using intra prediction modes in the slice[i][j] and is set equal to $P_{intraMbs}$[i][j] defined as follows:

$$P_{intraMbs}[\,i\,][\,j\,] = \mathrm{Floor}\left(\frac{N_{intraMbs}[\,i\,][\,j\,]}{N_{mbsInSlice}[\,i\,][\,j\,]} * 255\right) \quad (6\text{-}11)$$

where $N_{intraMbs}$[i][j] is the number of macroblocks using intra prediction modes in the slice[i][j]. $N_{intraMbs}$[i][j] is derived from portion_intra_predicted_macroblocks[i][j] and $N_{mbsInSlice}$[i][j] in the decoder.

portion_six_tap_filterings[i][j] indicates the portion of 6-tap filterings (STFs) in the specified slice[i][j] and is set equal to $P_{sixTapFilt}$[i][j] defined as follows:

$$P_{sixTapFilt}[\,i\,][\,j\,] = \mathrm{Floor}\left(\frac{N_{sixTapFilt}[\,i\,][\,j\,]}{N_{maxSixTapFiltInSlice}[\,i\,][\,j\,]} * 255\right) \quad (6\text{-}12)$$

where $N_{maxSixTapFiltInSlice}$[i][j] is the maximum number of STFs that can occur in the slice[i][j] and is derived from $N_{mbsInSlice}$[i][j] variable as

$$N_{maxSixTapFiltInSlice}[\,i\,][\,j\,] = 1664 * N_{mbsInSlice}[\,i\,][\,j\,] \quad (6\text{-}13)$$

and $N_{sixTapFilt}$[i][j] is the number of 6-tap filterings (STFs) within the slice[i][j]. Guidance for the counting of $N_{sixTapFilt}$[i][j] can be found in Annex B. $N_{sixTapFilt}$[i][j] is derived from portion_six_tap_filterings[i][j] and $N_{maxSixTapFiltInSlice}$[i][j] in the decoder.

portion_alpha_point_deblocking_instances[i][j] indicates the portion of alpha-point deblocking instances (APDIs) in the specified slice[i][j] and is set equal to $P_{alphaPtDbfs}$[i][j] defined as follows:

$$P_{alphaPtDbfs}[\,i\,][\,j\,] = \mathrm{Floor}\left(\frac{N_{alphaPtDbfs}[\,i\,][\,j\,]}{N_{maxAlphaPtDbfsInSlice}[\,i\,][\,j\,]} * 255\right) \quad (6\text{-}14)$$

where $N_{maxAlphaPtDbfsInSlice}$[i][j] is the maximum number of APDIs that can occur in the slice[i][j] and is derived from $N_{mbsInSlice}$[i][j] and $S_{chrMult}$ variables as

$$N_{maxAlphaPtDbfsInSlice}[\,i\,][\,j\,] = 128 * S_{chrMult} * N_{mbsInSlice}[\,i\,][\,j\,] \quad (6\text{-}15)$$

and $N_{alphaPtDbfs}$[i][j] is the number of alpha-point deblocking instances (APDIs) in slice[i][j]. $N_{alphaPtDbfs}$[i][j] is derived from portion_alpha_point_deblocking_instances[i][j] and $N_{maxAlphaPtDbfsInSlice}$[i][j] in the decoder.

num_layers_minus1 plus 1 indicates the number of scalable layers in the associated picture or in the specified period.

pic_parameter_set_id[l] indicates the picture parameter set in use for the l^{th} scalable layer. The value of pic_parameter_set_id[l] shall be in the range of 0 to 255, inclusive (as specified in ISO/IEC 14496-10:2022, 7.4.3).

priority_id[l] indicates a priority identifier for the NAL unit in the l^{th} scalable layer. The value of priority_id[l] shall be in the range of 0 to 63, inclusive (as specified in ISO/IEC 14496-10:2022, G.7.3.1.1).

dependency_id[l] indicates a dependency identifier for the NAL unit in the l^{th} scalable layer. The value of dependency_id[l] shall be in the range of 0 to 7, inclusive (as specified in ISO/IEC 14496-10:2022, G.7.3.1.1).

quality_id[l] indicates a quality identifier for the NAL unit in the l^{th} scalable layer. The value of quality_id[l] shall be in the range of 0 to 15, inclusive (as specified in ISO/IEC 14496-10:2022, G.7.3.1.1).

temporal_id[l] indicates a temporal identifier for the NAL unit in the l^{th} scalable layer. The value of temporal_id[l] shall be in the range of 0 to 7, inclusive (as specified in ISO/IEC 14496-10:2022, G.7.3.1.1).

portion_non_zero_8x8_blocks[l] indicates the portion of 8x8 blocks with non-zero transform coefficients values in the l^{th} scalable layer and is set equal to $P_{nonZero8x8Blks}[i][j]$ defined as follows:

$$P_{nonZero8x8Blks}[l] = Floor\left(\frac{N_{nonZero8x8Blks}[l]}{4 * N_{mbsInLayerInPeriod}[l]} * 255 \right) \quad (6\text{-}16)$$

$N_{mbsInLayerInPeriod}[l]$ is the total number of macroblocks in the l^{th} scalable layer in the specified period and is derived from $N_{mbsInLayerInPeriod}[l]$ and $N_{picsInPeriodForTempLayer}[$ temporal_id[l]] as

$$N_{mbsInLayerInPeriod}[l] = N_{mbsInLayer}[l] * N_{picsInPeriodForTempLayer}[temporal\_id[l]] \quad (6\text{-}17)$$

where $N_{mbsInLayer}[l]$ is the total number of macroblocks in the l^{th} scalable layer and determined after derivation of the number of macroblocks associated with pic_parameter_set_id[l], as specified in ISO/IEC 14496-10:2022, 7.4.3.

$N_{nonZero8x8Blks}[l]$ is the number of 8x8 blocks with non-zero transform coefficients values in the l^{th} scalable layer in the specified period. It is derived from portion_non_zero_8x8_blocks[l] and $N_{mbsInLayerInPeriod}[l]$ in the decoder.

portion_intra_predicted_macroblocks[l] indicates the portion of macroblocks using intra prediction modes in the l^{th} scalable layer and is set equal to $P_{intraMbs}[i][j]$ defined as follows:

$$P_{intraMbs}[l] = Floor\left(\frac{N_{intraMbs}[l]}{N_{mbsInLayerInPeriod}[l]} * 255 \right) \quad (6\text{-}18)$$

$N_{intraMbs}[l]$ is the number of macroblocks using intra prediction modes in the l^{th} scalable layer in the specified period. It is derived from portion_intra_predicted_macroblocks[l] and $N_{mbsInLayerInPeriod}[l]$ in the decoder.

portion_six_tap_filterings[l] indicates the portion of 6-tap filterings (STFs) in the specified l^{th} scalable layer in the specified period and is set equal to $P_{sixTapFilt}[i][j]$ defined as follows:

$$P_{sixTapFilt}[l] = Floor\left(\frac{N_{sixTapFilter}[l]}{N_{maxSixTapFiltInLayerInPeriod}[l]} * 255 \right) \quad (6\text{-}19)$$

$N_{maxSixTapFiltInLayerInPeriod}[l]$ is the maximum number of STFs that can occur in the l^{th} scalable layer in the specified period and is derived from $N_{mbsInLayerInPeriod}[l]$ variable as

$$N_{maxSixTapFiltInLayerInPeriod}[l] = 1664 * N_{mbsInLayerInPeriod}[l] \quad (6\text{-}20)$$

$N_{sixTapFilt}[l]$ is the number of 6-tap filterings (STFs) within the l^{th} scalable layer in the specified period. Guidance for the counting of $N_{sixTapFilt}[l]$ can be found in Annex B. It is derived from portion_six_tap_filterings[l] and $N_{maxSixTapFiltInLayerInPeriod}[l]$ in the decoder.

portion_alpha_point_deblocking_instances[l] indicates the portion of alpha-point deblocking instances (APDIs) in the specified l^{th} scalable layer in the specified period and is set equal to $P_{alphaPtDbfs}[\,i\,][\,j\,]$ defined as follows:

$$P_{alphaPtDbfs}[\,l\,] = \text{Floor}\left(\frac{N_{alphaPtDbfs}[\,l\,]}{N_{maxAlphaPtDbfsInLayerInPeriod}[\,l\,]} * 255\right) \tag{6-21}$$

$N_{maxAlphaPtDbfsInLayerInPeriod}[\,l\,]$ is the maximum number of APDIs that can occur in the l^{th} scalable layer in the specified period and is derived from $N_{mbsInLayerInPeriod}[\,l\,]$ and $S_{chrMult}$ variables as

$$N_{maxAlphaPtDbfsInLayerInPeriod}[\,l\,] = 128 * S_{chrMult} * N_{mbsInLayerInPeriod}[\,l\,] \tag{6-22}$$

$N_{alphaPtDbfs}[\,l\,]$ is the number of alpha-point deblocking instances (APDIs) in the l^{th} scalable layer in the specified period. It is derived from portion_alpha_point_deblocking_instances[l] and $N_{maxAlphaPtDbfsInLayerInPeriod}[\,l\,]$ in the decoder.

6.2.4.2 HEVC semantics

The semantics of various terms are defined below.

period_type specifies the type of upcoming period over which the complexity metrics are applicable and is defined in the Table 6.

Table 6 — specification of period_type for HEVC

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	complexity metrics are applicable to a single picture with slice or tile granularity
0x05-0xFF	reserved

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

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

$N_{picsInPeriod}$ is the number of pictures in the specified period. When period_type is 0, then $N_{picsInPeriod}$ is 1. When period_type is 1, then $N_{picsInPeriod}$ 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 $N_{picsInPeriod}$ is determined from the frame rate. When period_type is 3, then $N_{picsInPeriod}$ is equal to num_pictures.

$N_{4x4BlksInPeriod}$ is the total number of 4x4 blocks that are coded in the specified period.

It is determined by the following computation:

$$N_{4x4BlksInPeriod} = \sum_{n=1}^{N_{picsInPeriod}} N_{4x4BlksPic}(n) \tag{6-23}$$

where $N_{4x4BlksPic}(n)$ is derived for the n^{th} picture within the specified period, with $1 <= n <= N_{picsInPeriod}$, from HEVC variables CtbLog2SizeY and PicSizeInCtbsY as follows:

$$N_{4x4BlksPic}(n) = S_{picInCtb} * N_{ctbs} \tag{6-24}$$

where

— N_{ctbs} is set equal to (1 << (CtbLog2SizeY – 2))^2

— $S_{picInCtb}$ is set equal to PicSizeInCtbsY.

portion_non_zero_blocks_area indicates the portion of area covered by blocks with non-zero transform coefficients values, in the pictures of the specified period, using a 4x4 blocks granularity and is set equal to $P_{nonZeroBlksArea}$ defined as follows:

$$P_{nonZeroBlksArea} = Floor\left(\frac{N_{nonZeroBlks}}{N_{4x4BlksInPeriod}} * 255 \right) \quad (6\text{-}25)$$

where $N_{nonZeroBlks}$ is the number of blocks with non-zero transform coefficients values in the specified period using 4x4 granularity. At the encoder side, $N_{nonZeroBlks}$ is computed as follows:

$$N_{nonZeroBlks} = N_{nonZero4x4Blks} + 4*N_{nonZero8x8Blks} + 16*N_{nonZero16x16Blks} + 64*N_{nonZero32x32Blks} \quad (6\text{-}26)$$

where $N_{nonZero4x4Blks}$, $N_{nonZero8x8Blks}$, $N_{nonZero16x16Blks}$, $N_{nonZero32x32Blks}$ are the number of 4x4, 8x8, 16x16 and 32x32 blocks with non-zero transform coefficients values, respectively, in the specified period.

$N_{nonZeroBlks}$ is derived from portion_non_zero_blocks_area and $N_{4x4BlksInPeriod}$ in the decoder.

portion_8x8_blocks_in_non_zero_area indicates the portion of 8x8 blocks area in the non-zero area in the specified period and is set equal to $P_{nonZero8x8Blks}$ defined as follows:

$$P_{nonZero8x8Blks} = Floor\left(\frac{4 * N_{nonZero8x8Blks}}{N_{nonZeroBlks}} * 255 \right) \quad (6\text{-}27)$$

When not present, it is set equal to 0.

$N_{nonZero8x8Blks}$ is the number of 8x8 blocks with non-zero transform coefficients values in the specified period. It is derived from portion_8x8_blocks_in_non_zero_area and $N_{nonZeroBlks}$ in the decoder.

portion_16x16_blocks_in_non_zero_area indicates the portion of 16x16 blocks area in the non-zero area in the specified period and is set equal to $P_{nonZero16x16Blks}$ defined as follows:

$$P_{nonZero16x16Blks} = Floor\left(\frac{16 * N_{nonZero16x16Blks}}{N_{nonZeroBlks}} * 255 \right) \quad (6\text{-}28)$$

When not present, is equal to 0.

$N_{nonZero16x16Blks}$ is the number of 16x16 blocks with non-zero transform coefficients values in the specified period. It is derived from portion_16x16_blocks_in_non_zero_area and $N_{nonZeroBlks}$ in the decoder.

portion_32x32_blocks_in_non_zero_area indicates the portion of 32x32 blocks area in the non- zero area in the specified period and is set equal to $P_{nonZero32x32Blks}$ defined as follows:

$$P_{nonZero32x32Blks} = Floor\left(\frac{64 * N_{nonZero32x32Blks}}{N_{nonZeroBlks}} * 255 \right) \quad (6\text{-}29)$$

When not present, it is set equal to 0.

$N_{nonZero32x32Blks}$ is the number of 32x32 blocks with non-zero transform coefficients values in the specified period. It is derived from portion_32x32_blocks_in_non_zero_area and $N_{nonZeroBlks}$ in the decoder.

$N_{nonZero4x4Blks}$ is the number of 4x4 blocks with non-zero transform coefficients values in the specified period. $N_{nonZero4x4Blks}$ is derived from $N_{nonZeroBlks}$, $N_{nonZero8x8Blks}$, $N_{nonZero16x16Blks}$ and $N_{nonZero32x32Blks}$ as follows in the decoder:

$$N_{nonZero4x4Blks} = N_{nonZeroBlks} - 4*N_{nonZero8x8Blks} - 16*N_{nonZero16x16Blks} - 64*N_{nonZero32x32Blks} \quad (6\text{-}30)$$

portion_intra_predicted_blocks_area indicates the portion of area covered by intra predicted blocks in the pictures of the specified period using 4x4 granularity and is set equal to $P_{intraBlks}$ defined as follows:

$$P_{intraBlks} = \text{Floor}\left(\frac{4 * N_{intraBlks}}{N_{4x4BlksInPeriod}} * 255\right) \quad (6\text{-}31)$$

$N_{intraBlks}$ is the number of intra predicted blocks in the specified period using 8x8 granularity. At the encoder side, it is computed as follows:

$$N_{intraBlks} = N_{intra8x8Blks} + 4*N_{intra16x16Blks} + 16*N_{intra32x32Blks} + 64*N_{intra64x64Blks} \quad (6\text{-}32)$$

where $N_{intra8x8Blks}$, $N_{intra16x16Blks}$, $N_{intra32x32Blks}$ and $N_{intra64x64Blks}$ are the number of intra predicted 8x8, 16x16, 32x32 and 64x64 blocks respectively, in the specified period.

$N_{intraBlks}$ is derived from portion_intra_predicted_blocks_area and $N_{4x4BlksInPeriod}$ in the decoder.

portion_planar_blocks_in _intra_area indicates the portion of planar blocks area in the intra predicted area in the specified period and is set equal to $P_{planarBlksInIntra}$ defined as follows:

$$P_{planarBlksInIntra} = \text{Floor}\left(\frac{N_{planarBlks}}{4 * N_{intraBlks}} * 255\right) \quad (6\text{-}33)$$

When not present, it is set equal to 0.

$N_{planarBlks}$ is the number of intra planar predicted blocks in the specified period using 4x4 granularity. At the encoder side, it is computed as follows:

$$N_{planarBlks} = N_{planar4x4Blks} + 4*N_{planar8x8Blks} + 16*N_{planar16x16Blks} + 64*N_{planar32x32Blks} + 256*N_{planar64x64Blks} \quad (6\text{-}34)$$

where $N_{planar4x4Blks}$, $N_{planar8x8Blks}$, $N_{planar16x16Blks}$, $N_{planar32x32Blks}$ and $N_{planar64x64Blks}$ are the number of intra planar predicted 4x4, 8x8, 16x16, 32x32 and 64x64 blocks respectively, in the specified period.

$N_{planarBlks}$ is derived from portion_planar_blocks_in_intra_area and $N_{intraBlks}$ in the decoder.

portion_dc_blocks_in_intra_area indicates the portion of DC blocks area in the intra predicted area in the specified period and is set equal to $P_{DCBlksInIntra}$ defined as follows:

$$P_{DCBlksInIntra} = \text{Floor}\left(\frac{N_{DCBlks}}{4 * N_{intraBlks}} * 255\right) \quad (6\text{-}35)$$

When not present, it is set equal to 0.

N_{DCBlks} is the number of intra DC predicted blocks in the specified period using 4x4 granularity. At the encoder side, it is computed as follows:

$$N_{DCBlks} = N_{DC4x4Blks} + 4*N_{DC8x8Blks} + 16*N_{DC16x16Blks} + 64*N_{DC32x32Blks} + 256*N_{DC64x64Blks} \quad (6\text{-}36)$$

where $N_{DC4x4Blks}$, $N_{DC8x8Blks}$, $N_{DC16x16Blks}$, $N_{DC32x32Blks}$ and $N_{DC64x64Blks}$ are the number of intra DC predicted 4x4, 8x8, 16x16, 32x32 and 64x64 blocks respectively, in the specified period.

N_{DCBlks} is derived from portion_dc_blocks_in_intra_area and $N_{intraBlks}$ in the decoder.

portion_angular_hv_blocks_in_intra_area indicates the portion of angular horizontal or vertical blocks area in the intra predicted area in the specified period and is set equal to $P_{HVBlksInIntra}$ defined as follows:

$$P_{HVBlksInIntra} = \text{Floor}\left(\frac{N_{HVBlks}}{4 * N_{intraBlks}} * 255 \right) \tag{6-37}$$

When not present, it is set equal to 0.

$N_{angularHVBlks}$ is the number of intra angular horizontally or vertically predicted blocks in the specified period using 4x4 granularity. At the encoder side, it is computed as follows:

$$\begin{aligned} N_{angularHVBlks} = {} & N_{angularHV4x4Blks} + 4*N_{angularHV8x8Blks} + 16*N_{angularHV16x16Blks} \\ & + 64*N_{angularHV32x32Blks} + 256*N_{angularHV64x64Blks} \end{aligned} \tag{6-38}$$

where $N_{angularHV4x4Blks}$, $N_{angularHV8x8Blks}$, $N_{angularHV16x16Blks}$, $N_{angularHV32x32Blks}$ and $N_{angularHV64x64Blks}$ are the number of intra angular horizontally or vertically predicted 4x4, 8x8, 16x16, 32x32 and 64x64 blocks respectively, in the specified period.

$N_{angularHVBlks}$ is derived from portion_angular_hv_blocks_in_intra_area and $N_{intraBlks}$ in the decoder.

portion_blocks_a_c_d_n_filterings indicates the portion of prediction blocks whose luma samples position are located in sub-sample position a, c, d or n, as defined in Annex B, in the specified period and is set equal to $P_{acdnFiltBlks}$ defined as follows:

$$P_{acdnFiltBlks} = \text{Floor}\left(\frac{N_{acdnFiltBlks}}{N_{4x4BlksInPeriod}} * 255 \right) \tag{6-39}$$

When not present, it is set equal to 0.

$N_{acdnFiltBlks}$ is the number of prediction blocks whose luma samples position are located in sub-sample position a, c, d or n, as defined in Annex B, in the specified period. It is derived from portion_blocks_a_c_d_n_filterings and $N_{4x4BlksInPeriod}$ in the decoder.

portion_blocks_h_b_filterings indicates the portion of prediction blocks whose luma samples position are located in sub-sample position h or b, as defined in Annex B, in the specified period and is set equal to $P_{hbFiltBlks}$ defined as follows:

$$P_{hbFiltBlks} = \text{Floor}\left(\frac{N_{hbFiltBlks}}{N_{4x4BlksInPeriod}} * 255 \right) \tag{6-40}$$

When not present, it is set equal to 0.

$N_{hbFiltBlks}$ is the number of prediction blocks whose luma samples position are located in sub-sample position h or b, as defined in Annex B, in the specified period.

It is derived from portion_blocks_h_b_filterings and $N_{4x4BlksInPeriod}$ in the decoder.

portion_blocks_f_i_k_q_filterings indicates the portion of prediction blocks whose luma samples position are located in sub-sample position f, i, k or q, as defined in Annex B, in the specified period and is set equal to $P_{fikqFiltBlks}$ defined as follows:

$$P_{fikqFilt} = \text{Floor}\left(\frac{N_{fikqFiltBlks}}{N_{4x4BlksInPeriod}} * 255 \right) \tag{6-41}$$

When not present, it is set equal to 0.

$N_{fikqFiltBlks}$ is the number of prediction blocks whose luma samples position are located in sub-sample position f, i, k or q as defined in Annex B, in the specified period.

It is derived from portion_blocks_f_i_k_q_filterings and $N_{4x4BlksInPeriod}$ in the decoder.

portion_blocks_j_filterings indicates the portion of prediction blocks whose luma samples position are located in sub-sample position j, as defined in Annex B, in the specified period and is set equal to $P_{jFiltBlks}$ defined as follows:

$$P_{jFiltBlks} = \text{Floor}\left(\frac{N_{jFiltBlks}}{N_{4x4BlksInPeriod}} * 255 \right) \tag{6-42}$$

When not present, it is set equal to 0.

$N_{jFiltBlks}$ is the number of prediction blocks whose luma samples position are located in sub-sample position j, as defined in Annex B, in the specified period.

It is derived from portion_blocks_j_filterings and $N_{4x4BlksInPeriod}$ in the decoder.

portion_blocks_e_g_p_r_filterings indicates the portion of prediction blocks whose luma blocks position are located in sub-sample position e, g, p or r, as defined in Annex B, in the specified period and is set equal to $P_{egprFiltBlks}$ defined as follows:

$$P_{egprFiltBlks} = \text{Floor}\left(\frac{N_{egprFiltBlks}}{N_{4x4BlksInPeriod}} * 255 \right) \tag{6-43}$$

When not present, it is set equal to 0.

$N_{egprFiltBlks}$ is the number of prediction blocks whose luma samples position are located in sub-sample position e, g, p or r, as defined in Annex B, in the specified period.

It is derived from portion_blocks_e_g_p_r_filterings and $N_{4x4BlksInPeriod}$ in the decoder.

portion_deblocking_instances indicates the portion of deblocking filtering instances in the specified period and is set equal to $P_{dbfInstances}$ defined as follows:

$$P_{dbfInstances} = \text{Floor}\left(\frac{N_{dbfInstances}}{4 * S_{chrMult} * N_{4x4BlksInPeriod}} * 255 \right) \tag{6-44}$$

$S_{chrMult}$ depends on the HEVC variables separate_colour_plane_flag and chroma_format_idc as shown in the Table 7.

Table 7 — specification of $S_{chrMult}$ for HEVC

$S_{chrMult}$	separate_colour_plane_flag	chroma_format_idc	Comment
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
3	1	3	separate colour plane

$N_{dbfInstances}$ is the number of deblocking filtering instances in the specified period. It is derived from portion_deblocking_instances, $N_{4x4BlksInPeriod}$ and $S_{chrMult}$ in the decoder.

max_num_slices_tiles_minus1 specifies the maximum number between the number of slices and the number of tiles in the associated picture.

first_ctb_in_slice_or_tile[t] specifies the first Coding Tree Block (CTB) number in slice[t] or tile[t] in raster scan order.

$N_{4x4BlksInSliceOrTile}[t]$ is the total number of 4x4 blocks in the slice[t] or tile[t] and is determined, after derivation of the Coding tree block raster and tile scanning conversion process (CtbAddrRsToTs) as specified in ISO/IEC 23008-2:2020, 6.5.1, by the following computation:

— The value $I_{firstCtb}$ is set equal to CtbAddrRsToTs[first_ctb_in_slice_or_tile[t]].

— If t is lower than num_max_slices_tiles_minus1, the value $I_{lastCtb}$ is set equal to CtbAddrRsToTs[first_ctb_in_slice_or_tile[t+1]].

— Otherwise (t equal to num_max_slices_tiles_minus1), the value $I_{lastCtb}$ is set equal to CtbAddrRsToTs[PicSizeInCtbsY].

— $N_{4x4BlksInSliceOrTile}[t]$ is derived as follows:

$$N_{4x4BlksInSliceOrTile}[t] = (I_{lastCtb} - I_{firstCtb}) * N_{ctbs} \quad (6\text{-}45)$$

portion_non_zero_blocks_area[t] indicates the portion of area covered by blocks with non-zero transform coefficients values in the slice[t] or tile[t] using a 4x4 blocks granularity and is set equal to $P_{nonZeroBlksAreaInSliceOrTile}[t]$ defined as follows:

$$P_{nonZeroBlksAreaInSliceOrTile}[t] = Floor\left(\frac{N_{nonZeroBlksInSliceOrTile}[t]}{N_{4x4BlksInSliceOrTile}[t]} * 255 \right) \quad (6\text{-}46)$$

$N_{nonZeroBlksInSliceOrTile}[t]$ is the number of blocks with non-zero transform coefficients values in the slice[t] or tile[t] using 4x4 granularity. At the encoder side, it is computed as follows:

$$\begin{aligned} N_{nonZeroBlksInSliceOrTile}[t] &= N_{nonZero4x4BlksInSliceOrTile}[t] \\ &+ 4*N_{nonZero8x8BlksInSliceOrTile}[t] + 16*N_{nonZero16x16BlksInSliceOrTile}[t] \\ &+ 64*N_{nonZero32x32BlksInSliceOrTile}[t] \end{aligned} \quad (6\text{-}47)$$

where $N_{nonZero4x4BlksInSliceOrTile}[t]$, $N_{nonZero8x8BlksInSliceOrTile}[t]$, $N_{nonZero16x16BlksInSliceOrTile}[t]$, $N_{nonZero32x32BlksInSliceOrTile}[t]$ are the number of non-zero 4x4, 8x8, 16x16, 32x32 blocks in the slice[t] or tile[t] respectively.

$N_{nonZeroBlksInSliceOrTile}[t]$ is derived from portion_non_zero_blocks_area[t] and $N_{4x4BlksInSliceOrTile}[t]$ in the decoder.

portion_8x8_blocks_in_non_zero_area[t] indicates the portion of 8x8 blocks area in the non-zero area in the slice[t] or tile[t] and is set equal to $P_{nonZero8x8BlksInSliceOrTile}[t]$ defined as follows:

$$P_{nonZero8x8BlksInSliceOrTile}[t] = Floor\left(\frac{4 * N_{nonZero8x8BlksInSliceOrTile}[t]}{N_{nonZeroBlksInSliceOrTile}[t]} * 255 \right) \quad (6\text{-}48)$$

When not present, it is set equal to 0.

$N_{nonZero8x8BlksInSliceOrTile}[t]$ is the number of 8x8 blocks with non-zero transform coefficients values in the slice[t] or tile[t]. It is derived from portion_8x8_blocks_in_non_zero_area[t] and $N_{nonZeroBlksInSliceOrTile}[t]$ in the decoder.

portion_16x16_blocks_in_non_zero_area[t] indicates the portion of 16x16 blocks area in the non-zero area in the slice[t] or tile[t] and is set equal to $P_{nonZero16x16BlksInSliceOrTile}[t]$ defined as follows:

$$P_{nonZero16x16BlksInSliceOrTile}[t] = Floor\left(\frac{16 * N_{nonZero16x16BlksInSliceOrTile}[t]}{N_{nonZeroBlksInSliceOrTile}[t]} * 255 \right) \quad (6\text{-}49)$$

When not present, it is set equal to 0.

$N_{nonZero16x16BlksInSliceOrTile}[\ t\ ]$ is the number of 16x16 blocks with non-zero transform coefficients values in the slice[t] or tile[t]. It is derived from portion_16x16_blocks_in_non_zero_area[t] and $N_{nonZeroBlksInSliceOrTile}[\ t\ ]$ in the decoder.

portion_32x32_blocks_in_non_zero_area[t] indicates the portion of 32x32 blocks area in the non- zero area in the slice[t] or tile[t] and is set equal to $P_{nonZero32x32BlksInSliceOrTile}[\ t\ ]$ defined as follows:

$$P_{nonZero32x32BlksInSliceOrTile}[\ t\ ] = \text{Floor}\left(\frac{64 * N_{nonZero32x32BlksInSliceOrTile}[\ t\ ]}{N_{nonZeroBlksInSliceOrTile}[\ t\ ]} * 255 \right) \tag{6-50}$$

When not present, it is set equal to 0.

$N_{nonZero32x32BlksInSliceOrTile}[\ t\ ]$ is the number of 32x32 blocks with non-zero transform coefficients values in the slice[t] or tile[t]. It is derived from portion_32x32_blocks_in_non_zero_area[t] and $N_{nonZeroBlksInSliceOrTile}[\ t\ ]$ in the decoder.

$N_{nonZero4x4BlksInSliceOrTile}[\ t\ ]$ is the number of 4x4 blocks with non-zero transform coefficients values in the slice[t] or tile[t]. It is derived from $N_{nonZeroBlksInSliceOrTile}[\ t\ ]$, $N_{nonZero8x8BlksInSliceOrTile}[\ t\ ]$, $N_{nonZero16x16BlksInSliceOrTile}[\ t\ ]$ and $N_{nonZero32x32BlksInSliceOrTile}[\ t\ ]$ as follows in the decoder:

$$\begin{aligned} N_{nonZero4x4BlksInSliceOrTile}[\ t\ ] &= N_{nonZeroBlksInSliceOrTile}[\ t\ ] \\ &- 4 * N_{nonZero8x8BlksInSliceOrTile}[\ t\ ] - 16 * N_{nonZero16x16BlksInSliceOrTile}[\ t\ ] \\ &- 64 * N_{nonZero32x32BlksInSliceOrTile}[\ t\ ] \end{aligned} \tag{6-51}$$

portion_intra_predicted_blocks_area[t] indicates the portion of area covered by intra predicted blocks in the slice[t] or tile[t] using 8x8 granularity and is set equal to $P_{intraBlks}[\ t\ ]$ defined as follows:

$$P_{intraBlks}[\ t\ ] = \text{Floor}\left(\frac{4 * N_{intraBlksInSliceOrTile}[\ t\ ]}{N_{4x4BlksInSliceOrTile}[\ t\ ]} * 255 \right) \tag{6-52}$$

$N_{intraBlksInSliceOrTile}[\ t\ ]$ is the number of intra predicted blocks using 8x8 granularity in the slice[t] or tile[t]. At the encoder side, it is computed as follows:

$$\begin{aligned} N_{intraBlksInSliceOrTile}[\ t\ ] &= N_{intra8x8BlksInSliceOrTile}[\ t\ ] \\ &+ 4 * N_{intra16x16BlksInSliceOrTile}[\ t\ ] + 16 * N_{intra32x32BlksInSliceOrTile}[\ t\ ] \\ &+ 64 * N_{intra64x64BlksInSliceOrTile}[\ t\ ] \end{aligned} \tag{6-53}$$

where $N_{intra8x8BlksInSliceOrTile}[\ t\ ]$, $N_{intra16x16BlksInSliceOrTile}[\ t\ ]$, $N_{intra32x32BlksInSliceOrTiles}[\ t\ ]$ and $N_{intra64x64BlksInSliceOrTile}[\ t\ ]$ are the number of intra predicted 8x8, 16x16, 32x32 and 64x64 blocks in the slice[t] or tile[t] respectively.

$N_{intraBlksInSliceOrTile}[\ t\ ]$ is derived from portion_intra_predicted_blocks_area[t] and $N_{4x4BlksInSliceOrTile}[\ t\ ]$ in the decoder.

portion_planar_blocks_in_intra_area[t] indicates the portion of planar blocks in the intra predicted area the slice[t] or tile[t] and is set equal to $P_{planarBlksInIntra}[\ t\ ]$ defined as follows:

$$P_{planarBlksInIntra}[\ t\ ] = \text{Floor}\left(\frac{N_{planarBlksInSliceOrTile}[\ t\ ]}{4 * N_{intraBlksInSliceOrTile}[\ t\ ]} * 255 \right) \tag{6-54}$$

When not present, it is set equal to 0.

$N_{planarBlksInSliceOrTile}[\,t\,]$ is the number of intra planar predicted blocks in the slice[t] or tile[t] using 4x4 granularity. At the encoder side, it is computed as follows:

$$N_{planarBlksInSliceOrTile}[\,t\,] = N_{planar4x4Blks}[\,t\,] + 4*N_{planar8x8Blks}[\,t\,] + 16*N_{planar16x16Blks}[\,t\,] + 64*N_{planar32x32Blks}[\,t\,] + 256*N_{planar64x64Blks}[\,t\,] \quad (6\text{-}55)$$

where $N_{planar4x4Blks}[\,t\,]$, $N_{planar8x8Blks}[\,t\,]$, $N_{planar16x16Blks}[\,t\,]$, $N_{planar32x32Blks}[\,t\,]$ and $N_{planar64x64Blks}[\,t\,]$ are the number of intra planar predicted 4x4, 8x8, 16x16, 32x32 and 64x64 blocks in the slice[t] or tile[t] respectively.

$N_{planarBlksInSliceOrTile}[\,t\,]$ is derived from portion_planar_blocks_in_intra_area[t] and $N_{intraBlksInSliceOrTile}[\,t\,]$ in the decoder.

portion_dc_blocks_in_intra_area[t] indicates the portion of DC blocks in the intra predicted area in the slice[t] or tile[t] and is set equal to $P_{DCBlksInIntra}[\,t\,]$ defined as follows:

$$P_{DCBlksInIntra}[\,t\,] = \mathrm{Floor}\left(\frac{N_{DCBlksInSliceOrTile}[\,t\,]}{4 * N_{intraBlksInSliceOrTile}[\,t\,]} * 255 \right) \quad (6\text{-}56)$$

When not present, it is set equal to 0.

$N_{DCBlksInSliceOrTile}[\,t\,]$ is the number of intra DC predicted blocks in the slice[t] or tile[t] using 4x4 granularity. At the encoder side, it is computed as follows:

$$N_{DCBlksInSliceOrTile}[\,t\,] = N_{DC4x4Blks}[\,t\,] + 4*N_{DC8x8Blks}[\,t\,] + 16*N_{DC16x16Blks}[\,t\,] + 64*N_{DC32x32Blks}[\,t\,] + 256*N_{DC64x64Blks}[\,t\,] \quad (6\text{-}57)$$

where $N_{DC4x4Blks}[\,t\,]$, $N_{DC8x8Blks}[\,t\,]$, $N_{DC16x16Blks}[\,t\,]$, $N_{DC32x32Blks}[\,t\,]$ and $N_{DC64x64Blks}[\,t\,]$ are the number of intra DC predicted 4x4, 8x8, 16x16, 32x32 and 64x64 blocks in the slice[t] or tile[t] respectively.

$N_{DCBlksInSliceOrTile}[\,t\,]$ is derived from portion_dc_blocks_in_intra_area[t] and $N_{intraBlksInSliceOrTile}[\,t\,]$ in the decoder.

portion_angular_hv_blocks_in_intra_area[t] indicates the portion of angular horizontal or vertical blocks in the intra predicted area in the slice[t] or tile[t] and is set equal to $P_{HVBlksInIntra}[\,t\,]$ defined as follows:

$$P_{HVBlksInIntra}[\,t\,] = \mathrm{Floor}\left(\frac{N_{HVBlksInSliceOrTile}[\,t\,]}{4 * N_{intraBlksInSliceOrTile}[\,t\,]} * 255 \right) \quad (6\text{-}58)$$

When not present, it is set equal to 0.

$N_{HVBlksInSliceOrTile}[\,t\,]$ is the number of intra angular horizontally or vertically predicted blocks in the slice[t] or tile[t] using 4x4 granularity. At the encoder side, it is computed as follows:

$$N_{HVBlksInSliceOrTile}[\,t\,] = N_{HV4x4Blks}[\,t\,] + 4*N_{HV8x8Blks}[\,t\,] + 16*N_{HV16x16Blks}[\,t\,] + 64*N_{HV32x32Blks}[\,t\,] + 256*N_{HV64x64Blks}[\,t\,] \quad (6\text{-}59)$$

where $N_{HV4x4Blks}[\,t\,]$, $N_{HV8x8Blks}[\,t\,]$, $N_{HV16x16Blks}[\,t\,]$, $N_{HV32x32Blks}[\,t\,]$ and $N_{HV64x64Blks}[\,t\,]$ are the number of intra angular horizontally or vertically predicted 4x4, 8x8, 16x16, 32x32 and 64x64 blocks in the slice[t] or tile[t] respectively.

$N_{HVBlksInSliceOrTile}[\,t\,]$ is derived from portion_angular_hv_blocks_in_intra_area[t] and $N_{intraBlksInSliceOrTile}[\,t\,]$ in the decoder.

portion_blocks_a_c_d_n_filterings[t] indicates the portion of prediction blocks whose luma samples position are located in sub-sample position a, c, d or n, as defined in Annex B, in the slice[t] or tile[t]. When not present, is equal to 0. portion_blocks_a_c_d_n_filterings[t] is set equal to $P_{acdnFiltBlks}[\,t\,]$ defined as follows:

$$P_{acdnFiltBlks}[\,t\,] = Floor\left(\frac{N_{acdnFiltBlks}[\,t\,]}{N_{4x4BlksInSliceOrTile}[\,t\,]} * 255 \right) \tag{6-60}$$

$N_{acdnFiltBlks}[\,t\,]$ is the number of prediction blocks whose luma samples position are located in sub-sample position a, c, d or n, as defined in Annex B, in the slice[t] or tile[t]. It is derived from portion_blocks_a_c_d_n_filterings[t] and $N_{4x4BlksInSliceOrTile}[\,t\,]$ in the decoder.

portion_blocks_h_b_filterings[t] indicates the portion of prediction blocks whose luma samples position are located in sub-sample position h or b, as defined in Annex B, in the slice[t] or tile[t]. When not present, is equal to 0. portion_blocks_h_b_filterings[t] is set equal to $P_{hbFiltBlks}[\,t\,]$ defined as follows:

$$P_{hbFiltBlks}[\,t\,] = Floor\left(\frac{N_{hbFiltBlks}[\,t\,]}{N_{4x4BlksInSliceOrTile}[\,t\,]} * 255 \right) \tag{6-61}$$

$N_{hbFiltBlks}[\,t\,]$ is the number of prediction blocks whose luma samples position are located in sub-sample position h or b, as defined in Annex B, in the slice[t] or tile[t]. It is derived from portion_blocks_h_b_filterings[t] and $N_{4x4BlksInSliceOrTile}[\,t\,]$ in the decoder.

portion_blocks_f_i_k_q_filterings[t] indicates the portion of prediction blocks whose luma samples position are located in sub-sample position f, i, k or q, as defined in Annex B, in the slice[t] or tile[t]. When not present, is equal to 0. portion_blocks_f_i_k_q_filterings[t] is set equal to $P_{fikqFiltBlks}[\,t\,]$ defined as follows:

$$P_{fikqFiltBlks}[\,t\,] = Floor\left(\frac{N_{fikqFiltBlks}[\,t\,]}{N_{4x4BlksInSliceOrTile}[\,t\,]} * 255 \right) \tag{6-62}$$

$N_{fikqFiltBlks}[\,t\,]$ is the number of prediction blocks whose luma samples position are located in sub-sample position f, i, k or q, as defined in Annex B, in the slice[t] or tile[t]. It is derived from portion_blocks_f_i_k_q_filterings[t] and $N_{4x4BlksInSliceOrTile}[\,t\,]$ in the decoder.

portion_blocks_j_filterings[t] indicates the portion of prediction blocks whose luma samples position are located in sub-sample position j, as defined in Annex B, in the slice[t] or tile[t]. When not present, is equal to 0. portion_blocks_j_filterings[t] is set equal to $P_{jFiltBlks}[\,t\,]$ defined as follows:

$$P_{jFiltBlks}[\,t\,] = Floor\left(\frac{N_{jFiltBlks}[\,t\,]}{N_{4x4BlksInSliceOrTile}[\,t\,]} * 255 \right) \tag{6-63}$$

$N_{jFiltBlks}[\,t\,]$ is the number of prediction blocks whose luma samples position are located in sub-sample position j, as defined in Annex B, in the slice[t] or tile[t]. It is derived from portion_blocks_j_filterings[t] and $N_{4x4BlksInSliceOrTile}[\,t\,]$ in the decoder.

portion_blocks_e_g_p_r_filterings[t] indicates the portion of prediction blocks whose luma samples position are located in sub-sample position e, g, p or r, as defined in Annex B, in the slice[t] or tile[t]. When not present, is equal to 0. portion_blocks_ e_g_p_r_filterings[t] is set equal to $P_{egprFiltBlks}[\,t\,]$ defined as follows:

$$P_{egprFiltBlks}[\,t\,] = Floor\left(\frac{N_{egprFiltBlks}[\,t\,]}{N_{4x4BlksInSliceOrTile}[\,t\,]} * 255 \right) \tag{6-64}$$

$N_{egprFiltBlks}[\,t\,]$ is the number of prediction blocks whose luma samples position are located in sub-sample position e, g, p or r, as defined in Annex B, in the slice[t] or tile[t]. It is derived from portion_blocks_e_g_p_r_filterings[t] and $N_{4x4BlksInSliceOrTile}[\,t\,]$ in the decoder.

portion_deblocking_instances[t] indicates the portion of deblocking filtering instances in the slice[t] or tile[t]. portion_deblocking_instances[t] is set equal to $P_{\text{dbfInstances}}$[t] defined as follows:

$$P_{\text{dbfInstances}}[\,t\,] = \text{Floor}\left(\frac{N_{\text{dbfInstances}}[\,t\,]}{4 * S_{\text{chrMult}} * N_{4\text{x}4\text{BlksInSliceOrTile}}[\,t\,]} * 255 \right) \tag{6-65}$$

$N_{\text{dbfInstances}}$[t] is the number of deblocking filtering instances in the slice[t] or tile[t]. It is derived from portion_deblocking_instances[t], $N_{\text{4x4BlksInSliceOrTile}}$[t] and S_{chrMult} in the decoder.

6.2.4.3 VVC semantics

The semantics of various terms are defined below.

period_type specifies the type of upcoming period over which the complexity metrics are applicable and is defined in the Table 8.

Table 8 — specification of period_type for VVC

Value	Description
0x0	complexity metrics are applicable to a single picture
0x1	complexity metrics are applicable to all pictures in decoding order, up to (but not including) the picture containing the next I slice
0x2	complexity metrics are applicable to all pictures over a specified time interval in seconds
0x3	complexity metrics are applicable over a specified number of pictures counted in decoding order
0x4-0xF	user-defined

granularity_type indicates the type of granularity which the complexity metrics are applicable and is defined in the Table 9.

Table 9 — specification of granularity_type for VVC

Value	Description
0x0	complexity metrics are applicable to picture granularity
0x1	complexity metrics are applicable to sub-picture granularity
0x2	complexity metrics are applicable to slice granularity
0x3	complexity metrics are applicable to tile granularity
0x4-0x7	user-defined

extended_representation_flag equal to 1 indicates that the syntax elements portion_non_zero_4_8_16_blocks_area, portion_non_zero_32_64_128_blocks_area, portion_non_zero_256_512_1024_blocks_area, portion_non_zero_2048_4096_blocks_area, portion_bi_and_gpm_predicted_blocks_area, portion_bdof_blocks_area, portion_sao_filtered_blocks, portion_non_zero_4_8_16_blocks_area[t], portion_non_zero_32_64_128_blocks_area[t], portion_non_zero_256_512_1024_blocks_area[t], portion_non_zero_2048_4096_blocks_area[t], portion_bi_and_gpm_predicted_blocks_area[t], portion_bdof_blocks_area[t] and portion_sao_filtered_blocks[t] may be present. extended_representation_flag equal to 0 indicates that these syntax elements are not present.

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

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

$N_{\text{picsInPeriod}}$ indicates the number of pictures in the specified period. When period_type is 0, then $N_{\text{picsInPeriod}}$ is 1. When period_type is 1, then $N_{\text{picsInPeriod}}$ 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 $N_{picsInPeriod}$ is determined from the frame rate. When period_type is 3, then $N_{picsInPeriod}$ is equal to num_pictures.

$N_{4SampleBlksInPeriod}$ is the total number of 4-samples luma and chroma blocks that are coded in the specified period.

It is determined by the following computation:

$$N_{4SampleBlksInPeriod} = \sum_{n=1}^{N_{picsInPeriod}} N_{4SampleBlksPic}(n) \quad (6\text{-}66)$$

where $N_{4SampleBlksPic}(n)$ is derived for the n^{th} picture within the specified period $1 <= n <= N_{picsInPeriod}$ from VVC variables PicSizeInCtbsY and CtbLog2SizeY specified for the decoding process of the n^{th} picture within the specified period, as follows:

$$N_{4SampleBlksPic}(n) = S_{chrMult} * S_{picInCtb} * N_{ctbs} \quad (6\text{-}67)$$

where

— N_{ctbs} is set equal to (1 << (CtbLog2SizeY – 1))^2

— $S_{picInCtb}$ is set equal to PicSizeInCtbsY

— $S_{chrMult}$ depends on the VVC variable sps_chroma_format_idc as shown in the Table 10

Table 10 — specification of $S_{chrMult}$ for VVC

$S_{chrMult}$	sps_chroma_format_idc	Comment
1	0	monochrome
1.5	1	4:2:0 sampling
2	2	4:2:2 sampling
3	3	4:4:4 sampling

portion_non_zero_blocks_area indicates the portion of area covered by blocks with non-zero transform coefficients values, in the pictures of the specified period, using 4-samples block granularity and is set equal to $P_{nonZeroBlksArea}$ defined as follows:

$$P_{nonZeroBlksArea} = \text{Floor}\left(\frac{N_{nonZeroBlks}}{N_{4SampleBlksInPeriod}} * 255 \right) \quad (6\text{-}68)$$

where $N_{nonZeroBlks}$ is the number of blocks with non-zero transform coefficients values in the specified period using 4-samples block granularity. At the encoder side, it is computed as follows:

$$N_{nonZeroBlks} = \sum_{X=4,8,16,32,64,128,256,512,1024,2048,4096} \left(\frac{X}{4} * N_{nonZeroBlks\_X} \right) \quad (6\text{-}69)$$

where $N_{nonZeroBlks\_X}$ is the number of blocks with non-zero transform coefficients values, for transform blocks with number of samples X=4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, respectively, in the specified period.

$N_{nonZeroBlks}$ is derived from portion_non_zero_blocks_area and $N_{4SampleBlksInPeriod}$ in the decoder.

portion_non_zero_4_8_16_blocks_area indicates the portion of 4-, 8- and 16-samples blocks area in the non-zero area in the specified period and is set equal to $P_{\text{nonZero4\_8\_16\_Blks}}$ defined as follows:

$$P_{\text{nonZero4\_8\_16\_Blks}} = \text{Floor}\left(\frac{N_{\text{nonZero4\_8\_16\_Blks}}}{N_{\text{nonZeroBlks}}} * 255\right) \tag{6-70}$$

When not present, portion_non_zero_4_8_16_blocks_area is set equal to 0.

$N_{\text{nonZero4\_8\_16\_Blks}}$ is the number of 4-, 8- and 16-samples transform blocks with non-zero transform coefficients values in the specified period using 4-samples block granularity. At the encoder side, it is computed as follows:

$$N_{\text{nonZero4\_8\_16\_Blks}} = \sum_{X=4,8,16}\left(\frac{X}{4} * N_{\text{nonZeroBlks\_}X}\right) \tag{6-71}$$

$N_{\text{nonZero4\_8\_16\_Blks}}$ is derived from portion_non_zero_4_8_16_blocks_area and $N_{\text{nonZeroBlks}}$ in the decoder.

portion_non_zero_32_64_128_blocks_area indicates the portion of 32-, 64- and 128-samples blocks area in the non-zero area in the specified period and is set equal to $P_{\text{nonZero32\_64\_128\_Blks}}$ defined as follows:

$$P_{\text{nonZero32\_64\_128\_Blks}} = \text{Floor}\left(\frac{N_{\text{nonZero32\_64\_128\_Blks}}}{N_{\text{nonZeroBlks}}} * 255\right) \tag{6-72}$$

When not present, portion_non_zero_32_64_128_blocks_area is set equal to 0.

$N_{\text{nonZero32\_64\_128\_Blks}}$ is the number of 32-, 64- and 128-samples transform blocks with non-zero transform coefficients values in the specified period using 4-samples block granularity. At the encoder side, it is computed as follows:

$$N_{\text{nonZero32\_64\_128\_Blks}} = \sum_{X=32,64,128}\left(\frac{X}{4} * N_{\text{nonZeroBlks\_}X}\right) \tag{6-73}$$

$N_{\text{nonZero32\_64\_128\_Blks}}$ is derived from portion_non_zero_32_64_128_blocks_area and $N_{\text{nonZeroBlks}}$ in the decoder.

portion_non_zero_256_512_1024_blocks_area indicates the portion of 256-, 512- and 1024-samples blocks area in the non-zero area in the specified period and is set equal to $P_{\text{nonZero256\_512\_1024\_Blks}}$ defined as follows:

$$P_{\text{nonZero256\_512\_1024\_Blks}} = \text{Floor}\left(\frac{N_{\text{nonZero256\_512\_1024\_Blks}}}{N_{\text{nonZeroBlks}}} * 255\right) \tag{6-74}$$

When not present, portion_non_zero_256_512_1024_blocks_area is set equal to 0.

$N_{\text{nonZero256\_512\_1024\_Blks}}$ is the number of 256-, 512- and 1024-samples transform blocks with non-zero transform coefficients values in the specified period using 4-samples block granularity. At the encoder side, it is computed as follows:

$$N_{\text{nonZero256\_512\_1024\_Blks}} = \sum_{X=256,512,1024}\left(\frac{X}{4} * N_{\text{nonZeroBlks\_}X}\right) \tag{6-75}$$

$N_{\text{nonZero256\_512\_1024\_Blks}}$ is derived from portion_non_zero_256_512_1024_blocks_area and $N_{\text{nonZeroBlks}}$ in the decoder.

portion_non_zero_2048_4096_blocks_area indicates the portion of 2048- and 4096-samples blocks area in the non-zero area in the specified period and is set equal to $P_{\text{nonZero2018\_4096\_Blks}}$ defined as follows:

$$P_{\text{nonZero2048\_4096\_Blks}} = \text{Floor}\left(\frac{N_{\text{nonZero2048\_4096\_Blks}}}{N_{\text{nonZeroBlks}}} * 255\right) \tag{6-76}$$

When not present, portion_non_zero_2048_4096_blocks_area is set equal to 0.

$N_{\text{nonZero2048\_4096\_Blks}}$ is the number of 2048- and 4096-samples transform blocks with non-zero transform coefficients values in the specified period using 4-samples block granularity. At the encoder side, it is computed as follows:

$$N_{\text{nonZero2048\_4096\_Blks}} = \sum_{X=2048,4096}\left(\frac{X}{4} * N_{\text{nonZeroBlks\_}X}\right) \tag{6-77}$$

$N_{\text{nonZero2048\_4096\_Blks}}$ is derived from portion_non_zero_2048_4096_blocks_area and $N_{\text{nonZeroBlks}}$ in the decoder.

portion_non_zero_transform_coefficients_area indicates the portion of area covered by non-zero transform coefficients in non-zero transform blocks in the pictures of the specified period, using 4-samples block granularity and is set equal to $P_{\text{nonZeroCoefsArea}}$ defined as follows:

$$P_{\text{nonZeroCoefsArea}} = \text{Floor}\left(\frac{N_{\text{nonZeroTransformCoefs}}}{4 * N_{\text{nonZeroBlks}}} * 255\right) \tag{6-78}$$

$N_{\text{nonZeroTransformCoefs}}$ is the area covered by non-zero transform coefficients in non-zero transform blocks in the specified period using 4-samples block granularity.

$N_{\text{nonZeroTransformCoefs}}$ is derived from portion_non_zero_transform_coefficients_area and $N_{\text{nonZeroBlks}}$ in the decoder.

portion_intra_predicted_blocks_area indicates the portion of area covered by intra predicted blocks in the pictures of the specified period using 4-samples block granularity and is set equal to $P_{\text{intraPredBlks}}$ defined as follows:

$$P_{\text{intraPredBlks}} = \text{Floor}\left(\frac{N_{\text{intraPredBlks}}}{N_{\text{4SampleBlksInPeriod}}} * 255\right) \tag{6-79}$$

$N_{\text{intraPredBlks}}$ is the number of intra predicted blocks in the specified period using 4-samples block granularity. At the encoder side, it is computed as follows:

$$N_{\text{intraPredBlks}} = \sum_{X=4,8,16,32,64,128,256,512,1024,2048,4096}\left(\frac{X}{4} * N_{\text{intraPredBlks\_}X}\right) \tag{6-80}$$

where $N_{\text{intraPredBlks\_}X}$ is the number of blocks using intra prediction, for blocks with number of samples X=4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, in the specified period.

$N_{\text{intraPredBlks}}$ is derived from portion_intra_predicted_blocks_area and $N_{\text{4SampleBlksInPeriod}}$ in the decoder.

portion_bi_and_gpm_predicted_blocks_area indicates the portion of area covered by inter bi-predicted or GPM-predicted blocks in the pictures of the specified period using 4-samples block granularity and is set equal to $P_{\text{biGpmPredBlks}}$ defined as follows:

$$P_{\text{biGpmBlks}} = \text{Floor}\left(\frac{N_{\text{biAndGpmPredBlks}}}{N_{\text{4SampleBlksInPeriod}}} * 255\right) \tag{6-81}$$

$N_{\text{biAndGpmPredBlks}}$ is the number of inter bi-predicted and GPM-predicted blocks in the specified period using 4-samples block granularity. At the encoder side, it is computed as follows:

$$N_{\text{biAndGpmPredBlks}} = \sum_{X=4,8,16,32,64,128,256,512,1024,2048,4096} \left(\frac{X}{4} * N_{\text{biAndGpmPredBlks\_}X}\right) \tag{6-82}$$

Where $N_{\text{biAndGpmPredBlks\_}X}$ are the number of blocks using inter bi-prediction or GPM prediction, for blocks with number of samples X=4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, in the specified period.

$N_{\text{biAndGpmPredBlks}}$ is derived from portion_bi_and_gpm_predicted_blocks_area and $N_{\text{4SampleBlksInPeriod}}$ in the decoder.

portion_bdof_blocks_area indicates the portion of area covered by inter blocks using BDOF in the pictures of the specified period using 4-samples block granularity and is set equal to P_{bdofBlks} defined as follows:

$$P_{\text{bdofBlks}} = \text{Floor}\left(\frac{N_{\text{bdofBlks}}}{N_{\text{4SampleBlksInPeriod}}} * 255\right) \tag{6-83}$$

N_{bdofBlks} is the number of inter blocks using BDOF in the specified period using 4-samples block granularity. At the encoder side, it is computed as follows:

$$N_{\text{bdofBlks}} = \sum_{X=128,256,512,1024,2048,4096} \left(X / 4 * N_{\text{bdofBlks\_}X}\right) \tag{6-84}$$

Where $N_{\text{bdofBlks\_}X}$ are the number of inter-coded blocks using BDOF, for blocks with number of samples X=128, 256, 512, 1024, 2048, 4096, in the specified period.

N_{bdofBlks} is derived from portion_bdof_blocks_area and $N_{\text{4SampleBlksInPeriod}}$ in the decoder.

portion_deblocking_instances indicates the portion of deblocking filtering instances in the specified period and is set equal to $P_{\text{dbfInstances}}$ defined as follows:

$$P_{\text{dbfInstances}} = \text{Floor}\left(\frac{N_{\text{dbfInstances}}}{4 * N_{\text{4SampleBlksInPeriod}}} * 255\right) \tag{6-85}$$

$N_{\text{dbfInstances}}$ is the number of deblocking filtering instances in the specified period. It is derived from portion_deblocking_instances, and $N_{\text{4SampleBlksInPeriod}}$ in the decoder.

portion_sao_filtered_blocks indicates the portion of SAO filtered blocks in the specified period using 4-samples block granularity. At the encoder side, it is set equal to P_{saoBlks} computed as follows:

$$P_{\text{saoBlks}} = \text{Floor}\left(\frac{N_{\text{saoFilteredBlks}}}{N_{\text{4SampleBlksInPeriod}}} * 255\right) \tag{6-86}$$

$N_{\text{saoFilteredBlks}}$ is the number of SAO filtered blocks in the specified period using 4-samples block granularity. It is derived from portion_sao_filtered_blocks, $N_{\text{4SampleBlksInPeriod}}$ in the decoder.

portion_alf_filtered_blocks indicates the portion of ALF filtered blocks in the specified period using 4-samples block granularity. At the encoder side, it is set equal to P_{alfBlks} computed as follows:

$$P_{\text{alfBlks}} = \text{Floor}\left(\frac{N_{\text{alfFilteredBlks}}}{N_{\text{4SampleBlksInPeriod}}} * 255 \right) \tag{6-87}$$

$N_{\text{alfFilteredBlks}}$ is the number of ALF filtered blocks in the specified period using 4-samples block granularity. It is derived from portion_alf_filtered_blocks and $N_{\text{4SampleBlksInPeriod}}$ in the decoder.

max_num_segments_minus1 indicates the number of subpictures, slices or tiles in the associated picture.

segment_address[t] indicates the address of the t^{th} segment. When granularity_type is equal to 1, segment_address[t] indicates the subpicture ID of the t^{th} subpicture subpicture[t]. When granularity_type is equal to 2 or 3, segment_address[t] indicates the picture raster scan address of the first coding tree block (CTB) number in slice[t] or tile[t].

$N_{\text{4SampleBlksInSegment}}$[t] is the total number of 4-samples luma and chroma blocks in the slice[t] or tile[t] or subpicture[t]. $N_{\text{4SampleBlksInSegment}}$[t] is determined by the following computation.

— If granularity_type is equal to 1, $N_{\text{4SampleBlksInSegment}}$[t] is derived as follows from VVC variables sps_subpic_id, sps_subpic_width_minus1, sps_subpic_height_minus1 and CtbLog2SizeY specified in ISO/IEC 23090-3

  — s is defined as the index value such that sps_subpic_id[s] is equal to the subpicture ID segment_address[t].

  — W_{subpic} is set equal to (1 + sps_subpic_width_minus1[s]) << (CtbLog2SizeY – 1).

  — H_{subpic} is set equal to (1 + sps_subpic_height_minus1[s]) << (CtbLog2SizeY – 1).

  — $N_{\text{4SampleBlksInSegment}}$[t] is set equal to (S_{chrMult} * W_{subpic} * H_{subpic}).

— if granularity_type is equal to 2, $N_{\text{4SampleBlksInSegment}}$[t] is derived as follows from VVC variables NumCtusInSlice and CtbLog2SizeY specified in ISO/IEC 23090-3:

  — $N_{\text{4SampleBlksInSegment}}$[t] is set equal to S_{chrMult} * ((NumCtusInSlice[t]) << (CtbLog2SizeY – 1))

— Otherwise, if granularity_type is equal to 3, $N_{\text{4SampleBlksInSegment}}$[t] is derived as follows from VVC variables ctbToTileColIdx, ctbToTileRowIdx, ColWidthVal and RowHeightVal specified in ISO/IEC 23090-3:

  — ctbAddrX is set equal to segment_address[t].

  — tileColIdx is set equal to ctbToTileColIdx[ctbAddrX].

  — tileRowIdx is set equal to ctbToTileRowIdx[ctbAddrX].

  — W_{tile} is set equal to ColWidthVal[tileColIdx] << (CtbLog2SizeY – 1).

  — H_{tile} is set equal to RowHeightVal[tileRowIdx] << (CtbLog2SizeY – 1).

  — $N_{\text{4SampleBlksInSegment}}$[t] is set equal to (S_{chrMult} * W_{tile} * H_{tile}).

portion_non_zero_blocks_area[t] indicates the portion of area covered by blocks with non-zero transform coefficients values, in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity and is set equal to $P_{\text{nonZeroBlksArea}}[\,t\,]$ defined as follows:

$$P_{\text{nonZeroBlksArea}}[\,t\,] = \text{Floor}\left(\frac{N_{\text{nonZeroBlksInSegment}}[\,t\,]}{N_{\text{4SampleBlksInSegment}}[\,t\,]} * 255 \right) \tag{6-88}$$

where $N_{\text{nonZeroBlksInSegment}}[\,t\,]$ is the number of blocks with non-zero transform coefficients values, in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity. At the encoder side, it is computed as follows:

$$N_{\text{nonZeroBlksInSegment}}[\,t\,] = \sum_{X=4,8,16,32,64,128,256,512,1024,2048,4096} \left(X / 4 * N_{\text{nonZeroBlksInSegment\_}X}[\,t\,] \right) \tag{6-89}$$

where $N_{\text{nonZeroBlksInSegment\_}X}[\,t\,]$ is the number of blocks with non-zero transform coefficients values, in the slice[t] or tile[t] or subpicture[t], for blocks with number of samples X=4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, respectively.

$N_{\text{nonZeroBlksInSegment}}[\,t\,]$ is derived from portion_non_zero_blocks_area[t] and $N_{\text{4SampleBlksInSegment}}[\,t\,]$ in the decoder.

portion_non_zero_4_8_16_blocks_area[t] indicates the portion of 4-, 8- and 16-samples blocks area in the non-zero area in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity and is set equal to $P_{\text{nonZero4\_8\_16\_Blks}}[\,t\,]$ defined as follows:

$$P_{\text{nonZero4\_8\_16\_Blks}}[\,t\,] = \text{Floor}\left(\frac{N_{\text{nonZero4\_8\_16\_BlksInSegment}}[\,t\,]}{N_{\text{nonZeroBlksInSegment}}[\,t\,]} * 255 \right) \tag{6-90}$$

When not present, portion_non_zero_4_8_16_blocks_area[t] is set equal to 0.

$N_{\text{nonZero4\_8\_16\_BlksInSegment}}[\,t\,]$ is the number of 4-, 8- and 16-samples blocks with non-zero transform coefficients values in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity. At the encoder side, it is computed as follows:

$$N_{\text{nonZero4\_8\_16\_BlksInSegment}}[\,t\,] = \sum_{X=4,8,16} \left(\frac{X}{4} * N_{\text{nonZeroBlksInSegment\_}X}[\,t\,] \right) \tag{6-91}$$

$N_{\text{nonZero4\_8\_16\_BlksInSegment}}[\,t\,]$ is derived from portion_non_zero_4_8_16_blocks_area[t] and $N_{\text{nonZeroBlksInSegment}}[\,t\,]$ in the decoder.

portion_non_zero_32_64_128_blocks_area[t] indicates the portion of 32-, 64- and 128-samples blocks area in the non-zero area in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity and is set equal to $P_{\text{nonZero32\_64\_128\_Blks}}[\,t\,]$ defined as follows:

$$P_{\text{nonZero32\_64\_128\_Blks}}[\,t\,] = \text{Floor}\left(\frac{N_{\text{nonZero32\_64\_128\_BlksInSegment}}[\,t\,]}{N_{\text{nonZeroBlksInSegment}}[\,t\,]} * 255 \right) \tag{6-92}$$

When not present, portion_non_zero_32_64_128_blocks_area[t] is set equal to 0.

$N_{nonZero32\_64\_128\_BlksInSegment}[\,t\,]$ is the number of 32-, 64- and 128-samples blocks with non-zero transform coefficients values in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity. At the encoder side, it is computed as follows:

$$N_{nonZero32\_64\_128\_BlksInSegment}[\,t\,]=\sum_{X=32,64,128}\left(\frac{X}{4}*N_{nonZeroBlksInSegment\_X}[\,t\,]\right) \tag{6-93}$$

$N_{nonZero32\_64\_128\_BlksInSegment}[\,t\,]$ is derived from portion_non_zero_32_64_128_blocks_area[t] and $N_{nonZeroBlksInSegment}[\,t\,]$ in the decoder.

portion_non_zero_256_512_1024_blocks_area[t] indicates the portion of 256-, 512- and 1024-samples blocks area in the non-zero area in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity and is set equal to $P_{nonZero256\_512\_1024\_Blks}[\,t\,]$ defined as follows:

$$P_{nonZero256\_512\_1024\_Blks}[\,t\,]=\mathrm{Floor}\left(\frac{N_{nonZero256\_512\_1024\_BlksInSegment}[\,t\,]}{N_{nonZeroBlksInSegment}[\,t\,]}*255\right) \tag{6-94}$$

When not present, portion_non_zero_256_512_1024_blocks_area[t] is set equal to 0.

$N_{nonZero256\_512\_1024\_BlksInSegment}[\,t\,]$ is the number of 256-, 512- and 1024-samples blocks with non-zero transform coefficients values in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity. At the encoder side, it is computed as follows:

$$N_{nonZero256\_512\_1024\_BlksInSegment}[\,t\,]=\sum_{X=256,512,1024}\left(\frac{X}{4}*N_{nonZeroBlksInSegment\_X}[\,t\,]\right) \tag{6-95}$$

$N_{nonZero256\_512\_1024\_BlksInSegment}[\,t\,]$ is derived from portion_non_zero_256_512_1024_blocks_area[t] and $N_{nonZeroBlksInSegment}[\,t\,]$ in the decoder.

portion_non_zero_2048_4096_blocks_area[t] indicates the portion of 2048- and 4096-samples blocks area in the non-zero area in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity and is set equal to $P_{nonZero2048\_4096\_Blks}[\,t\,]$ defined as follows:

$$P_{nonZero2048\_4096\_Blks}[\,t\,]=\mathrm{Floor}\left(\frac{N_{nonZero2048\_4096\_BlksInSegment}[\,t\,]}{N_{nonZeroBlksInSegment}[\,t\,]}*255\right) \tag{6-96}$$

When not present, portion_non_zero_2048_4096_blocks_area[t] is set equal to 0.

$N_{nonZero2048\_4096\_BlksInSegment}[\,t\,]$ is the number of 2048- and 4096-samples blocks with non-zero transform coefficients values in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity. At the encoder side, it is computed as follows:

$$N_{nonZero2048\_4096\_BlksInSegment}[\,t\,]=\sum_{X=2048,4096}\left(\frac{X}{4}*N_{nonZeroBlksInSegment\_X}[\,t\,]\right) \tag{6-97}$$

$N_{nonZero2048\_4096\_BlksInSegment}[\,t\,]$ is derived from portion_non_zero_2048_4096_blocks_area[t] and $N_{nonZeroBlksInSegment}[\,t\,]$ in the decoder.

portion_non_zero_transform_coefficients_area[t] indicates the portion of area covered by non-zero transform coefficients in non-zero transform blocks in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity and is set equal to $P_{nonZeroCoefsArea}[\,t\,]$ defined as follows:

$$P_{nonZeroCoefsArea}=\mathrm{Floor}\left(\frac{N_{nonZeroTransformCoefs}[\,t\,]}{4*N_{nonZeroBlksInSegment}[\,t\,]}*255\right) \tag{6-98}$$

$N_{nonZeroTransformCoefs}[\,t\,]$ is the area covered by non-zero transform coefficients in non-zero blocks in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity.

$N_{nonZeroTransformCoefs}[\,t\,]$ is derived from portion_non_zero_transform_coefficients_area[t] and $N_{nonZeroBlksInSegment}[\,t\,]$ in the decoder.

portion_intra_predicted_blocks_area[t] indicates the portion of area covered by intra predicted blocks in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity and is set equal to $P_{\text{intraPredBlks}}[\,t\,]$ defined as follows:

$$P_{\text{intraPredBlks}}[\,t\,] = \text{Floor}\left(\frac{N_{\text{intraPredBlks}}[\,t\,]}{N_{\text{4SampleBlksInSegment}}[\,t\,]} * 255 \right) \tag{6-99}$$

$N_{\text{intraPredBlks}}[\,t\,]$ is the number of intra predicted blocks in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity. At the encoder side, it is computed as follows:

$$N_{\text{intraPredBlks}}[\,t\,] = \sum_{X=4,8,16,32,64,128,256,512,1024,2048,4096} \left(\frac{X}{4} * N_{\text{intraPredBlks\_}X}[\,t\,] \right) \tag{6-100}$$

Where $N_{\text{intraPredBlks\_}X}[\,t\,]$ is the number of blocks using intra prediction, for blocks with number of samples X=4, 8, 16, 32, 64, 128, 256, 512, 1024, 2048, 4096, in the slice[t] or tile[t] or subpicture[t].

$N_{\text{intraPredBlks}}[\,t\,]$ is derived from portion_intra_predicted_blocks_area[t] and $N_{\text{4SampleBlksInSegment}}[\,t\,]$ in the decoder.

portion_bi_and_gpm_predicted_blocks_area[t] indicates the portion of area covered by inter bi-predicted or GPM-predicted blocks in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity and is set equal to $P_{\text{biGpmPredBlks}}[\,t\,]$ defined as follows:

$$P_{\text{biGpmPredBlks}}[\,t\,] = \text{Floor}\left(\frac{N_{\text{biAndGpmPredBlks}}[\,t\,]}{N_{\text{4SampleBlksInSegment}}[\,t\,]} * 255 \right) \tag{6-101}$$

$N_{\text{biAndGpmPredBlks}}[\,t\,]$ is the number of inter bi-predicted and GPM-predicted blocks in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity. At the encoder side, it is computed as follows:

$$N_{\text{biAndGpmPredBlks}}[\,t\,] = \sum_{X=16,32,64,128,256,512,1024,2048,4096} \left(\frac{X}{4} * N_{\text{biAndGpmPredBlks\_}X}[\,t\,] \right) \tag{6-102}$$

Where $N_{\text{biAndGpmPredBlks\_}X}[\,t\,]$ are the number of blocks using inter bi-predicted prediction, for blocks with number of samples X=16, 32, 64, 128, 256, 512, 1024, 2048, 4096, in the slice[t] or tile[t] or subpicture[t].

$N_{\text{biAndGpmPredBlks}}[\,t\,]$ is derived from portion_bi_and_gpm_predicted_blocks_area[t] and $N_{\text{4SampleBlksInSegment}}[\,t\,]$ in the decoder.

portion_bdof_blocks_area[t] indicates the portion of area covered by inter blocks using BDOF in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity and is set equal to $P_{\text{bdofBlks}}[\,t\,]$ defined as follows:

$$P_{\text{bdofBlks}}[\,t\,] = \text{Floor}\left(\frac{N_{\text{bdofBlks}}[\,t\,]}{N_{\text{4SampleBlksInSegment}}[\,t\,]} * 255 \right) \tag{6-103}$$

$N_{\text{bdofBlks}}[\,t\,]$ is the number of inter blocks using BDOF in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity. At the encoder side, it is computed as follows:

$$N_{\text{bdofBlks}}[\,t\,] = \sum_{X=128,256,512,1024,2048,4096} \left(\frac{X}{4} * N_{\text{bdofBlks\_}X} \right) \tag{6-104}$$

Where $N_{\text{bdofBlks\_}X}[\,t\,]$ are the number of blocks using inter blocks using BDOF, for blocks with number of samples X=128, 256, 512, 1024, 2048, 4096, in the slice[t] or tile[t] or subpicture[t].

$N_{\text{bdofBlks}}[\,t\,]$ is derived from portion_bdof_blocks_area[t] and $N_{\text{4SampleBlksInSegment}}[\,t\,]$ in the decoder.

portion_deblocking_instances[t] indicates the portion of deblocking filtering instances in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity and is set equal to $P_{dbfInstances}[t]$ defined as follows:

$$P_{dbfInstances}[t] = Floor\left(\frac{N_{dbfInstances}[t]}{4*N_{4SampleBlksInSegment}[t]} * 255 \right) \quad (6\text{-}105)$$

$N_{dbfInstances}[t]$ is the number of deblocking filtering instances in the specified period. It is derived from portion_deblocking_instances[t] and $N_{4SampleBlksInSegment}[t]$ in the slice[t] or tile[t] or subpicture[t].

portion_sao_filtered_blocks[t] indicates the portion of SAO filtered blocks in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity. At the encoder side, it is set equal to $P_{saoBlks}[t]$ computed as follows:

$$P_{saoBlks}[t] = Floor\left(\frac{N_{saoFilteredBlks}[t]}{N_{4SampleBlksInSegment}[t]} * 255 \right) \quad (6\text{-}106)$$

$N_{saoFilteredBlks}[t]$ is the number of SAO filtered blocks in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity. It is derived from portion_sao_filtered_blocks[t], $N_{4SampleBlksInSegment}[t]$ in the decoder.

portion_alf_filtered_blocks[t] indicates the portion of ALF filtered blocks in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity. At the encoder side, it is set equal to $P_{alfBlks}[t]$ computed as follows:

$$P_{alfBlks}[t] = Floor\left(\frac{N_{alfFilteredBlks}[t]}{N_{4SampleBlksInSegment}[t]} * 255 \right) \quad (6\text{-}107)$$

$N_{alfFilteredBlks}[t]$ is the number of ALF filtered blocks in the slice[t] or tile[t] or subpicture[t], using 4-samples block granularity. It is derived from portion_alf_filtered_blocks[t], $N_{4SampleBlksInSegment}[t]$ in the decoder.

6.3 Interactive signalling for remote decoder-power reduction

6.3.1 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.

6.3.2 Syntax

The syntax for interactive signalling for remote decoder-power reduction is described in Table 11.

Table 11 — syntax for interactive signalling for remote decoder-power reduction

	Descriptor
dec_pow_reduction_type	u(2)
if (dec_pow_reduction_type = = 0) {	
dec_ops_reduction_req	s(6)
else if (dec_pow_reduction_type = = 1) {	
disable_loop_filters	u(1)
disable_bi_prediction	u(1)
disable_intra_in_B	u(1)
disable_fracpel_filtering	u(1)

Table 11 *(continued)*

user_defined_req	u(2)
}	
else if (dec_pow_reduction_type = = 2) {	
pic_width_in_luma_samples	u(14)
pic_height_in_luma_samples	u(14)
frames_per_second	u(10)
}	

6.3.3 Signalling

The transmitter in each device sends a decoding operation reduction request (DOR-Req) message to the attention of the remote encoder. In a first mode (dec_pow_reduction_type equal to 0), 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 matches the power saving implied by the DOR-Req message. In a second mode (dec_pow_reduction_type equal to 1), this message requests the remote encoder to disable coding tools so that, when the local decoder decodes the bitstream, the power consumption of the local decoder is decreased. In a third mode (dec_pow_reduction_type equal to 2), this message requests the remote encoder to adjust the picture resolution and video frame rate so that, when the local decoder decodes the bitstream, the power consumption of the local decoder is decreased.

6.3.4 Semantics

dec_pow_reduction_type indicates the type of the decoder power reduction method which is requested by the receiver. The type is indicated by an unsigned integer. The types are explained in Table 12.

Table 12 — definition of dec_pow_reduction_type

dec_pow_reduction_type	Definition
0	Decoder operations reduction
1	Coding tool configuration
2	Spatial and temporal scaling
3	Undefined

dec_ops_reduction_req indicates the requested variation 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. dec_ops_reduction_req is an integer in the interval [-31, 32]. When not present, dec_ops_reduction_req is set equal to 0.

$P_{\text{DecOpsReductionReq}}$ is derived by dec_ops_reduction_req and indicates the requested percentage change of local decoding operations by

$$P_{\text{DecOpsReductionReq}} = 2^{*} d_{\text{req}} \tag{6-108}$$

where d_{req} is set equal to dec_ops_reduction_req.

A negative percentage means a decrease of decoding operations. $P_{\text{DecOpsReductionReq}}$ is an integer in the interval [-62, 64] in steps of two.

disable_loop filters equal to 1 indicates that loop filters are requested to be disabled, disable_loop_filters equal to 0 specifies that loop filters are requested to be enabled. Loop filters include, upon availability, the deblocking filter, sample adaptive offset, and the adaptive loop filter.

disable_bi_prediction equal to 1 indicates bi-prediction is requested to be disabled in B slices. disable_bi_prediction equal to 0 indicates bi-prediction is requested to be enabled in B slices.

disable_intra_in_B equal to 1 indicates intra prediction is requested to be disabled in B slices. disable_intra_in_B equal to 0 indicates intra prediction is requested to be enabled in B slices.

disable_fracpel_filtering equal to 1 indicates fractional pel filtering is requested to be disabled in P slices or B slices. disable_fracpel_filtering equal to 0 indicates fractional pel filtering is requested to be enabled in P slices or B slices.

user_defined_req indicates a request to enable or disable user-defined coding tools.

pic_width_in_luma_samples indicates the requested picture width in the units of luma samples.

pic_height_in_luma_samples indicates the requested picture height in the units of luma samples.

frames_per_second indicates the requested frame rate.

7 Display power reduction using display adaptation

7.1 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.

7.2 Syntax

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

The message format used to send metadata from the transmitter to the receiver is described in Table 13.

Table 13 — syntax for display power reduction

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

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

The receiver first uses the message format to signal information to the transmitter described in Table 14.

Table 14 — syntax from receiver for display power reduction

	Descriptor
constant_backlight_voltage_time_interval	u(16)
max_variation	u(8)

The transmitter then uses the message format to signal metadata to the receiver described in Table 15.

Table 15 — syntax to receiver for display power reduction

	Descriptor
num_quality_levels	u(4)
lower_bound	u(8)
if (lower_bound > 0)	
upper_bound	u(8)
rgb_component_for_infinite_psnr	u(8)
for (i = 1; i <= num_quality_levels; i++) {	
max_rgb_component[i]	u(8)
scaled_psnr_rgb[i]	u(8)
}	

7.3 Signalling

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

Green metadata can be carried as specified in ISO/IEC 13818-1 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 7.2.1, 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.

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

Using the first message format described in 7.2.2, the receiver first signals constant_backlight_voltage_time_interval and max_variation to the transmitter. The transmitter then uses the second message format in 7.2.2 to send a message to the receiver. The DA metadata is applicable to the presentation subsystem until the next message containing DA metadata arrives.

7.4 Semantics

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

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

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

max_variation[j] indicates 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 $V_{\text{BacklightScalingFactor}}[\,k\,][\,j\,][\,i\,]$ for that frame. $V_{\text{BacklightScalingFactor}}[\,k\,][\,j\,][\,i\,]$ is derived from max_rgb_

component[k][j][i] and the peak signal variable P_S , as (max_rgb_component[k][j][i] / P_S) for the k^{th} constant_backlight_voltage_time_interval, j^{th} max_variation and i^{th} quality level.

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.

constant_backlight_voltage_time_interval[k] indicates the minimum time interval, in milliseconds, that shall 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.

lower_bound[k][j] indicates 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] indicates 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 P_S. The RGB components in the range (lower_bound[k][j], upper_bound[k][j]) are mapped linearly onto the range (0, P_S).

rgb_component_for_infinite_psnr[k][j] indicates for the k^{th} constant_backlight_voltage_time_interval and j^{th} max_variation, the largest RGB component in the reconstructed frames. Therefore, the scaled frames $F_{ScaledFrames}$[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 $F_{ScaledFrames}$[k][j][0] have a PSNR of infinity relative to the reconstructed frames.

max_rgb_component[k][j][i] indicates for the k^{th} constant_backlight_voltage_time_interval, j^{th} max_variation and i^{th} quality level, the maximum RGB component that is 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_psnr_rgb[k][j][i] indicates the PSNR of $F_{ScaledFrames}$[k][j][i] relative to the reconstructed frames. $F_{ScaledFrames}$[k][j][i] are 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.

scaled_psnr_rgb[k][j][i] is set equal to the PSNR value v_{PSNR}, defined as follows for 0 < i <= num_quality_levels:

$$v_{\mathrm{PSNR}} = \mathrm{Clip}\left(\mathrm{Round}\left(10\,\mathrm{Log}_{10}\left(\frac{{P_S}^2 * w * h * N_{\mathrm{colour}} * N_{\mathrm{frames}}}{\sum_{n=1}^{N_{\mathrm{frames}}} \sum_{c=1}^{N_{\mathrm{colour}}} \sum_{l=X_s+1}^{P_S} \mathrm{N_{c,n}}(l) * \left(l - X_s\right)^2} \right) \right) \right) \tag{7-1}$$

where

— w is the width of a video frame.

— h is the height of a video frame.

— N_{colour} is the number of colour channels. For RGB colourspace, N_{colour} = 3.

— N_{frames} is the number of frames in the reconstructed frames.

— $N_{c,n}$ (l) is the number of RGB components samples 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.

8 Energy-efficient media selection

8.1 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 (as defined in ISO/IEC 23009-1:2022, 3.1.40);

— 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 7.4, 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.

8.2 Syntax

The decoder-power indication metadata is a pair of decoder operations reduction ratios. The syntax is described in Table 16.

Table 16 — syntax for decoder-power indication

	Descriptor
dec_ops_reduction_ratio_from_max	u(8)
dec_ops_reduction_ratio_from_prev	s(16)

The display-power indication metadata contains a list of ms_num_quality_levels pairs. The syntax is described in Table 17.

Table 17 — syntax for display-power indication

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

8.3 Signalling

Green metadata may 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 and ISO/IEC/TR 23009-3[4] and illustrated in Annex B.

8.4 Semantics

8.4.1 Decoder-power indication metadata semantics

dec_ops_reduction_ratio_from_max(i) indicates the percentage by which decoding operations are reduced in the ith representation compared to the most demanding representation of the current video Segment. dec_ops_reduction_ratio_from_max(i) is set equal to $d_{\text{opsReducRatioFromMax}}$(i) derived as follows:

$$d_{\text{opsReducRatioFromMax}}(\text{i}) = \text{Floor}\left(\frac{N_{\text{MaxNumDecOps}} - N_{\text{DecOps}}(\text{i})}{N_{\text{MaxNumDecOps}}} * 100\right) \tag{8-1}$$

$N_{\text{maxNumDecOps}}$ is the estimated number of decoding operations required for the most demanding representation of the current video Segment.

N_{DecOps}(i) is the estimated number of decoding operations required for the ith representation of the current video Segment.

dec_ops_reduction_ratio_from_prev(i) indicates the percentage by which decoding operations are reduced in the current video Segment compared to the previous video Segment for the ith representation in a given Period (as defined in ISO/IEC 23009-1:2022, 3.1.34). A negative value means an increase in decoding operations. dec_ops_reduction_ratio_from_prev(i) is set equal to $d_{\text{opsReducRatioFromPrev}}$(i) derived as follows:

$$d_{\text{opsReducRatioFromPrev}}(\text{i}) = \text{Floor}\left(\frac{N_{\text{PrevDecOps}}(\text{i}) - N_{\text{DecOps}}(\text{i})}{N_{\text{DecOps}}(\text{i})} * 100\right) \tag{8-2}$$

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

$N_{\text{PrevDecOps}}$(i) is the estimated number of decoding operations required for the ith representation of the previous video Segment in a given Period. If the current video Segment is the first Segment of a Period, then $N_{\text{PrevDecOps}}(\text{i}) = N_{\text{DecOps}}(\text{i})$.

8.4.2 Display-power indication metadata semantics

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

ms_rgb_component_for_infinite_psnr indicates the average, over the *N* reconstructed frames of the video Segment, of the largest RGB component in each of the reconstructed frames.

ms_max_rgb_component[i] indicates for the ith quality level (1 <= i <= num_quality_levels), the average, over the *N* reconstructed frames of the video Segment, of the maximum RGB component that is 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] indicates for the ith 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 7.4, 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.

9 Metrics for quality recovery after low-power encoding

9.1 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 associated picture or subpicture is delivered as metadata to the decoder. The metric is utilized, by the decoder, in conjunction with the associated frame or subpicture 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.

9.2 Syntax

9.2.1 AVC and HEVC syntax

For AVC and HEVC bitstreams, the encoder embeds the metadata message in the last picture of each Segment using the syntax of Table 18.

Table 18 — syntax for quality metrics for AVC and HEVC

	Descriptor
xsd_metric_type	u(8)
xsd_metric_value	u(16)

9.2.2 VVC syntax

For VVC bitstreams, the encoder embeds the metadata message in the associated picture using the syntax of Table 19.

Table 19 — syntax for quality metrics for VVC

	Descriptor
xsd_subpic_number_minus1	u(16)
xsd_subpic_id[i]	u(16)
xsd_metric_number_minus1[i]	u(8)
xsd_metric_type[i][j]	u(8)
xsd_metric_value[i][j]	u(16)

9.3 Signalling

SEI messages can be used to signal green metadata in an AVC, HEVC or VVC bitstream. The green metadata SEI message payload type for AVC is specified in ISO/IEC 14496-10. The green metadata SEI message payload type for HEVC is specified in International Standard ISO/IEC 23008-2. The green metadata SEI message payload type for VVC is specified in International Standard ISO/IEC 23090-3.

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

9.4 Semantics

9.4.1 AVC and HEVC Semantics

xsd_metric_type indicates the type of the objective quality metric as shown in the Table 20. PSNR, as defined in ISO/IEC 23001-10, is the only type currently supported. A definition of PSNR is provided in Annex D.

Table 20 — specification of xsd_metric_type for AVC and HEVC

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 V_{PSNR} value (in dB) as follows, with m set equal to xsd_metric_value:

$$V_{PSNR} = \frac{m}{100} \tag{9.1}$$

9.4.2 VVC Semantics

xsd_subpic_number_minus1 plus 1 indicates the number of subpictures associated with the metrics specified in the SEI message. The value of xsd_subpics_number_minus1[i] shall be in the range of 0 to MaxSlicesPerAu – 1, inclusive, where MaxSlicesPerAu is defined in ISO/IEC 23090-3.

xsd_subpic_id[i] indicates the subpicture ID of the i^{th} subpicture.

xsd_metric_number_minus1[i] plus 1 indicates the number of objective quality metrics associated with the i^{th} subpicture.

xsd_metric_type[i][j] indicates the type of the j^{th} objective quality metric associated with the i^{th} subpicture as shown in Table 21. PSNR, wPSNR, WS-PSNR and SSIM are the only types currently supported. Definitions of PSNR, wPSNR, WS-PSNR and SSIM are provided in Annex D.

Table 21 — specification of xsd_metric_type for VVC

Value	Description
0x00	PSNR
0x01	SSIM
0x02	wPSNR
0x03	WS-PSNR
0x04-0xFF	User-defined

xsd_metric_value[i][j] contains the value of the j^{th} objective quality metric associated with the i^{th} subpicture. When xsd_subpic_number_minus1 is equal to 0, xsd_metric_value[0][j] contains the value of j^{th} objective quality metric of the associated picture.

When xsd_metric_type[i][j] is 0, then the stored 16-bit unsigned integer xsd_metric_value[i][j], is interpreted as a floating-point V_{PSNR} value (in dB) as follows, with m set equal to xsd_metric_value[i][j]:

$$V_{\text{PSNR}} = \frac{m}{100} \tag{9.2}$$

When xsd_metric_type[i] is 1, then the stored 16-bit unsigned integer xsd_metric_value[i][j], is interpreted as a floating-point V_{SSIM} value as follows, with m set equal to xsd_metric_value[i][j]:

$$V_{\text{SSIM}} = \frac{m}{100} \tag{9.3}$$

When xsd_metric_type[i] is 2, then the stored 16-bit unsigned integer xsd_metric_value[i][j], is interpreted as a floating-point V_{wPSNR} value (in dB) as follows, with m set equal to xsd_metric_value[i][j]:

$$V_{\text{wPSNR}} = \frac{m}{100} \tag{9.4}$$

When xsd_metric_type[i] is 3, then the stored 16-bit unsigned integer xsd_metric_value[i][j], is interpreted as a floating-point $V_{\text{WS-PSNR}}$ value (in dB) as follows, with m set equal to xsd_metric_value[i][j]:

$$V_{\text{WS–PSNR}} = \frac{m}{100} \tag{9.5}$$

10 Conformance and reference software

Conformance and reference software for green metadata shall be used as specified in Annex C.

Annex A
(normative)

Supplemental Enhancement Information (SEI) syntax

A.1 Syntax and semantics of green metadata SEI message carried in AVC NAL units

This clause describes the payload syntax and semantics if payloadType 56 appears in an AVC NAL unit with nal_unit_type set to 6.

A.1.1 Syntax

green_metadata(payload_size) {	**Descriptor**
green_metadata_type	u(8)
switch (green_metadata_type) {	
case 0:	
period_type	u(8)
if (period_type = = 2) || (period_type <= 7) {	
num_seconds	u(16)
}	
else if (period_type = = 3) || (period_type = = 8) {	
num_pictures	u(16)
}	
if (period_type = = 8) {	
temporal_map	u(8)
for (t=0; t<8; t++) {	
if ((temporal_map>>t)%2 = = 1)	
num_pictures_in_temporal_layers[t]	u(16)
}	
}	
if (period_type <= 3) {	
portion_non_zero_8x8_blocks	u(8)
portion_intra_predicted_macroblocks	u(8)
portion_six_tap_filterings	u(8)
portion_alpha_point_deblocking_instances	u(8)
}	
else if (period_type = = 4) {	
for (i=0; i<= num_slice_groups_minus1; i++) {	
num_slices_minus1[i]	u(16)
}	
for (i=0; i<= num_slice_groups_minus1; i++) {	
for (j=0; j<=num_slices_minus1[i]; j++) {	
first_mb_in_slice[i][j]	u(16)
portion_non_zero_8x8_blocks[i][j]	u(8)

portion_intra_predicted_macroblocks[i][j]	u(8)
portion_six_tap_filterings[i][j]	u(8)
portion_alpha_point_deblocking_instances[i][j]	u(8)
}	
}	
}	
else if (period_type >= 5) && (period_type <= 8) {	
num_layers_minus1	u(16)
for (l=0; l<= num_layers_minus1; l++) {	
picture_parameter_set_id[l]	u(8)
priority_id[l]	u(6)
dependency_id[l]	u(3)
quality_id[l]	u(4)
temporal_id[l]	u(3)
portion_non_zero_8x8_blocks[l]	u(8)
portion_intra_predicted_macroblocks[l]	u(8)
portion_six_tap_filterings[l]	u(8)
portion_alpha_point_deblocking_instances[l]	u(8)
}	
}	
break;	
case 1:	
xsd_metric_type	u(8)
xsd_metric_value	u(16)
break;	
default:	
}	
}	

A.1.2 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. Other values of green_metadata_type are reserved for future use by ISO/IEC.

A.2 Syntax and semantics of green metadata SEI message carried in HEVC NAL units

This clause describes the payload syntax and semantics if payloadType 56 appears in an HEVC NAL unit with nal_unit_type set to PREFIX_SEI_NUT.

A.2.1 Syntax

green_metadata(payload_size) {	**Descriptor**
green_metadata_type	u(8)
switch (green_metadata_type) {	
case 0:	
period_type	u(8)

if (period_type = = 2) {	
num_seconds	u(16)
}	
else if (period_type = = 3) {	
num_pictures	u(16)
}	
if (period_type <= 3) {	
portion_non_zero_blocks_area	u(8)
if (portion_non_zero_blocks_area != 0) {	
portion_8x8_blocks_in_non_zero_area	u(8)
portion_16x16_blocks_in_non_zero_area	u(8)
portion_32x32_blocks_in_non_zero_area	u(8)
}	
portion_intra_predicted_blocks_area	u(8)
if (portion_intra_predicted_blocks_area = = 255) {	
portion_planar_blocks_in_intra_area	u(8)
portion_dc_blocks_in_intra_area	u(8)
portion_angular_hv_blocks_in_intra_area	u(8)
}	
else {	
portion_blocks_a_c_d_n_filterings	u(8)
portion_blocks_h_b_filterings	u(8)
portion_blocks_f_i_k_q_filterings	u(8)
portion_blocks_j_filterings	u(8)
portion_blocks_e_g_p_r_filterings	u(8)
}	
portion_deblocking_instances	u(8)
}	
else if(period_type = = 4) {	
max_num_slices_tiles_minus1	u(16)
for (t=0; t<=max_num_slices_tiles_minus1; t++) {	
first_ctb_in_slice_or_tile[t]	u(16)
portion_non_zero_blocks_area[t]	u(8)
if (portion_non_zero_blocks_area[t] != 0) {	
portion_8x8_blocks_in_non_zero_area[t]	u(8)
portion_16x16_blocks_in_non_zero_area[t]	u(8)
portion_32x32_blocks_in_non_zero_area[t]	u(8)
}	
portion_intra_predicted_blocks_area[t]	u(8)
if (portion_intra_predicted_blocks_area[t] = = 255	
portion_planar_blocks_in_intra_area[t]	u(8)
portion_dc_blocks_in_intra_area[t]	u(8)
portion_angular_hv_blocks_in_intra_area[t]	u(8)
}	
else {	
portion_blocks_a_c_d_n_filterings[t]	u(8)
portion_blocks_h_b_filterings[t]	u(8)
portion_blocks_f_i_k_q_filterings[t]	u(8)

portion_blocks_j_filterings[t]	u(8)
portion_blocks_e_g_p_r_filterings[t]	u(8)
}	
portion_deblocking_instances[t]	u(8)
}	
}	
break;	
case 1:	
xsd_metric_type	u(8)
xsd_metric_value	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. Other values of green_metadata_type are reserved for future use by ISO/IEC.

A.3 Syntax and semantics of green metadata SEI message carried in VVC NAL units

This clause describes the payload syntax and semantics if payloadType 56 appears in a VVC NAL unit with nal_unit_type set to PREFIX_SEI_NUT.

A.3.1 Syntax

green_metadata(payload_size) {	**Descriptor**
green_metadata_type	u(8)
switch (green_metadata_type) {	
case 0:	
period_type	u(4)
granularity_type	u(3)
extended_representation_flag	u(1)
if (period_type = = 2) {	
num_seconds	u(16)
}	
else if (period_type = = 3) {	
num_pictures	u(16)
}	
if (granularity_type = = 0) {	
portion_non_zero_blocks_area	u(8)
portion_non_zero_transform_coefficients_area	u(8)
portion_intra_predicted_blocks_area	u(8)
portion_deblocking_instances	u(8)
portion_alf_instances	u(8)
if (extended_representation_flag) {	

if (portion_non_zero_blocks_area != 0) {	
portion_non_zero_4_8_16_blocks_area	u(8)
portion_non_zero_32_64_128_blocks_area	u(8)
portion_non_zero_256_512_1024_blocks_area	u(8)
portion_non_zero_2048_4096_blocks_area	u(8)
}	
if (portion_intra_predicted_blocks_area < 255) {	
portion_bi_and_gpm_predicted_blocks_area	u(8)
portion_bdof_blocks_area	u(8)
}	
portion_sao_instances	u(8)
}	
}	
else if(granularity_type <= 3) {	
max_num_segments_minus1	u(16)
for (t=0; t<= max_num_segments_minus1; t++) {	
segment_address[t]	u(16)
portion_non_zero_blocks_area[t]	u(8)
portion_non_zero_transform_coefficients_area[t]	u(8)
portion_intra_predicted_blocks_area[t]	u(8)
portion_deblocking_instances[t]	u(8)
portion_alf_filtered_blocks[t]	u(8)
if (extended_representation_flag) {	
if (portion_non_zero_blocks_area[t] != 0) {	
portion_non_zero_4_8_16_blocks_area[t]	u(8)
portion_non_zero_32_64_128_blocks_area[t]	u(8)
potion_non_zero_256_512_1024_blocks_area[t]	u(8)
portion_non_zero_2048_4096_blocks_area[t]	u(8)
}	
if (portion_intra_predicted_blocks_area[t] < 255) {	
portion_bi_predicted_blocks_area[t]	u(8)
portion_bdof_block_area[t]	u(8)
}	
portion_sao_filtered_blocks[t]	u(8)
}	
}	
}	
break;	
case 1:	
xsd_subpic_number_minus1	u(16)
for (i=0; i<= xsd_subpic_number_minus1; i++) {	
xsd_subpic_idc[i]_	u(16)
xsd_metric_number_minus1[i]	u(8)
for (j=0; j<= xsd_metric_number_minus1[i]; j++) {	
xsd_metric_type[i][j]	u(8)
xsd_metric_value[i][j]	u(16)
}	

`      }`	
`      break;`	
`    default:`	
`   }`	
`}`	

A.3.2 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. Other values of green_metadata_type are reserved for future use by ISO/IEC.

Annex B
(informative)

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 achieve power reduction.

B.1.2 Derivation of the complexity metrics

6.2.2 specifies CMs associated to AVC: portion_non_zero_8x8_blocks, portion_intra_predicted_macroblocks, portion_six_tap_filterings and portion_alpha_point_deblocking_instances. The computation of the first two CMs, as explained in 6.2.4, is straightforward. However, computation of portion_six_tap_filterings and portion_alpha_point_deblocking_instances is more involved. To provide a better understanding of these two CMs, the next two subclauses describe how $N_{\text{maxNumSixTapFiltPic}}(i)$ and $N_{\text{maxAlphaPointDbfsPic}}(i)$ are derived.

6.2.2 specifies CMs associated to HEVC: portion_blocks_a_c_d_n_filterings, portion_blocks_h_b_filterings, portion_blocks_f_i_k_q_filterings, portion_blocks_j_filterings and portion_blocks_e_g_p_r_filterings. To provide a better understanding of these five CMs, the different sub-sample position a, b, c, d, e, f, g, h, i, j, k, n, p, q and r, are represented in Figure B.1, where upper-case letters represent integer samples and lower-case letters represent sub-sample positions derived.

6.2.2 specifies CMs associated to VVC: portion_non_zero_blocks_area, portion_non_zero_4_8_16_blocks_area, portion_non_zero_32_64_128_blocks_area, portion_non_zero_256_512_1024_blocks_area, portion_non_zero_2048_4096_blocks_area, portion_non_zero_transform_coefficients_area, portion_intra_predicted_blocks_area, portion_bi_and_gpm_predicted_blocks_area, portion_bdof_blocks_area, portion_deblocking_instances, portion_sao_filtered_blocks, portion_alf_filtered_blocks.

B.1.2.1 Deriving the worst-case, largest value for $N_{\text{maxNumSixTapFiltPic}}(i)$

To determine $N_{\text{maxNumSixTapFiltPic}}(i)$, 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 6-tap filterings (STFs) occurs in a picture when all partitions consist of 4x4 blocks that are 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 shall 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 shall 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 shall compute 16 points for each of the other fractional-sample positions (b_G, c_G, ..., r_G) that the MV can 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 can point to.

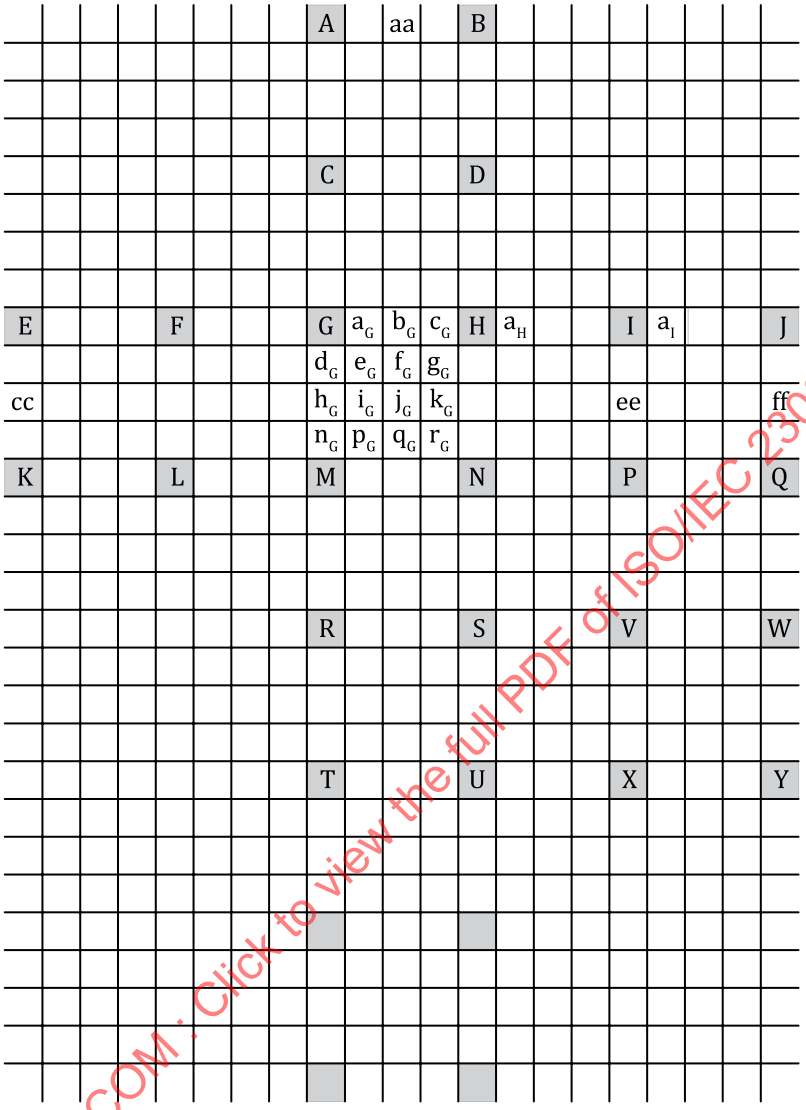

Figure B.1 — Quarter-sample interpolation of the 4x4-block consisting of samples G, H, I, J, M, N, P, Q, R, S, V, W, T, U, X, Y

a) If the MV points to b_G, then to interpolate b_G, the decoder shall 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.

b) If the MV points to h_G, then to interpolate h_G, the decoder shall 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.

c) If the MV points to j_G, then to interpolate j_G, the decoder shall 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.

   1) 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).

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

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

4) 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.

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

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

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

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

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

i) 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 (c), column GMRT was analysed first and the analysis was then repeated for the other 3 columns (HNSU, IPVX, JQWY).

j) 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.

k) 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.

l) 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 (a), (b)). Therefore 32 STFs are needed to compute e_G, ..., e_Y for the 4x4 block.

m) 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.

n) 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.

o) If the MV points to r_G, then to interpolate r_G, the decoder needs 2 STFs to get m_G and $_{sG}$. Therefore, 32 STFs are needed to compute r_G, ..., r_Y for the 4x4 block.

From (a),...,(n), 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 derived based on the following pseudo-code:

```
N_MaxNumSixTapFiltPic(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 * PicSizeInMbs * 16
                         = 1664 * PicSizeInMbs                                    (B-1)
```

B.1.2.2 Deriving the worst-case, largest value for $N_{\text{maxAlphaPointDbfsPic}}(i)$

To determine $N_{\text{maxAlphaPointDbfsPic}}(i)$, 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 occurs on each row of the edge because the q_0 samples 1, 17, ... 241 will all be APDIs. Therefore, 16 APDIs occur in vertical edge (0, I). Similarly, when vertical edge (IV, V) is filtered, there are also 16 APDIs corresponding to the 16 (p_0, q_0) sample pairs (20, 21), (36, 37), ... (244, 245). Thus, there are 16*4 = 64 APDIs from vertical-edge filtering. After horizontal-edge filtering, there are an additional 64 APDIs because each horizontal edge contributes 16 APDIs. For example, horizontal edge (viii, ix) contributes 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

a) 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.

b) 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.

c) 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.

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

To conclude, since each picture has PicSizeInMbs macroblocks, the worst-case number of APDIs per picture, is derived based on the following pseudo-code:

$$
\begin{aligned}
N_{\texttt{maxAlphaPointDbfsPic}}(\texttt{i}) &= \texttt{PicSizeInMbs} * (128 + 64) = 192 * \texttt{PicSizeInMbs, for 4:2:0,} \\
&= \texttt{PicSizeInMbs} * (128 + 128) = 256 * \texttt{PicSizeInMbs, for 4:2:2,} \\
&= \texttt{PicSizeInMbs} * (128 + 256) = 384 * \texttt{PicSizeInMbs, for 4:4:4,} \\
&= 128 * \texttt{PicSizeInMbs, for a single colour plane.}
\end{aligned} \tag{B-2}
$$

B.1.3 Example usage of C-DVFS metadata

C-DVFS metadata may be signalled at a slice, layer, 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 B.3 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 portion_non_zero_8x8_blocks, portion_intra_predicted_macroblocks, portion_six_tap_filterings and portion_num_alpha_point_deblocking_instances. From these portion values and the corresponding worst-case instances the four CMs are derived: num_non_zero_8x8_blocks (n_{nz}), num_intra_predicted_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_{pict}) is estimated or predicted according to Formula B-3:

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

where C_{pict} 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 slice-by-slice, layer-by-layer, 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 do not 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 slice, layer, 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 $k_{\alpha} * n_{\alpha}$.

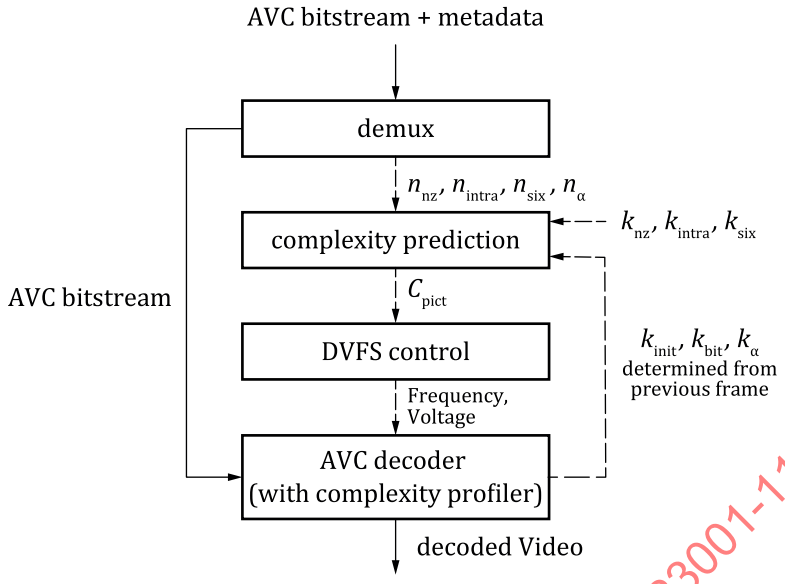

Figure B.3 — Example of parsing, complexity prediction, and DVFS control

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 7.2.1 may be stored using the file format specified in ISO/IEC 23001-10 or the metadata may be carried by MPEG-2 systems as specified in ISO/IEC 13818-1.

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 $F_{ScaledFrames}$[i] is estimated by saturating all RGB components of reconstructed frames to max_rgb_component[i]. The $F_{ScaledFrames}$[i] thus obtained are what would be perceived at the display after the receiver scales the RGB components of reconstructed frames by (P_S / max_rgb_component[i]), P_S being the peak signal and then applies the backlight scaling factor, b = (max_rgb_component[i] / P_S) to the LCD backlight. scaled_psnr_rgb[i] is computed at the transmitter using P_S and by assuming that the noise is the difference between $F_{ScaledFrames}$[i] and reconstructed frames accumulated over R, G and B components, as explained in 7.4.

The receiver examines the (num_quality_levels + 1) pairs of metadata and selects the pair (max_rgb_component[$i_{Selected}$], scaled_psnr_rgb[$i_{Selected}$]) for which scaled_psnr_rgb[$i_{Selected}$] is an acceptable quality level. Then, the receiver derives DA scaling factors from max_rgb_component[$i_{Selected}$]. Finally, the display scales the RGB components of reconstructed frames by P_S / max_rgb_component[$i_{Selected}$] and it scales the backlight or voltage level by max_rgb_component[$i_{Selected}$] / P_S. After backlight scaling, the displayed pixels are perceived as $F_{ScaledFrames}$[$i_{Selected}$]. 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[$i_{Selected}$] so that RBLL[$i_{Selected}$ − 1] > RBLLactual > RBLL[$i_{Selected}$], where RBLL[0] = 100 %. For each frame to be displayed, the device examines the display-adaptation metadata

and selects the pair indexed by $j_{Selected}$ for which Q[$i_{Selected}$ – 1] > scaled_psnr_rgb[$j_{Selected}$] > Q[$i_{Selected}$], where Q[0] = infinity. The metadata max_rgb_component[$j_{Selected}$] is then used to determine display-adaptation scaling parameters. Thus, the device implements 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] / P_S) 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] / P_S), and for each RGB component, x, of reconstructed frames, the scaling to S(x) is performed using the following pseudo-code:

```
S(x) = 0,                                                  for x in [0, lower_bound],
     = P_S *(x-lower_bound)/(upper_bound-lower_bound)       for x in (lower_bound, upper_
bound),
     = P_S                                                 for x in [upper_bound, P_S]
```

Observe that the interval (lower_bound, upper_bound) is mapped to the interval (0, P_S). 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 * P_S). Therefore, for RGB components within the interval (lower_bound, upper_bound), the perceived contrast enhancement is proportional to (b * P_S / (upper_bound – lower_bound)). This expression simplifies to b / (upper_bound – lower_bound), because P_S is a constant. For RGB components within the intervals [0, lower_bound] and [upper_bound, P_S], all contrast is lost because these intervals are mapped to 0 and P_S, 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 $V_{BacklightScalingFactor}$ corresponding to the lowest quality level as b = max_rgb_component[num_quality_levels] / P_S. 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 < P_S; lower_bound++){
      for (upper_bound = lower_bound; upper_bound < P_S; 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 are not 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 is 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 shall 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 shall 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 shall signal to the transmitter, constant_backlight_voltage_time_interval, the time interval over which the backlight (or voltage) shall 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 are widely used and it is 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 7.2.2 and 7.3.2. 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 $V_{\text{BacklightScalingFactor}}$ in 7.4) of successive video frames. To avoid such flicker, a transmitter may use the function adjust_backlight() 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 shall 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 $V_{\text{BacklightScalingFactor}}$ 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 $V_{\text{BacklightScalingFactor}}$ 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 V_BacklightScalingFactor 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 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 does 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 are widely used and it is 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 7.3.2. 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.

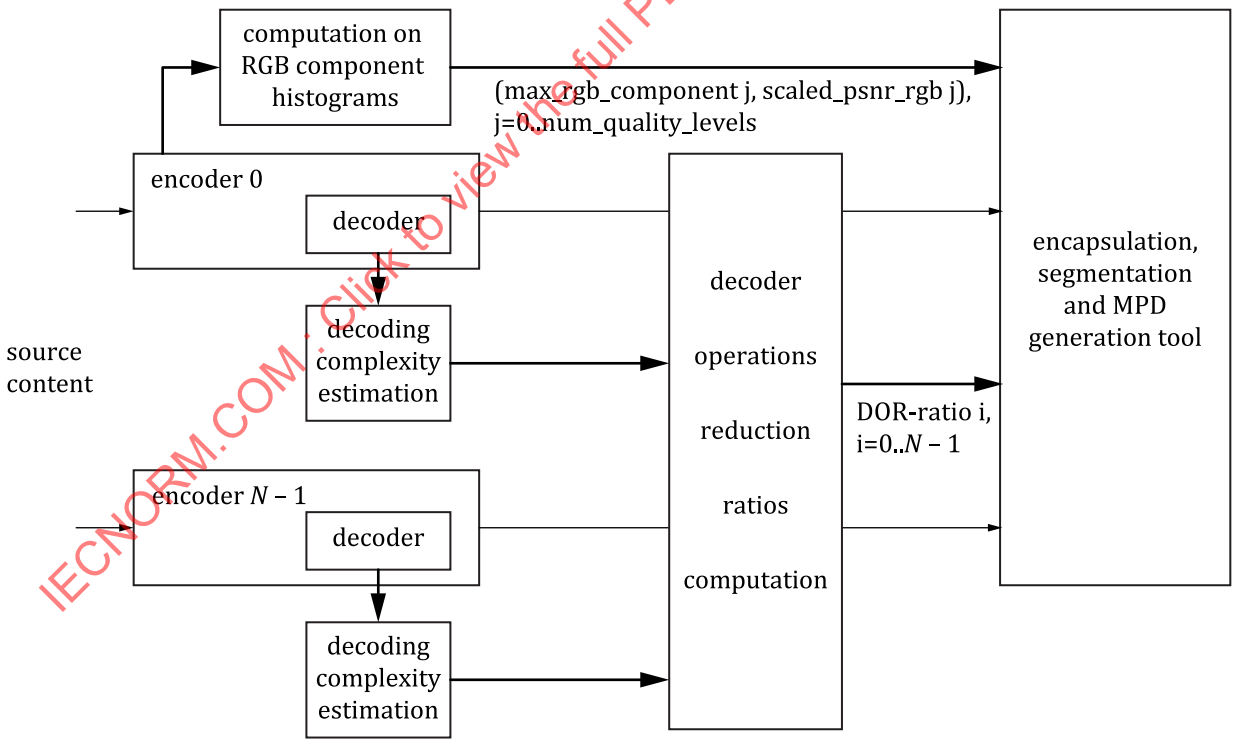

Figure B.4 — Green metadata computation and insertion

The DOR-Ratio-Max(i) associated with each video representation i of a Segmentis computed as the power-saving ratio from the most demanding video representation produced for the Segment, as defined in 8.4.1.

The DOR-Ratio-Prev(i) associated with each video representation i of a Segmentis computed as the power-saving ratio from the previous Segmentof the same representation, as defined in 8.4.1.

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 are considered as metadata representations. The available metadata representations are 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 Segmentand its associated media Segment(s) are time aligned on Segmentboundaries.

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

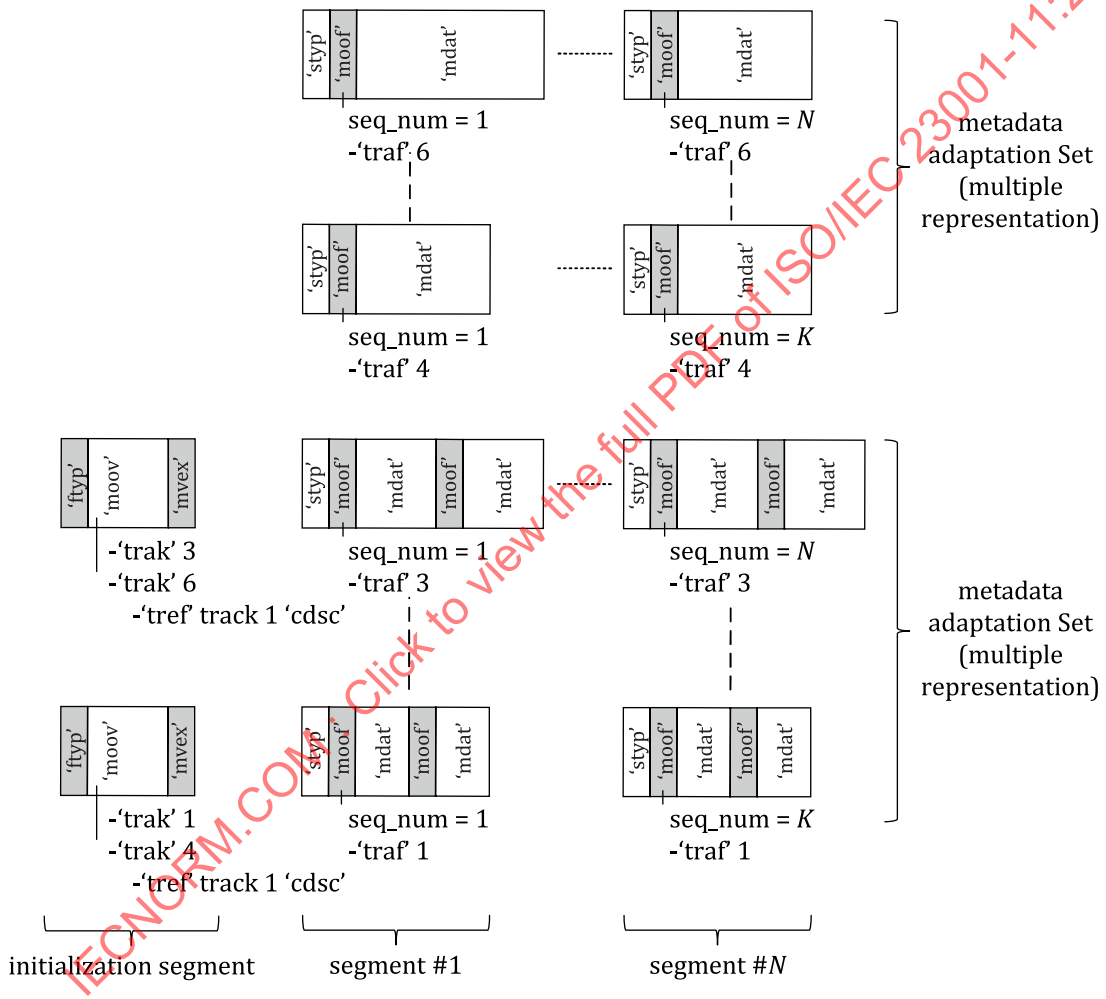

Figure B.5 — One metadata representation for one media representation

The following XML file provides an example of an MPD for decoder-power indication metadata:

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="urn:mpeg:DASH:schema:MPD:XXXX"
  xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:xxxx"
  type="dynamic"
  minimumUpdatePeriod="PT2S"
  timeShiftBufferDepth="PT30M"
  availabilityStartTime="2011-12-25T12:30:00"
  minBufferTime="PT4S"
```

```
  profiles="urn:mpeg:dash:profile:isoff-live:2011">

  <BaseURL>http://cdn1.example.com/</BaseURL>
  <BaseURL>http://cdn2.example.com/</BaseURL>

  <Period>
    <!-- Video -->
    <AdaptationSet
      id="video"
      mimeType="video/mp4"
      codecs="avc1.4D401F"
      frameRate="30000/1001"
      segmentAlignment="true"
      startWithSAP="1">
      <BaseURL>video/</BaseURL>
      <SegmentTemplate timescale="90000" media="$Bandwidth$/$Time$.mp4v">
        <SegmentTimeline>
          <S t="0" d="180180" r="432"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="v0" width="320" height="240" bandwidth="250000"/>
      <Representation id="v1" width="640" height="480" bandwidth="500000"/>
      <Representation id="v2" width="960" height="720" bandwidth="1000000"/>
    </AdaptationSet>
    <!-- English Audio -->
    <AdaptationSet mimeType="audio/mp4" codecs="mp4a.0x40" lang="en" segmentAlignment="0">
      <SegmentTemplate timescale="48000" media="audio/en/$Time$.mp4a">
        <SegmentTimeline>
          <S t="0" d="96000" r="432"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="a0" bandwidth="64000" />
    </AdaptationSet>
    <!-- French Audio -->
    <AdaptationSet mimeType="audio/mp4" codecs="mp4a.0x40" lang="fr" segmentAlignment="0">
      <SegmentTemplate timescale="48000" media="audio/fr/$Time$.mp4a">
        <SegmentTimeline>
          <S t="0" d="96000" r="432"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="a0" bandwidth="64000" />
    </AdaptationSet>
<!--AdaptationSet carrying Green Video Information for Video  -->
    <AdaptationSet id="green_video" codecs="depi"/>
      <BaseURL>video_green_depi/</BaseURL>
      <SegmentTemplate timescale="90000" media="$id$/$Time$.mp4m">
        <SegmentTimeline>
          <S t="0" d="180180" r="432"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="gv0" bandwidth="1000" associationId="v0"
associationType="cdsc"/>
      <Representation id="gv1" bandwidth="1000" associationId="v1"
associationType="cdsc"/>
      <Representation id="gv2" bandwidth="1000" associationId="v2"
associationType="cdsc"/>
    </AdaptationSet>
  </Period>

</MPD>
```

The display-power indication metadata is a list of (ms_num_quality_levels + 1) pairs of the form (ms_max_rgb_component[i], ms_scaled_psnr_rgb[i]) as defined in 8.4.1. This metadata is produced without considering any constraint on max_variation, the maximal backlight variation between two successive frames. It is also assumed that the backlight can be updated on each frame so that constant_backlight_voltage_time_interval is the inter-frame interval. Therefore, the display power-indication metadata provides the maximum power saving for a given quality level.

The display-power indication metadata is stored in a specific metadata file "id/$Time$.mp4m" (one for each Segment) using the format specified in ISO/IEC 23001-10. The display-power indication metadata representation is associated with all the available media representations as shown in Figure B.6.

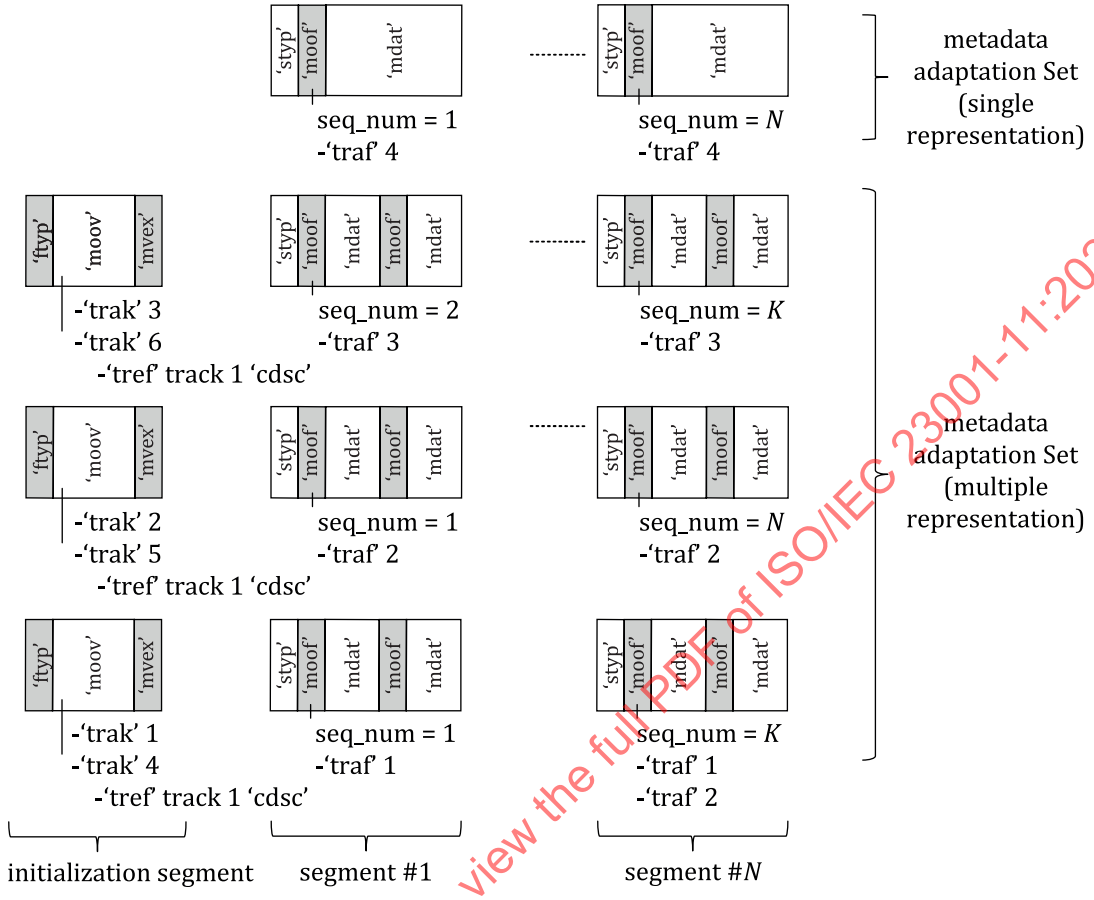

Figure B.6 — One metadata representation for all media representations

The following XML file provides an example of an MPD for display-power indication metadata:

```
<?xml version="1.0" encoding="UTF-8"?>
<MPD
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="urn:mpeg:DASH:schema:MPD:XXXX"
  xsi:schemaLocation="urn:mpeg:DASH:schema:MPD:xxxx"
  type="dynamic"
  minimumUpdatePeriod="PT2S"
  timeShiftBufferDepth="PT30M"
  availabilityStartTime="2011-12-25T12:30:00"
  minBufferTime="PT4S"
  profiles="urn:mpeg:dash:profile:isoff-live:2011">

  <BaseURL>http://cdn1.example.com/</BaseURL>
  <BaseURL>http://cdn2.example.com/</BaseURL>

  <Period>
    <!-- Video -->
    <AdaptationSet
      id="video"
      mimeType="video/mp4"
      codecs="avc1.4D401F"
      frameRate="30000/1001"
      segmentAlignment="true"
      startWithSAP="1">
      <BaseURL>video/</BaseURL>
```

```
      <SegmentTemplate timescale="90000" media="$Bandwidth$/$Time$.mp4v">
        <SegmentTimeline>
          <S t="0" d="180180" r="432"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="v0" width="320" height="240" bandwidth="250000"/>
      <Representation id="v1" width="640" height="480" bandwidth="500000"/>
      <Representation id="v2" width="960" height="720" bandwidth="1000000"/>
    </AdaptationSet>
    <!-- English Audio -->
    <AdaptationSet mimeType="audio/mp4" codecs="mp4a.0x40" lang="en" segmentAlignment="0">
      <SegmentTemplate timescale="48000" media="audio/en/$Time$.mp4a">
        <SegmentTimeline>
          <S t="0" d="96000" r="432"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="a0" bandwidth="64000" />
    </AdaptationSet>
    <!-- French Audio -->
    <AdaptationSet mimeType="audio/mp4" codecs="mp4a.0x40" lang="fr" segmentAlignment="0">
      <SegmentTemplate timescale="48000" media="audio/fr/$Time$.mp4a">
        <SegmentTimeline>
          <S t="0" d="96000" r="432"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="a0" bandwidth="64000" />
    </AdaptationSet>
<!--AdapatationSet carrying Green Video Information for Video  -->
    <AdaptationSet id="green_video" codecs="dipi"/>
      <BaseURL>video_green_dipi/</BaseURL>
      <SegmentTemplate timescale="90000" media="$id$/$Time$.mp4m">
        <SegmentTimeline>
          <S t="0" d="180180" r="432"/>
        </SegmentTimeline>
      </SegmentTemplate>
      <Representation id="gv0" bandwidth="1000" associationId="v0 v1 v2"
       associationType="cdsc"/>
    </AdaptationSet>
  </Period>

</MPD>
```

B.3.3 Use of green metadata at the client

The client (player/decoder) can determine its remaining battery life based on the energy consumption of the current representation it is using. If it detects that its battery life is insufficient for the total duration of the video content to be consumed (given parameter in the server or requirements of duration expressed by the user), the terminal can compute the power consumption saving ratio from the current representation.

Using the following information, the terminal can determine (for the next Segment) the best power-saving allocation strategy for the decoder and for the display:

— the decoder-power saving ratio of all available video representations in the next Segmentfrom the current (selected) representation in the previous Segment,

— the impact of RGB component scaling on video quality for the next Segment,

— for the last Segments, the decoder and display consumption as a fraction of the total consumption.

From this information, the terminal determines which representation it needs to download and what is the appropriate scaling of RGB components for this representation.

It is observed that the decoder-power saving ratio of all available video representations from the current representation in the previous Segment is not directly given by the power-indication metadata. At the server, what is given is a list of two decoder operations reduction ratios per video representation:

— the first one is the ratio of each representation from the most energy-consuming one at a given period of time T (dash arrows in Figure B.7),

— the second one is the ratio of each representation at a given period of time T from the previous period of time $T - 1$ (continuous arrow in Figure B.8).

The terminal can convert this list of ratios into a list of ratios from the current representation it was using in the previous Segment.

Let us define the following terms:

— I_{refRep} the index, in the current Segment, of the representation which was used by the client terminal in the previous Segment,

— dec_ops_reduction_ratio_from_max(i) the reduction ratio from the most energy consuming representation, received from the server,

— $R_{\text{decOpsReducFromRepRef}}(\text{i})$ the reduction ratio from representation RefRep in the current Segment,

— $R_{\text{decOpsReducFromPrevRepRef}}(\text{i})$ the reduction ratio from representation RefRep in the previous Segment,

It is possible to express $R_{\text{decOpsReducFromRepRef}}(\text{i})$ from dec_ops_reduction_ratio_from_max(i), using the following formula:

$$R_{\text{decOpsReducFromRefRep(i)}} = \left[1 - \frac{100 - \text{dec\_ops\_reduction\_ratio\_from\_max(i)}}{100 - \text{dec\_ops\_reduction\_ratio\_from\_max}\left(I_{\text{refRep}}\right)}\right] * 100 \qquad \text{(B-5)}$$

$R_{\text{decOpsReducFromRepRef}}(\text{i})$ are represented by dotted arrows in Figure B.7. It is then possible to express $R_{\text{decOpsReducFromPrevRepRef}}(\text{i})$ from $R_{\text{decOpsReducFromRepRef}}(\text{i})$, using the following formula:

$$R_{\text{decOpsReducFromPrevRefRep(i)}} = 100 - \frac{\left(100 - \text{dec\_ops\_reduction\_ratio\_from\_prev}\left(I_{\text{refRep}}\right)\right) * \left(100 - R_{\text{decOpsReducFromRefRep}}(\text{i})\right)}{100} \qquad \text{(B-6)}$$

$R_{\text{decOpsReducFromPrevRepRef}}(\text{i})$ are represented by dash arrows in Figure B.8.

NOTE 1 Floating-point numbers are used for these computations.