C:\Users\shinji\_w\Desktop\ISO-IEC\_23008-1\_(E).docFINAL DRAFT INTERNATIONAL STANDARD© ISO/IEC 2015 – All rights reservedISO/IEC  23001-11 63Part 11: Green MetadataInformation technology — Green ViÉlément introductif — Élément central — Partie 1: Titre de la partieInformation technology — MPEG Systems Technologies  — Part 11: Energy-Efficient Media Consumption (Green Metadata)E2013-04-26(40) EnquiryISO/IECISO/IEC J   International Standard 2013ISO/IEC 23008‑ISO/IEC 23008‑1Information technology11291 2見出し 2見出し 1    02 STD Version 2.1c240   4            ISO/IEC JTC 1/SC 29

Date: 2023-xx-xx

ISO/IEC  23001-11

ISO/IEC JTC 1/SC 29/WG 11

Secretariat:

Information technology — MPEG Systems Technologies  — Part 11: Energy-Efficient Media Consumption (Green Metadata)

*Élément introductif — Élément central — Partie 1: Titre de la partie*

Contents Page

Introduction 6

1 Scope 1

2 Normative references 1

3 Terms, definitions, symbols, abbreviated terms 1

3.1 Terms and definitions 1

3.2 Symbols and abbreviated terms 3

4 Conventions 4

4.1 Arithmetic operators 4

4.2 Relational operators 4

4.3 Bit-wise operators 4

4.4 Assignment operators 5

4.5 Range notation 5

4.6 Mathematical functions 5

4.7 Specification of syntax functions and descriptors 5

5 Functional architecture 6

5.1 Description of the functional architecture 6

5.2 Definition of components in the functional architecture 7

6 Decoder power reduction 7

6.1 General 7

6.2 Complexity metrics for decoder-power reduction 8

6.3 Interactive signalling for remote decoder-power reduction 35

7 Display power reduction using display adaptation 37

7.1 General 37

7.2 Syntax 37

7.3 Signalling 39

7.4 Semantics 40

8 Energy-efficient media selection 48

8.1 General 48

8.2 Syntax 48

8.3 Signalling 48

8.4 Semantics 49

9 Metrics for quality recovery after low-power encoding 50

9.1 General 50

9.2 Syntax 50

9.3 Signalling 50

9.4 Semantics 51

10 Conformance and reference software 53

Annex A (normative) Supplemental Enhancement Information (SEI) syntax 54

A.1 Syntax and semantics of green metadata SEI message carried in AVC NAL units 54

A.1.1 Syntax 54

A.1.2 Semantics 56

A.2 Syntax and semantics of green metadata SEI message carried in HEVC NAL units 56

A.2.1 Syntax 56

A.2.2 Semantics 59

A.3 Syntax and semantics of green metadata SEI message carried in VVC NAL units 59

A.3.1 Syntax 59

A.3.2 Semantics 62

Annex B (informative) Implementation guidelines for the usage of green metadata 63

B.1 Codec dynamic voltage frequency scaling for decoder-power reduction 63

B.1.1 General 63

B.1.2 Derivation of the complexity metrics 63

B.1.2.1 Deriving the worst-case, largest value for *N*maxNumSixTapFiltPic(i) 63

B.1.2.2 Deriving the worst-case, largest value for *N*maxAlphaPointDbfsPic(i) 66

B.1.3 Example usage of C-DVFS metadata 68

B.2 Display adaptation 70

B.2.1 General 70

B.2.2 Example usage of display-adaptation metadata 70

B.2.2.1 Example usage of display-adaptation metadata for contrast enhancement 70

B.2.2.2 Preventing flicker arising from control latency 72

B.2.2.3 Metadata for DA on displays with control-frequency limitations 72

B.2.2.4 DA metadata to prevent flicker from large variations 73

B.3 Energy-efficient media selection in adaptive streaming 75

B.3.1 General 75

B.3.2 Green metadata production and transmission at the server side 75

B.3.3 Use of green metadata at the client 80

B.4 Interactive signalling for remote decoder-power reduction 83

B.4.1 General 83

B.4.2 Decoding operations reduction request computation and transmission 83

B.4.3 Use of decoding operations reduction request 83

B.5 Cross-segment decoding for quality recovery after low-power encoding 85

B.5.1 General 85

B.5.2 Green metadata Usage 85

Annex C (normative) Conformance and reference software 87

C.1 Complexity metrics for decoder-power reduction 87

C.1.1 Conformance test vectors 87

C.1.2 Reference software 88

C.2 Display-power reduction using display adaptation 88

C.2.1 Conformance test vectors 88

C.2.2 Reference software 88

C.3 Energy-efficient media selection 89

C.3.1 Conformance test vectors 89

C.3.2 Reference software 89

C.4 Metrics for quality recovery after low-power encoding 90

C.4.1 Conformance test vectors 90

C.4.2 Reference software 91

Annex D (informative) Objective distortion metrics 92

D.1 General 92

D.2 PSNR 92

D.3 wPSNR 92

D.4 WS-PSNR 94

D.4.1 WS-PSNR calculation for equi-rectangular projection (ERP) format with optional padding 94

D.4.2 WS-PSNR calculation for cubemap projection (CMP) and hemisphere cubemap projection (HCMP) formats 95

D.5 SSIM 95

Bibliography 97

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](https://www.iso.org/directives-and-policies.html) or [www.iec.ch/members\_experts/refdocs](https://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](https://www.iso.org/iso-standards-and-patents.html)) 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](https://www.iso.org/iso/foreword.html). In the IEC, see [www.iec.ch/understanding-standards](https://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:

— The clause 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.

— The 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.

— The clause 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.

— The clause 7 related to display power reduction using display adaptation is amended by extending Green SEI metadata. These metadata rely on the use of Attenuation Maps transmitted thanks to auxiliary pictures conveyed along with the base video pictures.

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](https://www.iso.org/members.html) and [www.iec.ch/national-committees](https://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 Attenuation Map Information (AMI) metadata, RGB-component statistics and quality levels. A presentation subsystem uses this metadata to reduce power by modifying the content based on Attenuation Map and/or 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)

# Scope

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

# 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*

# Terms, definitions, symbols, abbreviated terms

## 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](https://www.iso.org/obp/ui)

— 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 colour space

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

3.1.12

RGB component

single sample representing one of the three primary colours of the *RGB colour space* [3.1.11]

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

## Symbols and abbreviated terms

|  |  |
| --- | --- |
| + | 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 |

|  |  |
| --- | --- |
| AMI  APDI | Attenuation Map Information  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 |
| DA | display adaptation |
| 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 |

# Conventions

## Arithmetic operators

|  |  |
| --- | --- |
| *xy* | exponentiation |
| *x/y* | division where no truncation or rounding is intended |
|  | division where no truncation or rounding is intended |
|  | summation of  with *i* taking all integer values from *x* up to and including *y* |
|  | summation of 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 |

## 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.

## 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.

## 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)

## 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* |

## Mathematical functions

Mathematical functions are defined as follows:

 (4-1)

 (4-2)

Clip3( *x*, *y*, *z* ) = (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)

(4-7)

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

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

## 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.

# Functional architecture

## 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.

## 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.

# Decoder power reduction

## 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.

## Complexity metrics for decoder-power reduction

### 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.

### 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) |
| **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) |
| **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) |
| } |  |
| 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) |
| } |  |
| } |  |
| } |  |

### 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.

### Semantics

#### 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*picsInPeriod specifies the number of pictures in the specified period. When period\_type is 0 or 4, 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*mbsInPeriod specifies the total number of macroblocks that are coded in the specified period. It is determined by the following computation:

(6‑1)

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\_mapindicates 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 tth 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 tth 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 tth 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 tth 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:

(6‑2)

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:

(6‑3)

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:

(6‑4)

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 ) (6‑5)

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:

(6‑6)

*N*maxAlphaPtDbfsInPeriod is the maximum number of APDIs that can occur within the specified period and is derived from *N*mbsInPeriod and *S*chrMultvariables as

(6‑7)

*S*chrMultdepends on the AVC variables separate\_colour\_plane\_flag and chroma\_format\_idc as shown in the Table 5.

Table 5 – specification of *S*chrMultfor 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])

*N*mbsInSlice[0][j] = first\_mb\_in\_slice[0][j+1] – first\_mb\_in\_slice[0][j]

else

*N*mbsInSlice[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 14496‑10:2019, 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++;

*N*mbsInSlice[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:

(6‑10)

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:

(6‑11)

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:

(6‑12)

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 ] (6‑13)

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:

(6‑14)

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*chrMultvariables as

*N*maxAlphaPtDbfsInSlice[ i ][ j ] = 128 \* *S*chrMult\* *N*mbsInSlice[ i ][ j ] (6‑15)

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 lth 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 14496‑10:2019, 7.4.3).

priority\_id[ l ] indicates a priority identifier for the NAL unit in the lth scalable layer. The value of priority\_id[ l ] shall be in the range of 0 to 63, inclusive (as specified in ISO/IEC 14496‑10:2019, G.7.3.1.1).

dependency\_id[ l ] indicates a dependency identifier for the NAL unit in the lth scalable layer. The value of dependency\_id[ l ] shall be in the range of 0 to 7, inclusive (as specified in ISO/IEC 14496‑10:2019, G.7.3.1.1).

quality\_id[ l ] indicates a quality identifier for the NAL unit in the lth scalable layer. The value of quality\_id[ l ] shall be in the range of 0 to 15, inclusive (as specified in ISO/IEC 14496‑10:2019, G.7.3.1.1).

temporal\_id[ l ] indicates a temporal identifier for the NAL unit in the lth scalable layer. The value of temporal\_id[ l ] shall be in the range of 0 to 7, inclusive (as specified in ISO/IEC 14496‑10:2019, 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 lth scalable layer and is set equal to *P*nonZero8x8Blks[ i ][ j ] defined as follows:

(6‑16)

*N*mbsInLayerInPeriod[ l ] is the total number of macroblocks in the lth scalable layer in the specified period and is derived from *N*mbsInLayerInPeriod[ l ] and*N*picsInPeriodForTempLayer[ temporal\_id[ l ] ] as

(6‑17)

where *N*mbsInLayer[ l ] is the total number of macroblocks in the lth scalable layer and determined after derivation of the number of macroblocks associated with pic\_parameter\_set\_id[ l ], as specified in ISO/IEC 14496‑10:2019, 7.4.3.

*N*nonZero8x8Blks[ l ] is the number of 8x8 blocks with non-zero transform coefficients values in the lth 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 lth scalable layer and is set equal to *P*intraMbs[ i ][ j ] defined as follows:

(6‑18)

*N*intraMbs[ l ] is the number of macroblocks using intra prediction modes in the lth 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 lth scalable layer in the specified period and is set equal to *P*sixTapFilt[ i ][ j ] defined as follows:

(6‑19)

*N*maxSixTapFiltInLayerInPeriod[ l ] is the maximum number of STFs that can occur in the lth scalable layer in the specified period and is derived from *N*mbsInLayerInPeriod[ l ] variable as

*N*maxSixTapFiltInLayerInPeriod[ l ] = 1664 \* *N*mbsInLayerInPeriod[ l ] (6‑20)

*N*sixTapFilt[ l ] is the number of 6-tap filterings (STFs) within the lth 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 lth scalable layer in the specified period and is set equal to *P*alphaPtDbfs[ i ][ j ] defined as follows:

(6‑21)

*N*maxAlphaPtDbfsInLayerInPeriod[ l ] is the maximum number of APDIs that can occur in the lth scalable layer in the specified period and is derived from *N*mbsInLayerInPeriod[ l ] and *S*chrMultvariables as

*N*maxAlphaPtDbfsInLayerInPeriod[ l ] = 128 \* *S*chrMult\* *N*mbsInLayerInPeriod[ l ] (6‑22)

*N*alphaPtDbfs[ l ] is the number of alpha-point deblocking instances (APDIs) in the lth scalable layer in the specified period. It is derived from portion\_alpha\_point\_deblocking\_instances[ l ] and *N*maxAlphaPtDbfsInLayerInPeriod[ l ] in the decoder.

#### 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:

(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 (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:

(6‑25)

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:

(6‑26)

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:

(6‑27)

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:

(6‑28)

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:

(6‑29)

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:

(6‑30)

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:

(6‑31)

*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:

(6‑32)

where *N*intra8x8Blks, *N*intra16x16Blks, *N*intra32x32Blks and *N*intra64x64Blksare 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:

(6‑33)

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:

(6‑34)

where*N*planar4x4Blks, *N*planar8x8Blks, *N*planar16x16Blks, *N*planar32x32Blks and *N*planar64x64Blksare 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:

(6‑35)

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:

(6‑36)

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:

(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:

(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:

(6‑39)

When not present, it is set equal to 0.

*N*acdnFiltBlksis 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:

(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:

(6‑41)

When not present, it is set equal to 0.

*N*fikqFiltBlksis 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\_filteringsindicates 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:

(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:

(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\_instancesindicates the portion of deblocking filtering instances in the specified period and is set equal to *P*dbfInstances defined as follows:

(6‑44)

*S*chrMultdepends on the HEVC variables separate\_colour\_plane\_flag and chroma\_format\_idc as shown in the Table 7.

Table 7 – specification of *S*chrMultfor 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*dbfInstancesis the number of deblocking filtering instances in the specified period. It is derived from portion\_deblocking\_instances, *N*4x4BlksInPeriod and *S*chrMultin 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 23008‑2:2017, 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 (6‑45)

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:

(6‑46)

*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:

(6‑47)

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:

(6‑48)

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:

(6‑49)

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:

(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:

(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:

(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:

(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:

(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:

(6‑55)

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:

(6‑56)

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:

(6‑57)

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:

(6‑58)

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:

(6‑59)

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:

(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:

(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:

(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:

(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:

(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*dbfInstances[ t ] defined as follows:

(6‑65)

*N*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*4x4BlksInSliceOrTile[ t ] and *S*chrMultin the decoder.

#### 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\_typeindicates 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*picsInPeriod indicates 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*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:

(6‑66)

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 (6‑67)

where

— *N*ctbs is set equal to ( 1 << ( CtbLog2SizeY – 1 ) )2

— *S*picInCtb is set equal to PicSizeInCtbsY

— *S*chrMultdepends on the VVC variable sps\_chroma\_format\_idc as shown in the Table 10

Table 10 – specification of *S*chrMultfor 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:

(6‑68)

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:

(6‑69)

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*nonZero4\_8\_16\_Blks defined as follows:

(6‑70)

When not present, portion\_non\_zero\_4\_8\_16\_blocks\_area is set equal to 0.

*N*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:

(6‑71)

*N*nonZero4\_8\_16\_Blks is derived from portion\_non\_zero\_4\_8\_16\_blocks\_area and *N*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*nonZero32\_64\_128\_Blks defined as follows:

(6‑72)

When not present, portion\_non\_zero\_32\_64\_128\_blocks\_area is set equal to 0.

*N*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:

(6‑73)

*N*nonZero32\_64\_128\_Blks is derived from portion\_non\_zero\_32\_64\_128\_blocks\_area and *N*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*nonZero256\_512\_1024\_Blks defined as follows:

(6‑74)

When not present, portion\_non\_zero\_256\_512\_1024\_blocks\_area is set equal to 0.

*N*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:

(6‑75)

*N*nonZero256\_512\_1024\_Blks is derived from portion\_non\_zero\_256\_512\_1024\_blocks\_area and *N*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*nonZero2018\_4096\_Blks defined as follows:

(6‑76)

When not present, portion\_non\_zero\_2048\_4096\_blocks\_area is set equal to 0.

*N*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:

(6‑77)

*N*nonZero2048\_4096\_Blks is derived from portion\_non\_zero\_2048\_4096\_blocks\_area and *N*nonZeroBlks in the decoder.

portion\_non\_zero\_transform\_coefficients\_areaindicates 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*nonZeroCoefsArea defined as follows:

(6‑78)

*N*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*nonZeroTransformCoefs is derived from portion\_non\_zero\_transform\_coefficients\_area and *N*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*intraPredBlks defined as follows:

(6‑79)

*N*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:

(6‑80)

where *N*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*intraPredBlks is derived from portion\_intra\_predicted\_blocks\_area and *N*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*biGpmPredBlks defined as follows:

(6‑81)

*N*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:

(6‑82)

Where *N*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*biAndGpmPredBlks is derived from portion\_bi\_and\_gpm\_predicted\_blocks\_area and *N*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:

(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:

(6‑84)

Where *N*bdofBlks\_*X* are the number of inter-coded blocks using BDOF, for blocks with number of samples *X*=256, 512, 1024, 2048, 4096, in the specified period.

*N*bdofBlks is derived from portion\_bdof\_blocks\_area and *N*4SampleBlksInPeriod 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:

(6‑85)

*N*dbfInstancesis the number of deblocking filtering instances in the specified period. It is derived from portion\_deblocking\_instances, and *N*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:

(6‑86)

*N*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*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:

(6‑87)

*N*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*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 tth segment. When granularity\_type is equal to 1, segment\_address[ t ] indicates the subpicture ID of the tth 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*4SampleBlksInSegment[ t ] is the total number of 4-samples luma and chroma blocks in the slice[ t ] or tile[ t ] or subpicture[ t ]. *N*4SampleBlksInSegment[ t ] is determined by the following computation.

— If granularity\_type is equal to 1, *N*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*4SampleBlksInSegment[ t ] is set equal to ( *S*chrMult\* *W*subpic \* *H*subpic ).
* if granularity\_type is equal to 2, *N*4SampleBlksInSegment[ t ] is derived as follows from VVC variables NumCtusInSlice and CtbLog2SizeY specified in ISO/IEC 23090-3:
* *N*4SampleBlksInSegment[ t ] is set equal to *S*chrMult\* ( ( NumCtusInSlice[ t ] ) << ( CtbLog2SizeY – 1) )
* Otherwise, if granularity\_type is equal to 3, *N*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*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*nonZeroBlksArea[ t ] defined as follows:

(6‑88)

where *N*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:

(6‑89)

where *N*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*nonZeroBlksInSegment[ t ] is derived from portion\_non\_zero\_blocks\_area[ t ] and *N*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*nonZero4\_8\_16\_Blks[ t ] defined as follows:

(6‑90)

When not present, portion\_non\_zero\_4\_8\_16\_blocks\_area[ t ] is set equal to 0.

*N*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:

(6‑91)

*N*nonZero4\_8\_16\_BlksInSegment[ t ] is derived from portion\_non\_zero\_4\_8\_16\_blocks\_area[ t ] and *N*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*nonZero32\_64\_128\_Blks[ t ] defined as follows:

(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:

(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:

(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:

(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:

(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:

(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:

(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*intraPredBlks[ t ] defined as follows:

(6‑99)

*N*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:

(6‑100)

Where *N*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*intraPredBlks[ t ] is derived from portion\_intra\_predicted\_blocks\_area[ t ] and *N*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*biGpmPredBlks[ t ] defined as follows:

(6‑101)

*N*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:

(6‑102)

Where *N*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*biAndGpmPredBlks[ t ] is derived from portion\_bi\_and\_gpm\_predicted\_blocks\_area[ t ] and *N*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*bdofBlks[ t ] defined as follows:

(6‑103)

*N*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:

(6‑104)

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

*N*bdofBlks[ t ] is derived from portion\_bdof\_blocks\_area[ t ] and *N*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:

(6‑105)

*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:

(6‑106)

*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:

(6‑107)

*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.

## Interactive signalling for remote decoder-power reduction

### 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.

### 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) |
| **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) |
| } |  |

### 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.

### 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*DecOpsReductionReq is derived by dec\_ops\_reduction\_req and indicates the requested percentage change of local decoding operations by

(6‑108)

where *d*req is set equal to dec\_ops\_reduction\_req.

A negative percentage means a decrease of decoding operations. *P*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.

# Display power reduction using display adaptation

## General

With respect to the functional architecture, display adaptation (DA) provides green metadata comprised of Attenuation Map Information (AMI), RGB-component statistics and quality indicators. The Attenuation Map Information (AMI) metadata indicate how to use Attenuation Maps carried as auxiliary pictures of type AUX\_ALPHA for display adaptation. The Attenuation Maps and their related Attenuation Map Information as well as RGB-component statistics are used to set display controls in the presentation subsystem so that desired quality levels and corresponding display power reductions are attained.

## Syntax

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

#### Systems using SEI messages to transmit DA green metadata

The syntax for the AMI metadata is described in Table 13. This syntax is the same for AVC, HEVC and VVC.

Table 13– syntax for the AMI metadata

|  |  |  |
| --- | --- | --- |
|  | | **Descriptor** |
| **ami\_cancel\_flag** | u(1) | |
| if ( !**ami\_cancel\_flag** ) { |  | |
| **ami\_display\_model** | u(4) | |
| **ami\_global\_flag** | u(1) | |
| **ami\_map\_approximation\_model** | u(4) | |
| **ami\_map\_number** | u(3) | |
| for ( i=0;i<ami\_map\_number;i++ ) { |  | |
| **ami\_layer\_id**[ i ] | u(8) | |
| **ami\_ols\_number**[ i ] | u(4) | |
| for ( j=0;j<ami\_ols\_number[i];j++){ |  | |
| **ami\_ols\_id**[ i ][ j ] | u(8) | |
| } |  | |
| **ami\_energy\_reduction\_rate**[ i ] | u(5) | |
| **ami\_max\_value**[ i ] | u(8) | |
| if ( !**ami\_global\_flag** ) or ( i == 0 ) { |  | |
| **ami\_attenuation\_use\_idc**[ i ] | u(4) | |
| **ami\_attenuation\_comp\_idc**[ i ] | u(4) | |
| **ami\_preprocessing\_flag**[ i ] | u(1) | |
| if( **ami\_preprocessing\_flag**[ i ] ){ |  | |
| **ami\_preprocessing\_type\_idc**[ i ] | u(2) | |
| } |  | |
| **ami\_preprocessing\_scale\_idc** [ i ] | u(8) | |
| **ami\_backlight\_scaling\_idc**[ i ] | u(4) | |
| } |  | |
| } |  | |
| } |  | |

#### Systems not using SEI message to transmit DA green metadata

The message format used to send metadata from the transmitter to the receiver is described in Table 14.

Table 14 – 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) |
| } |  |
| } |  |
| } |  |

### 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 15 – 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 16 – 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) |
| } |  |

## Signalling

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

#### Systems using SEI messages to transmit DA green metadata

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.

Attenuation Map Information (AMI) metadata describing how to use Attenuation Maps are carried through SEI message from the transmitter to the receiver. The Attenuation Maps are carried as auxiliary pictures of type AUX\_ALPHA for display adaptation, with the flag alpha\_channel\_use\_idc equal to 3.

The complete syntax of the green metadata SEI message payload, including the Attenuation Map Information, is specified in Annex A.

The SEI message containing the AMI metadata is transmitted at the start of an upcoming period. The next message containing AMI metadata 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.

#### Systems not using SEI messages to transmit DA green metadata

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 7.2.1.2, 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.

### 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.

## Semantics

#### Semantics using SEI messages to transmit DA green metadata

The semantics of various terms are defined below.

Note: In a preferred mode, the Attenuation Map shall be applied on the sample values of the decoded primary picture(s), i.e., just after the decoding process.

The Attenuation Map Information (AMI) metadata provide information about the interpretation of the Attenuation Map sample values coded in auxiliary pictures of type AUX\_ALPHA (in the following picAMI) and the post-processing intended to be applied to the one or more associated primary pictures of the CVS.

Note: the association of auxiliary pictures to primary pictures are specified in the SDI SEI message (see ISO/IEC 23002-7).

When a CVS does not contain an SDI SEI message with sdi\_aux\_id[ i ] equal to 1 for at least one value of i, no picture in the CVS shall be associated with a Green metadata SEI message.

When an access unit (AU) contains both an SDI SEI message with sdi\_aux\_id[ i ] equal to 1 for at least one value of i and a Green metadata SEI message, the SDI SEI message shall precede the Green metadata SEI message in decoding order.

When an AU contains a picAMI in a layer, with nuh\_layer\_id equal to nuhLayerIdAMI, that is indicated as an Alpha Map auxiliary layer by an SDI SEI message, the Attenuation Map sample values of picAMI persist in output order until one or more of the following conditions are true:

– The next picture, in output order, with nuh\_layer\_id equal to nuhLayerIdAMI is output.

– A CLVS containing the auxiliary picture picAMI ends.

– The bitstream ends.

– A CLVS of any associated primary layer of the auxiliary picture layer with nuh\_layer\_id equal to nuhLayerIdAMI ends.

The following semantics apply separately to each nuh\_layer\_id targetLayerId among the nuh\_layer\_id values to which the Green metadata SEI message applies.

ami\_cancel\_flag equal to 1 indicates that the SEI message cancels the persistence of any previous Attenuation Map Information SEI message in output order. ami\_cancel\_flag equal to 0 indicates that Attenuation Map Information parameters follow.

ami\_map\_number specifies the number of auxiliary pictures of type AUX\_ALPHA in the CVS.

ami\_display\_model is a bit field mask which indicates the display models on which the Attenuation Map sample values of the auxiliary picture picAMI may be used.

Table 17 - Interpretation of the bits of ami\_display\_model

|  |  |
| --- | --- |
| Bit number | Display model |
| 0 | Backlit pixel |
| 1 | Emissive pixel |
| 2..3 | Reserved for future types |

For example, ami\_display\_model=11 means the Attenuation Map Information can be used for both “Backlit” and “Emissive” display models.

ami\_map\_approximation\_model specifies the model used to extrapolate a set of received Attenuation Map sample values(s) from a set of decoded auxiliary picture(s) with individual energy reduction rate(s) to another set of Attenuation Map sample values with a different energy reduction rate.

ami\_map\_approximation\_model equal to 0 specifies that a linear scaling of the Attenuation Map sample values of the provided auxiliary picture given its respective ami\_energy\_reduction\_rate should be considered to obtain corresponding Attenuation Map sample values for another energy reduction rate. In case several auxiliary pictures picAMI are provided, the auxiliary picture with the lowest ami\_energy\_reduction\_rate is used for the linear scaling. This is the preferred type.

ami\_map\_approximation\_model equal to 1 specifies that a bilinear interpolation between the Attenuation Map sample values of the provided auxiliary picture(s) given their respective ami\_energy\_reduction\_rate should be considered to obtain corresponding Attenuation Map sample values for another energy reduction rate.

ami\_map\_approximation\_model equal to 2 specifies that an interpolation of type Lanczos between the Attenuation Map sample values of the provided auxiliary picture(s) given their respective ami\_energy\_reduction\_rate should be considered to obtain corresponding Attenuation Map sample values for another energy reduction rate.

ami\_map\_approximation\_model equal to 3 specifies that an interpolation of type bicubic between the Attenuation Map sample values of the provided auxiliary picture(s) given their respective ami\_energy\_reduction\_rate should be considered to obtain corresponding Attenuation Map sample values for another energy reduction rate.

ami\_map\_approximation\_model equal to 4 specifies that a proprietary user defined process should be used to infer corresponding Attenuation Map sample values for another energy reduction rate from the Attenuation Map sample values of the provided auxiliary picture(s) given their respective ami\_energy\_reduction\_rate.

Table 18 - Interpretation of ami\_map\_approximation\_model

|  |  |
| --- | --- |
| ami\_map\_approximation\_model | Attenuation Map interpolation process |
| 0 | Linear scaling |
| 1 | Bilinear interpolation |
| 2 | Lanczos interpolation |
| 3 | Bicubic interpolation |
| 4 | User defined |
| 5..15 | Reserved for future uses |

ami\_global\_flag equal to 0 indicates that ami\_attenuation\_use\_idc[ i ], ami\_attenuation\_comp\_idc[ i ], ami\_preprocessing\_flag[ i ], ami\_preprocessing\_type\_idc[ i ], ami\_preprocessing\_scale\_idc[ i ], ami\_backlight\_scaling\_idc[ i ] for i=0.. ami\_map\_number, shall be present. ami\_global\_flag equal to 1 indicates that only ami\_attenuation\_use\_idc[ 0 ], ami\_attenuation\_comp\_idc[ 0 ], ami\_preprocessing\_flag[ 0 ], ami\_preprocessing\_type\_idc[ 0 ], ami\_preprocessing\_scale\_idc[ 0 ], ami\_backlight\_scaling\_idc[ 0 ] shall be present.

ami\_attenuation\_use\_idc[ i ] specifies the use of the Attenuation Map sample values of the decoded auxiliary picture of index i.

ami\_attenuation\_use\_idc[ i ] equal to 0 specifies that the Attenuation Map sample values of the decoded auxiliary picture should be subtracted from one or more associated primary picture decoded sample(s) before displayed on screen. This is the preferred type.

ami\_attenuation\_use\_idc[ i ] equal to 1 specifies that the Attenuation Map sample values of the decoded auxiliary picture should be multiplied by one or more associated primary picture decoded sample(s) before displayed on screen.

ami\_attenuation\_use\_idc[ i ] equal to 2 specifies that the Attenuation Map sample values of the decoded auxiliary picture should be used according to a proprietary user defined process to modify the one or more associated primary picture decoded sample(s) before displayed on screen.

Table 19 - Interpretation of ami\_attenuation\_use\_idc[ i ]

|  |  |
| --- | --- |
| ami\_attenuation\_use\_idc[ i ] | Process to apply on associated primary picture decoded samples |
| 0 | Subtraction |
| 1 | Multiplication |
| 2 | User defined |
| 3..15 | Reserved for future uses |

ami\_attenuation\_comp\_idc[ i ] specifies on which colour component(s) of the associated primary picture(s) decoded samples the decoded auxiliary picture of type AUX\_ALPHA of index i should be applied using the process defined by ami\_attenuation\_use\_idc[ i ].

ami\_attenuation\_comp\_idc[ i ] equal to 0 specifies that the luma component of the decoded auxiliary picture of type AUX\_ALPHA of index i should be applied to the luma component of the associated primary picture(s) decoded samples. This is the preferred type.

ami\_attenuation\_comp\_idc[ i ] equal to 1 specifies that the luma component of the decoded auxiliary picture of type AUX\_ALPHA of index i should be applied to the luma component and the chroma components of the associated primary picture(s) decoded samples.

ami\_attenuation\_comp\_idc[ i ] equal to 2 specifies that the luma component of the decoded auxiliary picture of type AUX\_ALPHA of index i should be applied to the RGB components (after YUV to RGB conversion) of the associated primary picture(s) decoded samples.

ami\_attenuation\_comp\_idc[ i ] equal to 3 specifies that the luma component of the decoded auxiliary picture of type AUX\_ALPHA of index i should be applied to the first component of the associated primary picture(s) decoded samples.

ami\_attenuation\_comp\_idc[ i ] equal to 4 specifies that the luma component of the decoded auxiliary picture of type AUX\_ALPHA of index i should be applied to the second component of the associated primary picture(s) decoded samples.

ami\_attenuation\_comp\_idc[ i ] equal to 5 specifies that the luma component of the decoded auxiliary picture of type AUX\_ALPHA of index i should be applied to the third component of the associated primary picture(s) decoded samples.

ami\_attenuation\_comp\_idc[ i ] equal to 6 specifies that the mapping between the luma component of the decoded auxiliary picture of type AUX\_ALPHA of index i and the components of which to apply the decoded auxiliary picture of type AUX\_ALPHA of index i corresponds to some proprietary user-defined process.

Table 20 - Interpretation of ami\_attenuation\_comp\_idc[ i ]

|  |  |
| --- | --- |
| ami\_attenuation\_comp\_idc[ i ] | Mapping between components of the Attenuation Map and primary picture components on which to apply the Attenuation Map |
| 0 | Luma component in the picAMI applied to the luma component of the associated primary picture |
| 1 | Luma component in the picAMI applied to the luma and chroma components of the associated primary picture |
| 2 | Luma component in the picAMI applied to the three RGB components of the associated primary picture |
| 3 | Luma component in the picAMI applied to the first component of the associated primary picture |
| 4 | Luma component in the picAMI applied to the second component of the associated primary picture |
| 5 | Luma component in the picAMI applied to the third component of the associated primary picture |
| 6 | User defined |
| 7..15 | Reserved for future uses |

ami\_backlight\_scaling\_idc[ i ] specifies the process to compute the scaling factor of the backlight of transmissive pixel displays, derived from the Attenuation Map sample values of the decoded auxiliary picture of index i.

ami\_backlight\_scaling\_idc[ i ] equal to 0 specifies that the scaling to apply to the backlight of the display is computed as the ratio between the maximal values of the associated primary picture decoded samples after and before applying the Attenuation Map sample values of the decoded auxiliary picture of index i. The associated primary picture decoded sample(s) on which the Attenuation Map sample values of the decoded auxiliary picture of index i are applied are further rescaled to their maximal value before the application of the Attenuation Map sample values of the decoded auxiliary picture of index i. This is the preferred type.

ami\_backlight\_scaling\_idc[ i ] equal to 1 specifies that the scaling to apply to the backlight of the display is determined according to a proprietary user defined process derived from the Attenuation Map sample values of the decoded auxiliary picture of index i.

Table 21 - Interpretation of ami\_backlight\_scaling\_idc[ i ]

|  |  |
| --- | --- |
| ami\_backlight\_scaling\_idc[ i ] | Backlight scaling processing type |
| 0 | Scaling by the ratio between maximum values before and after the use of the Attenuation Map |
| 1 | User defined |
| 2..15 | Reserved for future uses |

ami\_preprocessing\_flag[ i ] specifies whether some pre-upsampling is to be used on the Attenuation Map sample values of the decoded auxiliary picture of index i. In that case it is supposed that the auxiliary coded picture(s) and the primary coded picture have different sizes.

ami\_preprocessing\_type\_idc[ i ] specifies the recommended type of the interpolation (e.g., bicubic) used to resample the Attenuation Map sample values of the decoded auxiliary picture of index i at the same resolution as the associated decoded picture.

ami\_preprocessing\_type\_idc[ i ] equal to 0 specifies that an interpolation of type bicubic between the Attenuation Map sample values of the provided auxiliary picture of index i should be considered to obtain the Attenuation Map sample values to apply to the sample values of the decoded picture. This is the preferred type.

ami\_preprocessing\_type\_idc[ i ] equal to 1 specifies that a bilinear interpolation between the Attenuation Map sample values of the provided auxiliary picture of index i should be considered to obtain the Attenuation Map sample values to apply to the sample values of the decoded picture.

ami\_preprocessing\_type\_idc[ i ] equal to 2 specifies that an interpolation of type Lanczos between the Attenuation Map sample values of the provided auxiliary picture of index i should be considered to obtain the Attenuation Map sample values to apply to the sample values of the decoded picture.

ami\_preprocessing\_type\_idc[ i ] equal to 3 specifies that a proprietary user defined process should be used to pre-upsample the Attenuation Map sample values of the provided auxiliary picture of index i to obtain the Attenuation Map sample values to apply to the sample values of the decoded picture.

Table 22- Interpretation of ami\_preprocessing\_type\_idc[ i ]

|  |  |
| --- | --- |
| ami\_preprocessing\_type\_idc[ i ] | Attenuation Map preprocessing type |
| 0 | Bicubic interpolation |
| 1 | Bilinear interpolation |
| 2 | Lanczos interpolation |
| 3 | User defined |
| 4..15 | Reserved for future uses |

ami\_preprocessing\_scale\_idc[ i ] specifies which scaling should be applied to the Attenuation Map of index i to obtain the Attenuation Map sample values before applying them on the sample values of the decoded picture.

ami\_preprocessing\_scale\_idc[ i ] equal 0 specifies that a scaling of should be applied to the Attenuation Map of index i to obtain the Attenuation Map sample values before applying them on the sample values of the decoded picture. This is the preferred type.

ami\_preprocessing\_scale\_idc[ i ] equal 1 specifies that a proprietary user defined scaling should be applied to the Attenuation Map of index i to obtain the Attenuation Map sample values before applying them on the sample values of the decoded picture.

Table 23 - Interpretation of ami\_preprocessing\_scale\_idc[ i ]

|  |  |
| --- | --- |
| ami\_preprocessing\_scale\_idc[ i ] | Attenuation Map scaling preprocess |
| 0 | Scaling of |
| 1 | User defined |
| 2..15 | Reserved for future uses |

ami\_layer\_id[ i ] specifies the identifier of the decoded layer for the Attenuation Map of index i.

ami\_ols\_number[ i ] specifies the number of Output Layer Sets to which the Attenuation Map of index i belongs.

ami\_ols\_id[ i ][ j ] specifies the identifier of the Output Layer Set of index j for the Attenuation Map of index i. This identifier shall be used to select the OLS to output both the primary decoded picture and the Attenuation Map of index i.

ami\_energy\_reduction\_rate[ i ] indicates the expected energy saving rate when the video is displayed after applying the Attenuation Map sample values of the decoded auxiliary picture of index i on the sample values of the decoded picture.

ami\_max\_value[ i ] indicates the maximum value of the attenuation map of index i. Such a maximal value can be optionally used to further adjust the dynamic of the encoded attenuation map in the scaling process.

#### Semantics not using SEI message for DA green metadata

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*BacklightScalingFactor[ k ][ j ][ i ] for that frame. *V*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 kth constant\_backlight\_voltage\_time\_interval, jth max\_variation and ith 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 jth 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 kth 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 kth constant\_backlight\_voltage\_time\_interval and jth max\_variation. If lower\_bound[ k ][ j ] = 0, then contrast-enhancement metadata is unavailable.

upper\_bound[ k ][ j ] indicates for the kth constant\_backlight\_voltage\_time\_interval and jth 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 kth constant\_backlight\_voltage\_time\_interval and jth 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 kth constant\_backlight\_voltage\_time\_interval, jth max\_variation and ith 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 kth constant\_backlight\_voltage\_time\_interval, jth max\_variation and ith 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 *vPSNR*, defined as follows for 0 < i <= num\_quality\_levels:

(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.
* *Xs* 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.

# Energy-efficient media selection

## 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 23009‑1:2019, 3.1.39);

— 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.

## Syntax

The decoder-power indication metadata is a pair of decoder operations reduction ratios. The syntax is described in Table 24 .

Table 24 – 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 25 – 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) |
| } |  |

## 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 23009‑3 [1] and illustrated in Annex B.

## Semantics

### 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*opsReducRatioFromMax(i) derived as follows:

(8-1)

*N*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 23009‑1:2019, 3.1.34). A negative value means an increase in decoding operations. dec\_ops\_reduction\_ratio\_from\_prev(i) is set equal to *d*opsReducRatioFromPrev(i) derived as follows:

(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*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*PrevDecOps(i) = *N*DecOps(i).

### 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.

# Metrics for quality recovery after low-power encoding

## 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.

## Syntax

### 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 26 - .

Table 26 - – syntax for quality metrics for AVC and HEVC

|  |  |
| --- | --- |
|  | **Descriptor** |
| **xsd\_metric\_type** | u(8) |
| **xsd\_metric\_value** | u(16) |

### VVC syntax

For VVC bitstreams, the encoder embeds the metadata message in the associated picture using the syntax of Table 28 .

Table 28 – 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) |

## 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.

## Semantics

### AVC and HEVC Semantics

xsd\_metric\_type indicates the type of the objective quality metric as shown in the Table 30 . PSNR, as defined in ISO/IEC 23001‑10, is the only type currently supported. A definition of PSNR is provided in Annex D.

Table 30 – 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:

(9.1)

### 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 ith subpicture.

xsd\_metric\_number\_minus1[ i ] plus 1 indicates the number of objective quality metrics associated with the ith subpicture.

xsd\_metric\_type[ i ][ j ] indicates the type of the jth objective quality metric associated with the ith subpicture as shown in Table 32 . 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 32 – 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 jth objective quality metric associated with the ith subpicture. When xsd\_subpic\_number\_minus1 is equal to 0, xsd\_metric\_value[ 0 ][ j ] contains the value of jth 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 ]:

(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 ]:

(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 ]:

(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*WS-PSNR value (in dB) as follows, with *m* set equal to xsd\_metric\_value[ i ][ j ]:

(9.5)

# Conformance and reference software

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

1. (normative)  
     
   Supplemental Enhancement Information (SEI) syntax
   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.

* + 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; |  |
| case 2: |  |
| **ami\_cancel\_flag** | u(1) |
| if ( !**ami\_cancel\_flag** ) { |  |
| **ami\_display\_model** | u(4) |
| **ami\_global\_flag** | u(1) |
| **ami\_map\_approximation\_model** | u(4) |
| **ami\_map\_number** | u(3) |
| for ( i=0;i<ami\_map\_number;i++ ) { |  |
| **ami\_layer\_id**[ i ] | u(8) |
| **ami\_ols\_number**[ i ] | u(4) |
| for ( j=0;j<ami\_ols\_number[i];j++){ |  |
| **ami\_ols\_id**[ i ][ j ] | u(8) |
| } |  |
| **ami\_energy\_reduction\_rate**[ i ] | u(5) |
| **ami\_max\_value**[ i ] | u(8) |
| if ( !**ami\_global\_flag** ) or ( i ==0 ) { |  |
| **ami\_attenuation\_use\_idc**[ i ] | u(4) |
| **ami\_attenuation\_comp\_idc**[ i ] | u(4) |
| **ami\_preprocessing\_flag**[ i ] | u(1) |
| if( **ami\_preprocessing\_flag**[ i ] ){ |  |
| **ami\_preprocessing\_type\_idc**[ i ] | u(2) |
| } |  |
| **ami\_preprocessing\_scale\_idc**[ i ] | u(8) |
| **ami\_backlight\_scaling\_idc**[ i ] | u(4) |
| } |  |
| } |  |
| } |  |
| break; |  |
| default: |  |
| } |  |
| } |  |

* + 1. 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. If green\_metadata\_type is 1, then metadata enabling quality recovery after low-power encoding is present. If green\_metadata\_type is 2, then metadata enabling the use of Attenuation Maps for Display Adaptation is present. Other values of green\_metadata\_type are reserved for future use by ISO/IEC.

* 1. 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.

* + 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; |  |
| case 2: |  |
| **ami\_cancel\_flag** | u(1) |
| if ( !**ami\_cancel\_flag** ) { |  |
| **ami\_display\_model** | u(4) |
| **ami\_global\_flag** | u(1) |
| **ami\_map\_approximation\_model** | u(4) |
| **ami\_map\_number** | u(3) |
| for ( i=0;i<ami\_map\_number;i++ ) { |  |
| **ami\_layer\_id**[ i ] | u(8) |
| **ami\_ols\_number**[ i ] | u(4) |
| for ( j=0;j<ami\_ols\_number[i];j++){ |  |
| **ami\_ols\_id**[ i ][ j ] | u(8) |
| } |  |
| **ami\_energy\_reduction\_rate**[ i ] | u(5) |
| **ami\_max\_value**[ i ] | u(8) |
| if ( !**ami\_global\_flag** ) or ( i ==0 ) { |  |
| **ami\_attenuation\_use\_idc**[ i ] | u(4) |
| **ami\_attenuation\_comp\_idc**[ i ] | u(4) |
| **ami\_preprocessing\_flag**[ i ] | u(1) |
| if( **ami\_preprocessing\_flag**[ i ] ){ |  |
| **ami\_preprocessing\_type\_idc**[ i ] | u(2) |
| } |  |
| **ami\_preprocessing\_scale\_idc**[ i ] | u(8) |
| **ami\_backlight\_scaling\_idc**[ i ] | u(4) |
| } |  |
| } |  |
| } |  |
| break; |  |
| default: |  |
| } |  |
| } |  |

* + 1. 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. If green\_metadata\_type is 1, then metadata enabling quality recovery after low-power encoding is present. If green\_metadata\_type is 2, then metadata enabling the use of Attenuation Maps for Display Adaptation is present. Other values of green\_metadata\_type are reserved for future use by ISO/IEC.

* 1. 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.

* + 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) |
| **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\_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; |  |
| case 2: |  |
| **ami\_cancel\_flag** | u(1) |
| if ( !**ami\_cancel\_flag** ) { |  |
| **ami\_display\_model** | u(4) |
| **ami\_global\_flag** | u(1) |
| **ami\_map\_approximation\_model** | u(4) |
| **ami\_map\_number** | u(3) |
| for ( i=0;i<ami\_map\_number;i++ ) { |  |
| **ami\_layer\_id**[ i ] | u(8) |
| **ami\_ols\_number**[ i ] | u(4) |
| for ( j=0;j<ami\_ols\_number[i];j++){ |  |
| **ami\_ols\_id**[ i ][ j ] | u(8) |
| } |  |
| **ami\_energy\_reduction\_rate**[ i ] | u(5) |
| **ami\_max\_value**[ i ] | u(8) |
| if ( !**ami\_global\_flag** ) or ( i ==0 ) { |  |
| **ami\_attenuation\_use\_idc**[ i ] | u(4) |
| **ami\_attenuation\_comp\_idc**[ i ] | u(4) |
| **ami\_preprocessing\_flag**[ i ] | u(1) |
| if( **ami\_preprocessing\_flag**[ i ] ){ |  |
| **ami\_preprocessing\_type\_idc**[ i ] | u(2) |
| } |  |
| **ami\_preprocessing\_scale\_idc**[ i ] | u(8) |
| **ami\_backlight\_scaling\_idc**[ i ] | u(4) |
| } |  |
| } |  |
| } |  |
| break; |  |
| default: |  |
| } |  |
| } |  |

* + 1. 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. If green\_metadata\_type is 1, then metadata enabling quality recovery after low-power encoding is present. If green\_metadata\_type is 2, then metadata enabling the use of Attenuation Maps of Display Adaptation is present. Other values of green\_metadata\_type are reserved for future use by ISO/IEC.

1. (informative)  
     
   Implementation guidelines for the usage of green metadata
   1. Codec dynamic voltage frequency scaling for decoder-power reduction
      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.

* + 1. Derivation of the complexity metrics

5.2.2 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*maxNumSixTapFiltPic(i) and *N*maxAlphaPointDbfsPic(i) are derived.

5.2.2 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.

5.2.2 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.

* + - 1. Deriving the worst-case, largest value for *N*maxNumSixTapFiltPic(i)

To determine *N*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: aG, bG, cG, dG, eG, fG, gG, hG, iG, jG, kG, nG, pG, qG, rG. If the MV points to aG, then the decoder shall compute aG and the 15 points (aH, aI, ...) that have the same respective relative locations to H, I, J, M, N, P, Q, R, S, V, W, T, U, X, Y that aG has to G. Similarly, the decoder shall compute 16 points for each of the other fractional-sample positions (bG, cG, ..., rG) 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

1. If the MV points to bG, then to interpolate bG, 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 bG, ..., bY for the 4x4 block.
2. If the MV points to hG, then to interpolate hG, 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 hG, ..., hY for the 4x4 block.
3. If the MV points to jG, then to interpolate jG, the decoder shall apply 6 STFs to compute aa, bb, bG, sM, gg, hh because these are unavailable. Next, 1 STF is needed to compute jG from aa, bb, bG, sM, gg, hh. So 7 STFs are required for jG.
4. To get jM, the decoder needs bb, bG, sM, gg, hh, ii. Only ii is unavailable. So 2 STFs are needed for jM (one for ii and one for jM).
5. To get jR, the decoder needs 2 STFs (one for jj and one for jR).
6. To get jT, the decoder needs 2 STFs (one for kk and one for jT).
7. Therefore, for jG, jM, jR and jT, 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 jG, ... jY for the 4x4 block.
8. If the MV points to aG, then to interpolate aG, the decoder needs 1 STF to get bG (from (a)) and therefore 16 STFs to compute aG, ..., aY for the 4x4 block.
9. If the MV points to cG, then to interpolate cG, the decoder needs 1 STF to get bG (from (a)) and therefore 16 STFs to compute cG, ..., cY for the 4x4 block.
10. If the MV points to dG, then to interpolate dG, the decoder needs 1 STF to get hG (from (b)) and therefore 16 STFs to compute dG, ..., dY for the 4x4 block.
11. If the MV points to nG, then to interpolate nG, the decoder needs 1 STF to get hG (from (b)) and therefore 16 STFs to compute nG, ..., nY for the 4x4 block.
12. If the MV points to fG, then to interpolate fG, the decoder needs 7 STFs to get jG (from (c)). Note that bG is included in these 7 STFs. Therefore, from (c), 52 STFs are required to compute fG, ... fY for the 4x4 block.
13. If the MV points to iG, then to interpolate iG, the decoder needs 7 STFs to get jG. Note that hG is computed by one of these 7 STFs. Therefore, 52 STFs are required to compute iG, ... iY for the 4x4 block. For this analysis, the row jG, jH, jI, jJ is computed first (to obtain hG) 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).
14. If the MV points to kG, then to interpolate kG, the decoder needs 7 STFs to get jG. Note that mG is computed by one of these 7 STFs. Therefore, 52 STFs are required to compute kG, ... kY for the 4x4 block.
15. If the MV points to qG, then to interpolate qG, the decoder needs 7 STFs to get jG. Note that sG is computed by one of these 7 STFs. Therefore, 52 STFs are required to compute qG, ... qY for the 4x4 block.
16. If the MV points to eG, then to interpolate eG, the decoder needs 2 STFs to get bG and hG (from (a), (b)). Therefore 32 STFs are needed to compute eG, ..., eY for the 4x4 block.
17. If the MV points to gG, then to interpolate gG, the decoder needs 2 STFs to get bG and mH. Therefore, 32 STFs are needed to compute gG, ..., gY for the 4x4 block.
18. If the MV points to pG, then to interpolate pG, the decoder needs 2 STFs to get hG and sG. Therefore, 32 STFs are needed to compute pG, ..., pY for the 4x4 block.
19. If the MV points to rG, then to interpolate rG, the decoder needs 2 STFs to get mG and sG. Therefore, 32 STFs are needed to compute rG, ..., rY for the 4x4 block.

From (a),…,(n), the worst-case, largest number of STFs is 52, when the MV points to jG, fG, iG, kG or qG. 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)

* + - 1. Deriving the worst-case, largest value for *N*maxAlphaPointDbfsPic(i)

To determine *N*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 pi and qi 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 q0 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 (p0, q0) 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 (p0, q0) 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.



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

1. For each chroma block in 4:2:0 format, two vertical edges and two horizontal edges are filtered. Each edge contributes 8 APDIs, in the worst-case. So, 8\*4\*2 = 64 APDIs are produced by worst-case deblocking of the two chroma blocks.
2. For 4:2:2 format, two vertical edges and four horizontal edges are filtered. Each vertical edge contributes 16 APDIs and each horizontal edge contributes 8 APDIs. So, 2\*(2\*16 + 4\*8) = 128 APDIs are produced by worst-case deblocking of the two chroma blocks.
3. For 4:4:4 format, the worst-case analysis for each chroma block is identical to that of the 16x16 luma block. Therefore, 256 APDIs are produced by worst-case deblocking of the two chroma blocks.
4. Finally, for separate colour planes, the worst-case analysis of a 16x16 block is identical to that 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:

*N*maxAlphaPointDbfsPic(i)= PicSizeInMbs \* (128 + 64) = 192 \* PicSizeInMbs, for 4:2:0,

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

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

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

* + 1. 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:

(B-3)

where *C*pict is the total picture complexity. The total number of macroblocks per picture (*nMB*) and the number of bits per picture (*nbit*) 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 kα 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 \* nMB, *k*bit \* *n*bit, *k*nz \* *n*nz, *k*intra \* *n*intra, *k*six \* *n*six, and *k*α \* *n*α.



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

* 1. Display adaptation
     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.

* + 1. Example usage of display-adaptation metadata

The metadata scaled\_psnr\_rgb[ i ] indicates the PSNR for the ith 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.

* + - 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.

* + - 1. 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.

* + - 1. 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.

* + - 1. 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*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*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*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.

* 1. Energy-efficient media selection in adaptive streaming
     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.

* + 1. 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>

* + 1. 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*decOpsReducFromRepRef(i) the reduction ratio from representation RefRep in the current Segment,

— *R*decOpsReducFromPrevRepRef(i) the reduction ratio from representation RefRep in the previous Segment,

It is possible to express *R*decOpsReducFromRepRef(i) from dec\_ops\_reduction\_ratio\_from\_max(i), using the following formula:

(B-5)

*R*decOpsReducFromRepRef(i) are represented by dotted arrows in Figure B.7. It is then possible to express *R*decOpsReducFromPrevRepRef(i) from *R*decOpsReducFromRepRef(i), using the following formula:

(B-6)

*R*decOpsReducFromPrevRepRef(i) are represented by dash arrows in Figure B.8.

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



Figure B.7 — Derivation of DecOpsReductionRatios within the current Segment



Figure B.8 — Derivation of DecOpsReductionRatios within the current Segmentfrom the previous Segment

Using the mapping between Processing frequency of processors or devices and Power Supply Voltage and the mapping between Power Supply Voltage and Power consumption, the terminal can translate this list into a list of decoder-power saving ratios from the representation which was used in the previous Segment.

In the case where the total duration of the video content to be consumed is not known (case of live content for example), the terminal can display the expected remaining usage duration based on current battery level and the energy consumption of the current representation it is using. The user can therefore act on its terminal to increase this usage duration, which are translated into a power saving ratio as in the previous case.

NOTE 2 Complexity metrics, as defined in 6.2 can be sent with each representation to allow the client to save energy by proactively invoking C-DVFS to make the selection of representation work at its best for energy saving.

NOTE 3 The dec\_ops\_reduction\_ratio is known to be stable across software-based platforms.

* 1. Interactive signalling for remote decoder-power reduction
     1. General

This clause explains how interactive green metadata can be computed at the decoder and how it can be used at the encoder.

* + 1. Decoding operations reduction request computation and transmission

A terminal can display the expected remaining usage duration based on current battery level and the energy consumption of the current video it is decoding. The user can then act on its terminal to increase this usage duration, which are translated into a power-consumption saving ratio.

This power-consumption saving ratio cannot be sent as is to the remote device because the relationship between power consumption and processing cycles of a processor is not linear and is processor/device dependent.

Using the mapping between power consumption and power supply voltage and the mapping between power supply voltage and processing frequency of processors or devices, the terminal can translate the power-consumption saving ratio into decoding operations reduction request (DOR-Req) metadata (step 1 in Figure B.9).

The DOR-Req metadata are sent as green feedback by the device in an out-of-band message (step 2 in Figure B.9).



Figure B.9 — Production of the DOR-Req message

* + 1. Use of decoding operations reduction request

The decoding operations reduction request (DOR-Req) metadata are extracted in the remote device and presented to the power optimization module (step 3 in Figure B.10) which translates this request into a configuration of the encoder (step 4 in Figure B.10), so that it can produce a stream which complies to the DOR-Req (step 5 in Figure B.10).



Figure B.10 — Usage of the DOR-Req message

Thus, each encoder can adapt the complexity of the encoded stream as a function of the battery level of the other device communicating with it.

The strategy used by the encoder to reduce complexity is non-normative. A gradual action can be used to find the best compromise: decoding complexity vs. perceived quality. For example, when the first DOR-Req mode is used (corresponding to dec\_pow\_reduction\_type equal to 0), the gradual action is controlled through the monitoring of the DOR-Req parameterdec\_ops\_reduction\_req. When the DOR-Req parameter dec\_ops\_reduction\_req becomes negative, the encoder knows that it can gradually increase the complexity to reverse its previous actions.

If the two devices are equipped with batteries, the best strategy can be defined by considering the power-saving requests of both devices as shown in Figure B.11.



Figure B.11 — Using local and remote information in the power-optimizer module

This technology achieves maximum power saving in association with C-DVFS technology. As shown in Figure B.10, the DOR-Req messages are directed at remote encoders. However, if these remote encoders produce C-DVFS SEI messages in association with the video stream, then the local decoders can use the C-DVFS SEI message for the maximum power saving achieved from the change initiated by the DOR-Req message.

* 1. Cross-segment decoding for quality recovery after low-power encoding
     1. General

An encoder can achieve power reduction by encoding alternate high-quality and low-quality Segments, in a segmented delivery mechanism such as DASH. The power reduction occurs because low-complexity encoding mechanisms (fewer encoding modes, fewer reference pictures, smaller search ranges, etc.) are used to produce the low-quality Segments. A metric describing the quality of the last picture of each Segment is delivered as metadata to the decoder. This clause describes how cross-segment decoding can be used to improve the quality of the low-quality Segments.

A cross-segment decoder utilizes quality metrics contained in the high-quality Segments (from high complexity encoding) to enhance decoding of the low-quality Segments (from low complexity encoding), producing a visual experience with significantly higher QoE, but with reduced average encoding complexity (and therefore reduced encoding power consumption).

Note that the decoding complexity for the first picture in the low-quality Segment is increased, while the decoding complexity for the other pictures remains the same as for regular decoders.

* + 1. Green metadata Usage

At the transmitter, the encoder records the quality metric of the last picture of each Segment using xsd\_metric\_type and xsd\_metric\_value. The XSD-enabled decoder, when it receives the metric data, uses the metrics to determine if it executes an enhancement algorithm. If the metric indicates that the last picture of the previous Segment is of better quality than the first picture of the new Segment, then it uses the last picture of the previous Segment to enhance the first picture of the new Segment as described below.

The XSD algorithm applies to the transition from a Segment with higher video quality to a temporally neighbouring Segment with poorer quality that is encoded independently of the higher quality Segment. The last picture (in display order) in the higher quality Segment is the “good picture” (GP). The first IDR picture of the poor quality Segment is the “start picture” (SP). The output from the current algorithm is the “fresh start” (FS). Note that the SP as an IDR picture was encoded without referencing the GP or any other pictures in the higher quality Segment. The goal of the enhancement algorithm is to use information contained in the GP to improve the quality of the decoded SP to get an improved reference picture, FS, for subsequent pictures in the low quality Segment.

Depending on the level of motion for different spatial regions of the SP, two enhancement methods are used by the decoder, one for relatively low-motion areas, the other for the higher-motion areas. For both algorithms, the decoder looks for matches between areas in the decoded GP and the SP, as determined by a distortion metric and a threshold calculated by the decoder.

NOTE MSD = Mean Square Difference in the following algorithm description.

// To estimate MVs, different algorithms can be used. Square Diamond Search is described below.

**for each** block B in SP

set the centre to B.

calculate the SAD between B and the co-located block B' in GP.

**repeat**  
 calculate the SAD between B and the block in the up left, up right,

down left, down right of B' in GP.

select the block leading to the minimum SAD as the next centre.

set B' as last centre

**until** the centre = last centre;

**repeat**  
 calculate the SAD between B and the block in the left, right, up, down of B' in GP.

select the block leading to the minimum SAD as the next centre.

set B' as last centre

**until** the centre = last centre

Set C' to the centre in GP.

Set MV(B) to the motion vector from C' to B

// Enhance the new SP

calculate the average Sum of Absolute Difference (AvgSAD) of the MVs.

calculate TMSD = 0.775 \* e0.4306 \* AvgSAD + 132.4.

**for** each 16 x16 patch P in SP

// QP is average quantization parameter in frame.

// width is the width of the frame.

// len(MV(P)) is the number of bits to code MV(P).

**if** (len(MV(P)) < width \* QP /30000) // Use low-motion enhancement method

calculate the MSD between P and the co-located patch P' in GP

**if** (MSD < = TMSD)

copy P' to P;

**else** // P is high motion – use high-motion enhancement method

**for** each 4x4 block B in P

good\_mv = 0;

**for** each of 8 mv near MV(B)

**if** mv = MV(B)

good\_mv++;  
 **if** (good\_mv >= 5)

calculate the MSD between B and co-located B' in GP referenced by MV(B)

**if** (MSD <= TMSD)

copy B' to B;

After the FS-Picture Generation algorithm is applied, decoding of subsequent pictures is done as usual.

1. (normative)  
     
   Conformance and reference software
   1. Complexity metrics for decoder-power reduction
      1. Conformance test vectors

Table C.1 describes the two 4:2:0 8 bit per sample AVC conformance bitstreams with embedded green metadata SEI message available at <http://standards.iso.org/iso-iec/23001/-11/ed-3/en>.

Table C.1 – AVC conformance bitstreams with complexity metrics

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **name** | **resolution /Frame rate** (Fps) | **RAP period**  (number of frames) | **bitrate**  (Mbps) | **green metadata SEI period**  (number of frames) | **profile** | **expected value** |
| mobcal\_480p50\_AVC\_HP.bin | 704x480p@50 | 15 | 2.798 | 1 | High | mobcal\_480p50\_AVC\_HP.txt |
| stockholm\_720p5994\_AVC\_HP.bin | 1280x720p@59.94 | 15 | 7.205 | 1 | High | stockholm\_720p5994\_AVC\_HP.txt |

To verify conformance of a software implementation of green metadata SEI message parsing, the conformance streams shall be used to check that extracted values match expected values given in the side text files provided with the conformance streams.

Table C.2 describes four 4:2:0 10 bit per sample VVC conformance bitstreams with embedded green metadata SEI message available at <http://standards.iso.org/iso-iec/23001/-11/ed-3/en>.

Table C.2 – VVC conformance bitstreams with complexity metrics

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **name** | **resolution /Frame rate** (Fps) | **RAP period**  (number of frames) | **bitrate**  (Mbps) | **green metadata SEI period**  (number of frames) | **profile** | **expected value** |
| blowingBubbles\_testCMs\_ext.vvc | 416x240@50 | 50 | 1.561 | 1 | Main10 | blowingBubbles\_testCMs\_ext.txt |
| blowingBubbles\_testCMs\_reduced.vvc | 416x240@50 | 50 | 1.561 | 1 | Main10 | blowingBubbles\_testCMs\_reduced.txt |
| cactus\_testCMs\_ext.txt | 1920x1080p@50 | 32 | 1.074 | 1 | Main10 | cactus\_testCMs\_ext.vvc |
| cactus\_testCMs\_reduced.txt | 1920x1080p@50 | 32 | 1.074 | 1 | Main10 | cactus\_testCMs\_reduced.vvc |

To verify conformance of a software implementation of VVC green metadata SEI message parsing, the conformance streams shall be used to check that extracted values match expected values given in the side text files provided with the conformance streams.

* + 1. Reference software

Reference decoder software provided in ISO/IEC 14496-5 or Rec. ITU-T H.264.2 integrates a green metadata SEI message parser for AVC bitstreams, which extracts and displays SEI messages from conformance and test bitstreams.

To enable the green metadata SEI message parser, the source code shall be compiled with the macro #define PRINT\_GREEN\_METADATA\_INFO.

To verify conformance of a test green metadata SEI message generated from a video in a test AVC bitstream, the reference decoder software provided in ISO/IEC 14496-5 or Rec. ITU-T H.264.2 shall be used to extract the test SEI message from the test bitstream and then to check the message for syntactic correctness and valid ranges.

Reference decoder software provided in ISO/IEC 23090-16 integrates a green metadata SEI message parser for VVC bitstreams, which extracts and displays SEI messages from conformance and test bitstreams.

To verify conformance of a test green metadata SEI message generated from a video in a test VVC bitstream, the reference decoder software provided in ISO/IEC 23090-16 shall be used to extract the test SEI message from the test bitstream and then to check the message for syntactic correctness and valid ranges.

* 1. Display-power reduction using display adaptation
     1. Conformance test vectors

One conformance ISO BMFF file, BasketballDrill\_28\_gamma.mp4m, which contains green metadata samples of ‘dfce’ sample entry type, as specified in ISO/IEC 23001-10, is available at <http://standards.iso.org/iso-iec/23001/-11/ed-3/en>.

It is composed of a sample entry which contains static metadata and samples which contain dynamic metadata.

To verify conformance of a software implementation of ‘dfce’ green metadata samples parsing in an ISO BMFF file, the conformance file shall be used to check that extracted values match expected values given in the side text file provided with the conformance file.

* + 1. Reference software

A reference software for parsing and display of ‘dfce’ green metadata samples in ISO BMFF file is available at <http://standards.iso.org/iso-iec/23001/-11/ed-3/en>.

It is linked with ISO BMFF reference software libraries (IsoLib), which are available in ISO/IEC 14496-5.

A readme.txt is provided to explain how to produce the executable in a Windows or Linux environment.

The reference software takes the ISO BMFF metadata file (\*.mp4m) as input and produces a text file as output, which gives a full description of the metadata stored in the samples of the input file.

To verify conformance of test metadata files, the reference software shall be used to parse the test metadata files and to check them for syntactic correctness and valid ranges.

* 1. Energy-efficient media selection
     1. Conformance test vectors

A conformance test vector for decoder-power indication metadata is available at <http://standards.iso.org/iso-iec/23001/-11/ed-3/en>.

It consists of a set of:

* ten ISO BMFF video files, which provide ten AVC video representations, with (sub)segments duration of 2 s, at the following resolutions and bitrates:
  + 1920x1080p50 @ 10Mbps,
  + 1920x1080p50 @ 8Mbps,
  + 1600x900p50 @ 8Mbps,
  + 1600x900p50 @ 6Mbps,
  + 1280x720p50 @ 6Mbps,
  + 1280x720p50 @ 5Mbps,
  + 960x540p50 @ 5Mbps,
  + 960x540p50 @ 3.5Mbps,
  + 768x432p50 @ 3.5Mbps,
  + 768x432p25 @ 2.5Mbps.
* ten ISO BMFF metadata files, which provide associated decoder-power indication (‘depi’) metadata representation of each video representation,
* a manifest file, conformant to ISO/IEC 23009-1.

The ISO BMFF metadata files contain green metadata samples of ‘depi’ sample entry type, as specified in ISO/IEC 23001-10.

To verify conformance of a software implementation of ‘depi’ green metadata samples parsing in an ISO BMFF file, the conformance metadata files shall be used to check that extracted values match expected values given in the side text files provided with the conformance files.

* + 1. Reference software

A reference software for parsing and display of decoder-power (‘depi’) or display-power (‘dipi’) indication metadata in ISO BMFF file is available at <http://standards.iso.org/iso-iec/23001/-11/ed-3/en>.

It is linked with ISO BMFF reference software libraries (IsoLib), which are available in ISO/IEC 14496-5.

A readme.txt is provided to explain how to produce the executable in Windows or Linux environment.

The reference software takes the ISO BMFF metadata file (\*.mp4m) as input and produces a text file as output, which gives a full description of the metadata stored in the samples of the input file.

To verify conformance of test metadata files, the reference software shall be used to parse the test metadata files and to check them for syntactic correctness and valid ranges.

* 1. Metrics for quality recovery after low-power encoding
     1. Conformance test vectors

Table C.3 describes 4:2:0 8 bits AVC conformance bitstream with embedded green metadata SEI message available at <http://standards.iso.org/iso-iec/23001/-11/ed-3/en>.

Table C.3 – AVC conformance bitstreams with quality metrics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **name** | **resolution /**  **frame rate** (Fps) | **bitrate** (Mbps) | **profile** | **expected value** |
| crowdrun\_1080p50\_AVC\_HP.bin | 1920x1080p@50 | 10 | High | 28.70 |

The bitstream embeds for the last frame a SEI message that contains PSNR value of this frame, as described in this document.

To verify conformance of a software implementation of green metadata SEI message parsing, the conformance AVC stream shall be used to check that the extracted PSNR value matches the expected value given in the side text file provided with the conformance stream.

Table C.4 describes 4:2:0 8 bit per sample HEVC conformance bitstream with embedded green metadata SEI message available at <http://standards.iso.org/iso-iec/23001/-11/ed-3/en>.

Table C.4 – HEVC conformance bitstreams with quality metrics

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **name** | **resolution /**  **frame rate** (Fps) | **bitrate** (Mbps) | **profile** | **expected value** |
| kimono\_1080p24\_HEVC\_MP.bin | 1920x1080p@24 | 1 | Main | 37.99 |

The bitstream embeds for the last frame a SEI message that contains the PSNR value of this frame as described in this document.

To verify conformance of a software implementation of green metadata SEI message parsing, the conformance HEVC stream shall be parsed to check that the PSNR extracted value matches the expected value given in the side text file provided with the conformance stream.

Table C.5 describes 4:2:0 10 bit per sample VVC conformance bitstreams with embedded green metadata SEI message available at <http://standards.iso.org/iso-iec/23001/-11/ed-3/en>.

Table C.5 – VVC conformance bitstreams with quality metrics

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **name** | **resolution /**  **frame rate** (Fps) | **bitrate** (Mbps) | **profile** | **expected value (PSNR)** | **expected Value (SSIM)** |
| blowingBubbles\_testQMs.vvc | 416x240@50 | 6.264 | Main10 | frame 0: 42.60 frame 1: 39.61 | - |
| cactus\_testQMs.vvc | 1920x1080p@50 | 9.271 | Main10 | frame 0: 35.03 frame 1: 34.09 | frame 0: 0.97 frame 1: 0.97 |

The bitstreams embed for frames 0 and 1 a SEI message that contains the PSNR (blowingBubbles\_testQMs.vvc) and the PSNR and the SSIM (cactus\_testQMs.vvc) of two frames as described in this document.

To verify conformance of a software implementation of green metadata SEI message parsing, the conformance VVC streams shall be parsed to check that the PSNR and, when relevant, SSIM extracted values match the expected values given in the side text files provided with the conformance streams.

* + 1. Reference software

Reference decoder software provided in ISO/IEC 14496-5 or Rec. ITU-T H.264.2 integrates a green metadata SEI message parser, which extracts and displays SEI messages from conformance and test AVC bitstreams.

To enable the green metadata SEI message parser, the source code shall be compiled with the macro #define PRINT\_GREEN\_METADATA\_INFO.

To verify conformance of a test green metadata SEI message generated from a video in a test AVC bitstream, the reference software provided in ISO/IEC 14496-5 or Rec. ITU-T H.264.2 shall be used to extract the test SEI message from the test bitstream and then to check the message for syntactic correctness and valid ranges.

Reference decoder software provided in ISO/IEC 23008-5 integrates a green metadata SEI message parser, which extracts and displays SEI messages from conformance and test HEVC bitstreams.

To verify conformance of a test HEVC green metadata SEI message generated from a video in a test HEVC bitstream, the reference software provided in ISO/IEC 23008-5 shall be used to extract the test SEI message from the test bitstream and then to check the message for syntactic correctness and valid ranges.

Reference decoder software provided in ISO/IEC 23090-16 integrates a green metadata SEI message parser, which extracts and displays SEI messages from conformance and test VVC bitstreams.

To verify conformance of a test green metadata SEI message generated from a video in a test VVC bitstream, the reference software provided in ISO/IEC 23090-16 shall be used to extract the test SEI message from the test bitstream and then to check the message for syntactic correctness and valid ranges.

1. (informative)  
     
   Objective distortion metrics
   1. General

This annex provides a description of the processes for computing objective distortion metrics between samples of a reference picture and samples of a test picture, the reference and test pictures being made of 1 or 3 colour planes (e.g. YCbCr or RGB).

* 1. PSNR

The PSNR metric is calculated from the mean squared error of the sample values. PSNR is calculated individually for a single colour channel. A definition of PSNR is provided in ISO/IEC 23001‑10.

The inputs of PSNR metric estimation process are:

— the reference samples, with *c* being the index of the colour plane for which the metric is computed and and *p* being the samples relative location in the picture.

— the test samples, with *c* being the index of the colour plane for which the metric is computed and *p* being the samples relative location in the picture.

— the bit depth *B* of the samples of the colour channel c.

The output of this process is:

— the PSNR metric *V*PSNR.

A description of the metric derivation process follows.

The PSNR metric *V*PSNR is calculated as:

where *X* is the maximum sample value for the specific bit depth *B* derived as ( ( 1 << *B* ) – 1 ) and *V*MSE is given as:

where *Nc* is the number of samples in the considered colour plane of index *c*.

NOTE: In this document, the PSNR metric is derived for channel *c* = 0.

* 1. wPSNR

The wPSNR metric, or weighted PSNR metric, is calculated from a weighted mean squared error of the sample values, that can be used as distortion metric of Rec. BT.2100 PQ video content. wPSNR is calculated individually for a single luma (*c* = 0) or chroma (*c* = 1 or 2) channel. A definition of wPSNR is provided in document ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29 JVET-V2011.

The inputs of wPSNR metric estimation process are:

— the reference luma samples, with *p* being the samples relative location in the picture.

— the reference samples, with *c* being the index of the colour plane for which the metric is computed and and *p* being the samples relative location in the picture.

— the test samples, with *c* being the index of the colour plane for which the metric is computed and *p* being the samples relative location in the picture.

— the bit depth *B* of the samples of the colour channel *c*.

— the bit depth *B*0 of the samples of the luma colour channel *c* = 0.

The output of this process is:

— the wPSNR metric *v*wPSNR.

A description of the metric derivation process follows.

The wPSNR metric *V*wPSNR is calculated as:

where *X* is the maximum sample value for the specific bit depth *B* derived as ( ( 1 << *B* ) – 1 ) and *w*MSE is given as

where *N*c is the number of samples in the considered colour plane of index *c*, is a weight that is a function of the luma sample value at relative location *p* in the picture.

The weight is computed as follows:

— the value *X*L is derived as follows.

— if *B*0 is greater than or equal to 10, *X*L is set equal to .

— otherwise (*B*0 is lower than 10), *X*L is set equal to .

— the value *Y*P is set equal to Clip3(–3, 6, 0.015 \* *X*L – 1.5 – 6).

— is set equal to power(2.0, *Y*P ÷ 3.0).

NOTE: In this document, the wPSNR metric is derived for channel *c* = 0.

* 1. WS-PSNR

The WS-PSNR metric, or weighted to spherically uniform PSNR metric, is calculated from a weighted mean squared error of the sample values, that can be used as distortion metric of 360° video content. WS-PSNR calculates PSNR using all image samples on the 2D projection plane. The distortion at each position *p* is weighted by the spherical area covered by that sample position. WS-PSNR is calculated individually for a single luma (*c* = 0) or chroma (*c* = 1 or 2) channel. A definition of WS-PSNR is provided in document ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29 JVET-T2004.

The inputs of this process are:

— the reference samples, with *c* being the index of the colour plane for which the metric is computed and *p* being the samples relative location in the picture.

— the test samples, with *c* being the index of the colour plane for which the metric is computed and *p* being the samples relative location in the picture.

— the bit depth *B* of the samples of the colour channel *c*.

The output of this process is:

— the WS-PSNR metric *V*WS-PSNR.

A description of the metric derivation process follows.

The WS-PSNR metric *V*WS-PSNR is calculated as:

where *X* is the maximum sample value for the specific bit depth *B* derived as ( ( 1 << *B* ) – 1 ) and   
*V*WS-MSE is given as

where *N*c is the number of samples in the considered colour plane of index *c*, and is a weight that is a function of the relative location *p* in the picture.

Since WS-PSNR is calculated based on the projection plane, different weights are derived for different projection formats. Weight derivation for different projection formats is discussed below.

NOTE: In this document, the WS-PSNR metric is derived for channel *c* = 0.

* + 1. WS-PSNR calculation for equi-rectangular projection (ERP) format with optional padding

For an image in the ERP format, the weight  at relative position *p=( i , j )* in the picture is calculated as:

* + 1. WS-PSNR calculation for cubemap projection (CMP) and hemisphere cubemap projection (HCMP) formats

For an image in CMP format, weight distributions on all faces are the same. Therefore, only weights in one face are derived. For relative position *p*= on a CMP face with resolution , the weight is calculated as:

where is the radius, and is the squared distance between and the center of the face.

The weights for HCMP are based on those used for CMP, but only consider the active full face and four half-faces.

* 1. SSIM

SSIM is calculated individually for a single colour channel.

The inputs of SSIM metric estimation process are:

— the reference samples, with *c* being the index of the colour plane for which the metric is computed and and *p* being the samples relative location in the picture.

— the test samples, with *c* being the index of the colour plane for which the metric is computed and *p* being the samples relative location in the picture.

— the bit depth *B* of the samples of the colour channel *c*.

— the width *W* and height *H* of the colour channel *c*.

The output of this process is:

— the SSIM metric *V*SSIM.

A description of the metric derivation process follows.

The SSIM metric *V*SSIM is calculated as:

where *S=8* is the block width and height used to compute *V*SSIM, and *V*SSIM is computed as follows for the relative location *p*= in the picture as follows:

where *c1* = (*k1 \* X*)2, *c2* = (*k2 \* X*)2, *k1* = 0.01, *k2* = 0.03, *X* = ( ( 1 << *B* ) – 1 )*,*

and where , , , and are defined as follows*:*

— denotes the average value of reference samples , for *r* in the 8x8 window with top left pixel positioned at relative location *p*= in the picture,

— denotes the average value of test samples , for *r* in the 8x8 window with top left pixel positioned at relative location *p*= in the picture,

— denotes the variance value of reference samples , for *r* in the 8x8 window with top left pixel positioned at relative location *p*= in the picture,

— denotes the variance value of test samples , for *r* in the 8x8 window with top left pixel positioned at relative location *p*= in the picture,

— denotes the covariance value of reference samples and test samples , for *r* in the 8x8 window with top left pixel positioned at relative location *p*= in the picture.

is derived as follows, for *p*= and for sig=ref or test:

— the parameter is set equal to 0

— for *k*=0..(*S* – 1) and *l*=0..(*S* – 1) the following applies:

=

— is set equal to

is derived as follows, for *p*= and for *sig*=*ref* or *test*:

— the parameter is set equal to 0

— for *k*=0..(*S* – 1) and *l*=0..(*S* – 1) the following applies:

— is set equal to

is derived as follows, for *p*=:

— the parameter is set equal to 0

— for *k*=0..(*S* – 1) and *l*=0..(*S* – 1) the following applies:

=

—  is set equal to

NOTE: In this document, the SSIM metric is derived for channel *c* = 0.

Bibliography

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

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

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

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

[3] ISO/IEC 23008-5, Information technology — High efficiency coding and media delivery in heterogeneous environments — Part 5: Reference software for high efficiency video coding

[4] ISO/IEC 14496-5, Information technology — Coding of audio-visual objects — Part 5: Reference software | Rec. ITU-T H.264.2: Reference software for ITU-T H.264 advanced video coding

[5] ISO/IEC 23090-16, Information technology — Coded representation of immersive media — Part 16: Reference software for versatile video coding