|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  15th Meeting: Geneva, CH, 23 Oct. – 1 Nov. 2013 | Document: JCTVC-O1008\_v3 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **High efficiency video coding (HEVC) scalable extension Draft 4** | | |
| *Status:* | Output Document of JCT-VC | | |
| *Purpose:* | Draft of SHVC | | |
| *Author(s) or Contact(s):* | Jianle Chen, Qualcomm  Jill Boyce, Vidyo  Yan Ye, InterDigital  Miska M. Hannuksela, Nokia  Ye-kui Wang, Qualcomm | Email: | [cjianle@qti.qualcomm.com](mailto:cjianle@qti.qualcomm.com)  [jill@vidyo.com](mailto:jill@vidyo.com)  [Yan.Ye@interdigital.com](mailto:Yan.Ye@interdigital.com)  [miska.hannuksela@nokia.com](mailto:miska.hannuksela@nokia.com)  [yekuiw@qti.qualcomm.com](mailto:yekuiw@qti.qualcomm.com) |
| *Source:* | Editors | | |

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# Abstract

This document contains a preliminary version of SHVC Draft 4 text.

In this document, Annex F contains common syntax, semantics and decoding processes for multi-layer video coding extensions and Annex H contains Syntax, semantics and decoding processes for the scalable extensions. Modifications in long sections copied from the HEVC spec are highlighted in turquoise. Modifications to the MV-HEVC spec Annex C, D, E, F are highlighted in green. Open issues and editor's notes are highlighted in yellow.

Ed. Notes (Draft 4) (changes to JCTVC-N1008)

* --------- Release v3 -----------
* Accepted all change marks.
* ----------- Release v2 -----------
* (Fix Ticket #4)
* (Fix Ticket #3)
* (Review JB01): editorial improvements
* (Review JC02): editorial improvement and fix
* (Common HLS from JCT3V-F1004\_v6) Ported from JCT3V-F1004\_v6 and trace marks in Annex F is not recorded in this document. See JCT3V-F1004 for the integration detail of each proposal
* ----------- Release v1 -----------
* (Review MH02)
* (Review MH01):
  + editorial improvements and fixes, editor's notes
  + Made sure that for all profile constraints the following decision is obeyed: (JCTVC-O0253, Scalable Main profile decision 1): profile constraints apply to an output layer set
* (Review YY01): editorial improvements and fixes, especially related to Scalable Main profile constraints.
* (Review JB01): editorial improvement and fix
* (Review JC01): editorial improvement and fix
* (Scalable main profile decision 3): base layer bitstream shall be conformant to main profile and enhancement layers shall be YUV420 and 8 bits
* (JCTVC-O0094, Scalable Main profile decision 2): layer number in any dependency layer chain shall be less than or equal to 8
* (JCTVC-O0253, Scalable Main profile decision 1): profile constraints apply to an output layer set
* (JCTVC-O0216): Slice information derivation for inter-layer reference picture
* (JCTVC-O0215): Signaling a flag to specify the phase alignment between layers (zero or center phase shift) for upsampling process
* (JCTVC-O0199): Adding a flag in VPS VUI for indication of skipping enhancement layer IRAP picture when single\_layer\_for\_non\_irap\_flag is equal to 1
* (JCTVC-O0194): Supporting bit-depth scalability by reducing scaling step after resampling when higher bit depth is used in enhancement layer
* (SCE1): Arbitrary Spatial Ratio (ASR) with filters as documented in JCTVC-O0031 tables 2 and 3, first column

Ed. Notes (Draft 3) (changes to JCTVC-N0242)

* ----------- Release v2 -----------
* (Review JB01): editorial improvements
* (Review JC02): editorial improvement and fix
* (Review MH01): editorial fixes in the inter-layer constrained tile sets SEI message
* (Consisent ILRPS derivation with MV-HEVC text)
* (JCTVC-N0160): offset delay calculation for extended spatial scalability
* (Fix chroma filter coefficient at phase 11)
* (Review YY01): editorial improvement and fix
* ----------- Release v1 -----------
* (Subclause cross-reference clean up)
* (Review JB01)
* (Review JC01)
* (Require texture and MV prediction from same layer): Prohibit the case that, the inter-layer texture prediction is from one reference layer and the inter-layer motion prediction is from another reference layer for decoding one enhancement layer picture
* (JCTVC-N0108): Improve text clarity by adding explicit constraint that sample resampling may be done once per enhancement layer picture, and motion field resampling may be done once per enhancement layer picture
* (Note to disable TMVP when only inter-layer pred ) Add an editorial note for SHVC encoders to avoid use of TMVP when only the inter-layer reference pictures exist in the reference picture lists
* (Common HLS of multi-layer video coding extensions): Ported from JCT3V-E1004\_v5 and trace mark is not recorded in this document. See JCT3V-E1004\_v5 for the integration detail of each proposal
* (Motion mapping text completion): Picture and slice level information derivation for resampled interlayer reference picture
* (JCTVC-N0214):Intermediate data dynamic range control for the cases of 10-bits or higher input
* (JCTVC-N0139): Adding a rounding offset for the colocated position derivation in reference layer motion derivation
* (JCTVC-N0111): Using scaling factor to calculate the rounding offset for reference layer sample location derivation

Ed. Notes (JCTVC-N0242) (changes to JCTVC-M1008)

* ----------- Editorial improvement of Wording Draft 2 (submmited as to JCTVC-N0242)
* (Restructured Annexes) Annex F contain common parts of MV-HEVC and SHVC, Annex H contain SHVC specific text

Ed. Notes (Wording Draft 2) (changes to JCTVC-L1008)

* ----------- Release v3 -----------
* (Review JL03) Review, clean ups.
* (Review YY02) Review, Editorial improvement.
* (Common HLS03)Common high level syntax ported from MV-HEVC text JCT3V-D1004\_v3,
  + A group of high level syntax proposals and editorial improvement are ported with this track, please see JCT3V-D1004\_v3 for the integration detail of each proposal
* ----------- Release v2 -----------
* Modifications to JCTVC-M0309: scaled reference layer picture offsets
* (JCTVC-M0040): Using SHVC for adaptive resolution change
* (Review JL02) Review and editorial improvement for interlayer MV scaling,
* (Common HLS02)Common high level syntax ported from MV-HEVC text JCT3V-D1004\_v2,
  + A group of high level syntax proposals and editorial improvement are ported with this track, please see JCT3V-D1004\_v2 for the integration detail of each proposal
* ----------- Release v1 -----------
* (SHVC only adoption): Integrate annex G of SHVC test model text (JCTVC-L1007) with updates of the following SHVC only adoptions at 13th meeting
  + (Review YY01) Review and editorial improvement
  + (Review JB01) Review and editorial improvement
  + (Review JL01) Review and editorial improvement
  + (JCTVC-M0269): limit inter-layer prediction for a particular picture to use at most one inter-layer reference picture for cases where filtering is needed for each lower layer reference picture
  + (JCTVC-M0309): scaled reference layer picture offsets
  + (JCTVC-M0274): inter-layer referencing outside of conformance cropping window
  + (JCTVC-M0449 (JCTVC-M0188, JCTVC-M0322 and JCTVC-M0425)): division-free reference layer sample location derivation used in re-sampling process
  + (JCTVC-M0133): the division-free reference layer sample location derivation
  + (JCTVC-M0133): division-free MV scaling
  + Supporting YUV 422 and 444 format decoding
  + Bug fix in motion mapping, adding variable arrary predFlag
* (Common HLS01)Common high level syntax ported from MV-HEVC text JCT3V-D1004\_v1,
  + A group of high level syntax proposals are ported with this track, please see JCT3V-D1004\_v1 for the integration detail for of each proposal

**CONTENTS**

*Page*

[Abstract i](#_Toc373832671)

[8 Decoding process 2](#_Toc373832672)

[8.1 General decoding process 2](#_Toc373832673)

[8.1.1 Decoding process for a coded picture with nuh\_layer\_id equal to 0 4](#_Toc373832674)

[8.2 NAL unit decoding process 5](#_Toc373832675)

[8.3 Slice decoding process 5](#_Toc373832676)

[8.3.2 Decoding process for reference picture set 5](#_Toc373832677)

[8.3.3 Decoding process for generating unavailable reference pictures 9](#_Toc373832678)

[8.3.3.1 General decoding process for generating unavailable reference pictures 9](#_Toc373832679)

[Annex C Hypothetical reference decoder 11](#_Toc373832680)

[C.1 General 11](#_Toc373832681)

[C.2 Operation of coded picture buffer (CPB) and bitstream partition buffer (BPB) 17](#_Toc373832682)

[C.2.1 General 17](#_Toc373832683)

[C.2.2 Timing of decoding unit arrival 18](#_Toc373832684)

[C.2.3 Timing of decoding unit removal and decoding of decoding unit 19](#_Toc373832685)

[C.3 Operation of the decoded picture buffer (DPB) 22](#_Toc373832686)

[C.3.1 General 22](#_Toc373832687)

[C.3.2 Removal of pictures from the DPB 23](#_Toc373832688)

[C.3.3 Picture output 23](#_Toc373832689)

[C.3.4 Current decoded picture marking and storage 24](#_Toc373832690)

[C.4 Bitstream conformance 24](#_Toc373832691)

[C.5 Decoder conformance 26](#_Toc373832692)

[C.5.1 General 26](#_Toc373832693)

[C.5.2 Operation of the output order DPB 27](#_Toc373832694)

[C.5.2.1 General 27](#_Toc373832695)

[C.5.2.2 Output and removal of pictures from the DPB 28](#_Toc373832696)

[C.5.2.3 Picture decoding, marking, additional bumping, and storage 28](#_Toc373832697)

[C.5.2.4 "Bumping" process 29](#_Toc373832698)

[C.6 Demultiplexing process for deriving a bitstream partition 29](#_Toc373832699)

[Annex D Supplemental enhancement information 32](#_Toc373832700)

[D.1 SEI payload syntax 32](#_Toc373832701)

[D.1.1 General SEI message syntax 32](#_Toc373832702)

[Annex E Video usability information 33](#_Toc373832703)

[E.2 VUI semantics 33](#_Toc373832704)

[E.2.1 VUI parameters semantics 33](#_Toc373832705)

[Annex F Common syntax, semantics and decoding processes for multi-layer video coding extensions 34](#_Toc373832706)

[F.1 Scope 34](#_Toc373832707)

[F.2 Normative references 34](#_Toc373832708)

[F.3 Definitions 34](#_Toc373832709)

[F.4 Abbreviations 35](#_Toc373832710)

[F.5 Conventions 35](#_Toc373832711)

[F.6 Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships 35](#_Toc373832712)

[F.7 Syntax and semantics 35](#_Toc373832713)

[F.7.1 Method of specifying syntax in tabular form 35](#_Toc373832714)

[F.7.2 Specification of syntax functions, categories, and descriptors 35](#_Toc373832715)

[F.7.3 Syntax in tabular form 35](#_Toc373832716)

[F.7.3.1 NAL unit syntax 35](#_Toc373832717)

[F.7.3.2 Raw byte sequence payloads and RBSP trailing bits syntax 36](#_Toc373832718)

[F.7.3.3 Profile, tier and level syntax 49](#_Toc373832719)

[F.7.3.4 Scaling list data syntax 49](#_Toc373832720)

[F.7.3.5 Supplemental enhancement information message syntax 49](#_Toc373832721)

[F.7.3.6 Slice segment header syntax 50](#_Toc373832722)

[F.7.3.7 Short-term reference picture set syntax 52](#_Toc373832723)

[F.7.3.8 Slice segment data syntax 53](#_Toc373832724)

[F.7.4 Semantics 53](#_Toc373832725)

[F.7.4.1 General 53](#_Toc373832726)

[F.7.4.2 NAL unit semantics 53](#_Toc373832727)

[F.7.4.3 Raw byte sequence payloads, trailing bits, and byte alignment semantics 55](#_Toc373832728)

[F.7.4.4 Profile, tier and level semantics 72](#_Toc373832729)

[F.7.4.5 Scaling list data semantics 72](#_Toc373832730)

[F.7.4.6 Supplemental enhancement information message semantics 72](#_Toc373832731)

[F.7.4.7 Slice segment header semantics 72](#_Toc373832732)

[F.7.4.8 Short-term reference picture set semantics 74](#_Toc373832733)

[F.7.4.9 Slice segment data semantics 74](#_Toc373832734)

[F.8 Decoding process 75](#_Toc373832735)

[F.8.1 General decoding process 75](#_Toc373832736)

[F.8.1.1 Decoding process for a coded picture with nuh\_layer\_id equal to 0 75](#_Toc373832737)

[F.8.1.2 Decoding process for a coded picture with nuh\_layer\_id greater than 0 75](#_Toc373832738)

[F.8.1.3 Decoding process for starting the decoding of a coded picture with nuh\_layer\_id greater than 0 76](#_Toc373832739)

[F.8.1.4 Decoding process for ending the decoding of a coded picture with nuh\_layer\_id greater than 0 76](#_Toc373832740)

[F.8.1.5 Generation of unavailable reference pictures for pictures first in decoding order within a layer 77](#_Toc373832741)

[F.8.2 NAL unit decoding process 78](#_Toc373832742)

[F.8.3 Slice decoding processes 78](#_Toc373832743)

[F.8.3.1 Decoding process for picture order count 78](#_Toc373832744)

[F.8.3.2 Decoding process for reference picture set 79](#_Toc373832745)

[F.8.3.3 Decoding process for generating unavailable reference pictures 79](#_Toc373832746)

[F.8.3.4 Decoding process for reference picture lists construction 79](#_Toc373832747)

[F.8.3.5 Decoding process for collocated picture and no backward prediction flag 79](#_Toc373832748)

[F.8.4 Decoding process for coding units coded in intra prediction mode 79](#_Toc373832749)

[F.8.5 Decoding process for coding units coded in inter prediction mode 79](#_Toc373832750)

[F.8.6 Scaling, transformation and array construction process prior to deblocking filter process 79](#_Toc373832751)

[F.8.7 In-loop filter process 79](#_Toc373832752)

[F.9 Parsing process 79](#_Toc373832753)

[F.10 Specification of bitstream subsets 79](#_Toc373832754)

[F.11 (Void) 80](#_Toc373832755)

[F.12 Byte stream format 80](#_Toc373832756)

[F.13 Hypothetical reference decoder 80](#_Toc373832757)

[F.14 SEI messages 80](#_Toc373832758)

[F.14.1 SEI message syntax 80](#_Toc373832759)

[F.14.1.1 Layers not present SEI message syntax 80](#_Toc373832760)

[F.14.1.2 Inter-layer constrained tile sets SEI message syntax 80](#_Toc373832761)

[F.14.1.3 Bitstream partition nesting SEI message syntax 81](#_Toc373832762)

[F.14.1.4 Bitstream partition initial arrival time SEI message syntax 81](#_Toc373832763)

[F.14.1.5 Bitstream partition HRD parameters SEI message syntax 82](#_Toc373832764)

[F.14.2 SEI message semantics 82](#_Toc373832765)

[F.14.2.1 Layers not present SEI message semantics 83](#_Toc373832766)

[F.14.2.2 Inter-layer constrained tile sets SEI message semantics 83](#_Toc373832767)

[F.14.2.3 Bitstream partition nesting SEI message semantics 85](#_Toc373832768)

[F.14.2.4 Bitstream partition initial arrival time SEI message semantics 85](#_Toc373832769)

[F.14.2.5 Bitstream partition HRD parameters SEI message semantics 85](#_Toc373832770)

[F.15 Video usability information 86](#_Toc373832771)

[F.15.1 General 86](#_Toc373832772)

[F.15.2 VUI syntax 86](#_Toc373832773)

[F.15.3 VUI semantics 86](#_Toc373832774)

[F.15.3.1 VUI parameters semantics 86](#_Toc373832775)

[F.15.3.2 HRD parameters semantics 86](#_Toc373832776)

[F.15.3.3 Sub-layer HRD parameters semantics 86](#_Toc373832777)

[Annex H Syntax, semantics and decoding processes for scalable extension 87](#_Toc373832778)

[H.1 Scope 87](#_Toc373832779)

[H.2 Normative references 87](#_Toc373832780)

[H.3 Definitions 87](#_Toc373832781)

[H.4 Abbreviations 87](#_Toc373832782)

[H.5 Conventions 87](#_Toc373832783)

[H.6 Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships 87](#_Toc373832784)

[H.6.1 Derivation process for reference layer sample location 87](#_Toc373832785)

[H.6.2 Derivation process for reference layer sample location used in resampling 87](#_Toc373832786)

[H.7 Syntax and semantics 88](#_Toc373832787)

[H.8 Decoding processes 88](#_Toc373832788)

[H.8.1 General decoding process 88](#_Toc373832789)

[H.8.1.1 Decoding process for a coded picture with nuh\_layer\_id greater than 0 88](#_Toc373832790)

[H.8.1.2 Decoding process for inter-layer reference picture set 88](#_Toc373832791)

[H.8.1.3 Marking process for ending the decoding of a coded picture with nuh\_layer\_id greater than 0 89](#_Toc373832792)

[H.8.1.4 Resampling process for inter layer reference pictures 89](#_Toc373832793)

[H.8.2 NAL unit decoding process 96](#_Toc373832794)

[H.8.3 Slice decoding processes 96](#_Toc373832795)

[H.8.3.1 Decoding process for picture order count 96](#_Toc373832796)

[H.8.3.2 Decoding process for reference picture set 96](#_Toc373832797)

[H.8.3.3 Decoding process for generating unavailable reference pictures 96](#_Toc373832798)

[H.8.3.4 Decoding process for reference picture lists construction 96](#_Toc373832799)

[H.8.4 Decoding process for coding units coded in intra prediction mode 97](#_Toc373832800)

[H.8.5 Decoding process for coding units coded in inter prediction mode 97](#_Toc373832801)

[H.8.6 Scaling, transformation and array construction process prior to deblocking filter process 97](#_Toc373832802)

[H.8.7 In-loop filter process 97](#_Toc373832803)

[H.9 Parsing process 98](#_Toc373832804)

[H.10 Specification of bitstream subsets 98](#_Toc373832805)

[H.11 Profiles, tiers, and levels 98](#_Toc373832806)

[H.11.1 Profiles 98](#_Toc373832807)

[H.11.1.1 General 98](#_Toc373832808)

[H.11.1.2 Scalable Main profile 98](#_Toc373832809)

[H.11.2 Tiers and levels 98](#_Toc373832810)

[H.12 Byte stream format 98](#_Toc373832811)

[H.13 Hypothetical reference decoder 98](#_Toc373832812)

[H.14 SEI messages 99](#_Toc373832813)

[H.15 Video usability information 99](#_Toc373832814)

*Add the following definitions to clause 3:*

**3.X base bitstream partition**: A *bitstream partition* that is also a conforming *bitstream* itself.

**3.X bitstream partition**: A sequence of bits, in the form of a *NAL unit stream* or a *byte stream*, that is a subset of a *bitstream* according to a *partitioning*.

**3.X output layer set**: A *layer set* that is associated with a set of target output layers.

*Replace the definition of operation point in clause 3 with the following:*

**3.X operation point**: A bitstream that is created from another *bitstream* by operation of the *sub-bitstream extraction process* with the another *bitstream*, a target highest TemporalId, and a target *layer identifier list* as inputs, and that is associated with a set of target output layers.

NOTE 14 – If the target highest TemporalId of an operation point is equal to the greatest value of TemporalId in the layer set associated with the target layer identification list, the operation point is identical to the layer set. Otherwise it is a subset of the layer set.

*Repl*ace subclauses 7.4.2.4.2 with the following (with differences indicated in turquois):

##### Order of VPS, SPS and PPS RBSPs and their activation

This subclause specifies the activation process of VPSs, SPSs, and PPSs.

NOTE  – The VPS, SPS, and PPS mechanism decouples the transmission of infrequently changing information from the transmission of coded block data. VPSs, SPSs, and PPSs may, in some applications, be conveyed "out-of-band".

A PPS RBSP includes parameters that can be referred to by the coded slice segment NAL units of one or more coded pictures. Each PPS RBSP is initially considered not active for any layer at the start of the operation of the decoding process. At most one PPS RBSP is considered active for each layer at any given moment during the operation of the decoding process, and the activation of any particular PPS RBSP for a particular layer results in the deactivation of the previously-active PPS RBSP for the particular layer (if any).

One PPS RBSP may be the active PPS RBSP for more than one layer. When not explicitly specified, the layer a PPS RBSP is active for is inferred to be the current layer in the context where the active PPS RBSP is referred to.

When a PPS RBSP (with a particular value of pps\_pic\_parameter\_set\_id) is not active for a particular layer and it is referred to by a coded slice segment NAL unit (using a value of slice\_pic\_parameter\_set\_id equal to the pps\_pic\_parameter\_set\_id value) of the particular layer, it is activated for the particular layer. This PPS RBSP is called the active PPS RBSP for the particular layer until it is deactivated by the activation of another PPS RBSP for the particular layer. A PPS RBSP, with that particular value of pps\_pic\_parameter\_set\_id, shall be available to the decoding process prior to its activation, included in at least one access unit with TemporalId less than or equal to the TemporalId of the PPS NAL unit or provided through external means.

Any PPS NAL unit containing the value of pps\_pic\_parameter\_set\_id for the active PPS RBSP for a coded picture (and consequently for the layer containing the coded picture) shall have the same content as that of the active PPS RBSP for the coded picture, unless it follows the last VCL NAL unit of the coded picture and precedes the first VCL NAL unit of another coded picture.

An SPS RBSP includes parameters that can be referred to by one or more PPS RBSPs or one or more SEI NAL units containing an active parameter sets SEI message. Each SPS RBSP is initially considered not active for any layer at the start of the operation of the decoding process. At most one SPS RBSP is considered active for each layer at any given moment during the operation of the decoding process, and the activation of any particular SPS RBSP for a particular layer results in the deactivation of the previously-active SPS RBSP for the particular layer value of nuh\_layer\_id (if any).

One SPS RBSP may be the active SPS RBSP for more than one layer. When not explicitly specified, the layer an SPS RBSP is active for is inferred to be the current layer in the context where the active PPS RBSP is referred to.

When an SPS RBSP (with a particular value of sps\_seq\_parameter\_set\_id) is not already active for a particular layer and it is referred to by activation of a PPS RBSP (in which pps\_seq\_parameter\_set\_id is equal to the sps\_seq\_parameter\_set\_id value) referred to by the particular layer or is referred to by an SEI NAL unit containing an active parameter sets SEI message (in which one of the active\_seq\_parameter\_set\_id[ i ] values is equal to the sps\_seq\_parameter\_set\_id value), it is activated for the particular layer. This SPS RBSP is called the active SPS RBSP for the particular layer until it is deactivated by the activation of another SPS RBSP for the particular layer. An SPS RBSP, with that particular value of sps\_seq\_parameter\_set\_id, shall be available to the decoding process prior to its activation, included in at least one access unit with TemporalId equal to 0 or provided through external means. An activated SPS RBSP for the base layer shall remain active for the entire CVS.

NOTE  – Because an IRAP access unit with NoRaslOutputFlag equal to 1 begins a new CVS and an activated SPS RBSP must remain active for the entire CVS, an SPS RBSP can only be activated by an active parameter sets SEI message when the active parameter sets SEI message is part of an IRAP access unit with NoRaslOutputFlag equal to 1.

Any SPS NAL unit containing the value of sps\_seq\_parameter\_set\_id for the active SPS RBSP for the base layer for a CVS shall have the same content as that of the active SPS RBSP for the base layer for the CVS, unless it follows the last access unit of the CVS and precedes the first VCL NAL unit and the first SEI NAL unit containing an active parameter sets SEI message (when present) of another CVS.

A VPS RBSP includes parameters that can be referred to by one or more SPS RBSPs or one or more SEI NAL units containing an active parameter sets SEI message. Each VPS RBSP is initially considered not active at the start of the operation of the decoding process. At most one VPS RBSP is considered active at any given moment during the operation of the decoding process, and the activation of any particular VPS RBSP results in the deactivation of the previously-active VPS RBSP (if any).

When a VPS RBSP (with a particular value of vps\_video\_parameter\_set\_id) is not already active and it is referred to by activation of an SPS RBSP (in which sps\_video\_parameter\_set\_id is equal to the vps\_video\_parameter\_set\_id value), or is referred to by an SEI NAL unit containing an active parameter sets SEI message (in which active\_video\_parameter\_set\_id is equal to the vps\_video\_parameter\_set\_id value), it is activated. This VPS RBSP is called the active VPS RBSP until it is deactivated by the activation of another VPS RBSP. A VPS RBSP, with that particular value of vps\_video\_parameter\_set\_id, shall be available to the decoding process prior to its activation, included in at least one access unit with TemporalId equal to 0 or provided through external means. An activated VPS RBSP shall remain active for the entire CVS.

NOTE  – Because an IRAP access unit with NoRaslOutputFlag equal to 1 begins a new CVS and an activated VPS RBSP must remain active for the entire CVS, a VPS RBSP can only be activated by an active parameter sets SEI message when the active parameter sets SEI message is part of an IRAP access unit with NoRaslOutputFlag equal to 1.

Any VPS NAL unit containing the value of vps\_video\_parameter\_set\_id for the active VPS RBSP for a CVS shall have the same content as that of the active VPS RBSP for the CVS, unless it follows the last access unit of the CVS and precedes the first VCL NAL unit, the first SPS NAL unit, and the first SEI NAL unit containing an active parameter sets SEI message (when present) of another CVS.

NOTE  – If VPS RBSP, SPS RBSP, or PPS RBSP are conveyed within the bitstream, these constraints impose an order constraint on the NAL units that contain the VPS RBSP, SPS RBSP, or PPS RBSP, respectively. Otherwise (VPS RBSP, SPS RBSP, or PPS RBSP are conveyed by other means not specified in this Specification), they must be available to the decoding process in a timely fashion such that these constraints are obeyed.

All constraints that are expressed on the relationship between the values of the syntax elements and the values of variables derived from those syntax elements in VPSs, SPSs, and PPSs and other syntax elements are expressions of constraints that apply only to the active VPS RBSP, the active SPS RBSP for the base layer, and the active PPS RBSP for the base layer. If any VPS RBSP, SPS RBSP, and PPS RBSP is present that is never activated in the bitstream, its syntax elements shall have values that would conform to the specified constraints if it was activated by reference in an otherwise conforming bitstream.

During operation of the decoding process (see clause 8), the values of parameters of the active VPS, the active SPS for the base layer, and the active PPS RBSP for the base layer are considered in effect. For interpretation of SEI messages, the values of the active VPS RBSP, the active SPS RBSP for the base layer, and the active PPS RBSP for the base layer for the operation of the decoding process for the VCL NAL units of the coded picture in the same access unit are considered in effect unless otherwise specified in the SEI message semantics.

*Replace clause 8, subclauses 8.1, 8.2, 8.3, 8.3.2, 8.3.3, and 8.3.3.1 with the following and add subclause 8.1.1 (with differences indicated in turquois):*

# Decoding process

## General decoding process

Input to this process is a bitstream. Output of this process is a list of decoded pictures.

The variable TargetOptLayerSetIdx, which specifies the index to the list of the output layer sets specified by the VPS, of the target output layer set, is specified as follows:

– If some external means, not specified in this Specification, is available to set TargetOptLayerSetIdx, TargetOptLayerSetIdx is set by the external means.

– Otherwise, if the decoding process is invoked in a bitstream conformance test as specified in subclause C.1, TargetOptLayerSetIdx is set as specified in subclause C.1.

– Otherwise, TargetOptLayerSetIdx is set equal to 0.

The variable TargetDecLayerSetIdx, the layer identifier list TargetOptLayerIdList, which specifies the list of nuh\_layer\_id values, in increasing order of nuh\_layer\_id values, of the pictures to be output, and the layer identifier list TargetDecLayerIdList, which specifies the list of nuh\_layer\_id values, in increasing order of nuh\_layer\_id values, of the NAL units to be decoded, are specified as follows:

TargetDecLayerSetIdx = output\_layer\_set\_idx\_minus1[ TargetOptLayerSetIdx ] + 1  
 lsIdx = TargetDecLayerSetIdx  
 for( k = 0, j = 0; j < NumLayersInIdList[ lsIdx ]; j++ ) {  
 TargetDecLayerIdList[ j ] = LayerSetLayerIdList[ lsIdx ][ j ] (8‑1)  
 if( output\_layer\_flag[ lsIdx ][ j ] )  
 TargetOptLayerIdList[ k++ ] = LayerSetLayerIdList[ lsIdx ][ j ]  
 }

The variable HighestTid, which identifies the highest temporal sub-layer to be decoded, is specified as follows:

– If some external means, not specified in this Specification, is available to set HighestTid, HighestTid is set by the external means.

– Otherwise, if the decoding process is invoked in a bitstream conformance test as specified in subclause C.1, HighestTid is set as specified in subclause C.1.

– Otherwise, HighestTid is set equal to sps\_max\_sub\_layers\_minus1.

The sub-bitstream extraction process as specified in clause 10 is applied with the bitstream, HighestTid, and TargetDecLayerIdList as inputs, and the output is assigned to a bitstream referred to as BitstreamToDecode.

The decoding processes specified in the remainder of this subclause apply to each coded picture, referred to as the current picture and denoted by the variable CurrPic, in BitstreamToDecode.

Depending on the value of chroma\_format\_idc, the number of sample arrays of the current picture is as follows:

– If chroma\_format\_idc is equal to 0, the current picture consists of 1 sample array SL.

– Otherwise (chroma\_format\_idc is not equal to 0), the current picture consists of 3 sample arrays SL, SCb, SCr.

The decoding process for the current picture takes as inputs the syntax elements and upper-case variables from clause 7. When interpreting the semantics of each syntax element in each NAL unit, the term "the bitstream" (or part thereof, e.g. a CVS of the bitstream) refers to BitstreamToDecode (or part thereof).

When the current picture is an IRAP picture, the variable HandleCraAsBlaFlag is derived as specified in the following:

– If some external means not specified in this Specification is available to set the variable HandleCraAsBlaFlag to a value for the current picture, the variable HandleCraAsBlaFlag is set equal to the value provided by the external means.

– Otherwise, the variable HandleCraAsBlaFlag is set equal to 0.

When the current picture is an IRAP picture and has nuh\_layer\_id equal to 0, the following applies:

– The variable NoClrasOutputFlag is specified as follows:

– If the current picture is the first picture in the bitstream, NoClrasOutputFlag is set equal to 1.

– Otherwise, if the current picture is a BLA picture or a CRA with HandleCraAsBlaFlag equal to 1, NoClrasOutputFlag is set equal to 1.

– Otherwise, if the current picture is an IDR picture and cross\_layer\_bla\_flag is equal to1, NoClrasOutputFlag is set equal to 1.

– Otherwise, if some external means not specified in this Specification is available to set NoClrasOutputFlag, NoClrasOutputFlag is set by the external means.

– Otherwise, NoClrasOutputFlag is set equal to 0.

– When NoClrasOutputFlag is equal to 1, the variable LayerInitializedFlag[ i ] is set equal to 0 for all values of i from 0 to vps\_max\_layer\_id, inclusive, and the variable FirstPicInLayerDecodedFlag[ i ] is set equal to 0 for all values of i from 1 to vps\_max\_layer\_id, inclusive.

The decoding process is specified such that all decoders will produce numerically identical cropped decoded pictures. Any decoding process that produces identical cropped decoded pictures to those produced by the process described herein (with the correct output order or output timing, as specified) conforms to the decoding process requirements of this Specification.

When the current picture is an IRAP picture, the following applies:

– If the current picture with a particular value of nuh\_layer\_id is an IDR picture, a BLA picture, the first picture with that particular value of nuh\_layer\_id in the bitstream in decoding order, or the first picture with that particular value of nuh\_layer\_id that follows an end of sequence NAL unit in decoding order, the variable NoRaslOutputFlag is set equal to 1.

– Otherwise, if LayerInitializedFlag[ nuh\_layer\_id ] is equal to 0 and LayerInitializedFlag[ refLayerId ] is equal to 1 for all values of refLayerId equal to RefLayerId[ nuh\_layer\_id ][ j ], where j is in the range of 0 to NumDirectRefLayers[ nuh\_layer\_id ] – 1, inclusive, the variable NoRaslOutputFlag is set equal to 1.

– Otherwise, the variable NoRaslOutputFlag is set equal to HandleCraAsBlaFlag.

When the current picture is an IRAP picture with NoRaslOutputFlag equal to 1 and one of the following conditions is true, LayerInitializedFlag[ nuh\_layer\_id ] is set equal to 1:

– nuh\_layer\_id is equal to 0.

– LayerInitializedFlag[ nuh\_layer\_id ] is equal to 0 and LayerInitializedFlag[ refLayerId ] is equal to 1 for all values of refLayerId equal to RefLayerId[ nuh\_layer\_id ][ j ], where j is in the range of 0 to NumDirectRefLayers[ nuh\_layer\_id ] – 1, inclusive.

When the current picture is a BLA picture that has nal\_unit\_type equal to BLA\_W\_LP or is a CRA picture, the following applies:

– If some external means not specified in this Specification is available to set the variable UseAltCpbParamsFlag to a value, UseAltCpbParamsFlag is set equal to the value provided by the external means.

– Otherwise, the value of UseAltCpbParamsFlag is set equal to 0.

Depending on the value of separate\_colour\_plane\_flag, the decoding process is structured as follows:

– If separate\_colour\_plane\_flag is equal to 0, the following decoding process is invoked a single time with the current picture being the output.

– Otherwise (separate\_colour\_plane\_flag is equal to 1), the following decoding process is invoked three times. Inputs to the decoding process are all NAL units of the coded picture with identical value of colour\_plane\_id. The decoding process of NAL units with a particular value of colour\_plane\_id is specified as if only a CVS with monochrome colour format with that particular value of colour\_plane\_id would be present in the bitstream. The output of each of the three decoding processes is assigned to one of the 3 sample arrays of the current picture, with the NAL units with colour\_plane\_id equal to 0, 1, and 2 being assigned to SL, SCb, and SCr, respectively.

NOTE – The variable ChromaArrayType is derived as equal to 0 when separate\_colour\_plane\_flag is equal to 1 and chroma\_format\_idc is equal to 3. In the decoding process, the value of this variable is evaluated resulting in operations identical to that of monochrome pictures (when chroma\_format\_idc is equal to 0).

When the current picture has nuh\_layer\_id equal to 0, the decoding process for a coded picture with nuh\_layer\_id equal to 0 as specified in subclause 8.1.1 is invoked.

### Decoding process for a coded picture with nuh\_layer\_id equal to 0

The decoding process operates as follows for the current picture CurrPic:

1. The decoding of NAL units is specified in subclause 8.2.
2. The processes in subclause 8.3 specify the following decoding processes using syntax elements in the slice segment layer and above:

– Variables and functions relating to picture order count are derived in subclause 8.3.1. This needs to be invoked only for the first slice segment of a picture.

– The decoding process for RPS in subclause 8.3.2 is invoked, wherein reference pictures may be marked as "unused for reference" or "used for long-term reference". This needs to be invoked only for the first slice segment of a picture.

– When the current picture is a BLA picture or is a CRA picture with NoRaslOutputFlag equal to 1, the decoding process for generating unavailable reference pictures specified in subclause 8.3.3 is invoked, which needs to be invoked only for the first slice segment of a picture.

– PicOutputFlag is set as follows:

– If the current picture is a RASL picture and NoRaslOutputFlag of the associated IRAP picture is equal to 1, PicOutputFlag is set equal to 0.

– Otherwise, PicOutputFlag is set equal to pic\_output\_flag.

– At the beginning of the decoding process for each P or B slice, the decoding process for reference picture lists construction specified in subclause 8.3.4 is invoked for derivation of reference picture list 0 (RefPicList0) and, when decoding a B slice, reference picture list 1 (RefPicList1).

1. The processes in subclauses 8.4, 8.5, 8.6, and 8.7 specify decoding processes using syntax elements in all syntax structure layers. It is a requirement of bitstream conformance that the coded slices of the picture shall contain slice segment data for every coding tree unit of the picture, such that the division of the picture into slices, the division of the slices into slice segments, and the division of the slice segments into coding tree units each form a partitioning of the picture.
2. After all slices of the current picture have been decoded, the decoded picture is marked as "used for short-term reference".

## NAL unit decoding process

Inputs to this process are VCL NAL units of the current picture and their associated non-VCL NAL units.

Outputs of this process are the parsed RBSP syntax structures encapsulated within the NAL units of the access unit containing the current picture.

The decoding process for each NAL unit extracts the RBSP syntax structure from the NAL unit and then parses the RBSP syntax structure.

## Slice decoding process

[Ed. (CY): consider moving the remaining part of 8.3, the entire 8.1 and entire 8.2 to Annex F. To be confirmed before the action is taken.]

### Decoding process for reference picture set

This process is invoked once per picture, after decoding of a slice header but prior to the decoding of any coding unit and prior to the decoding process for reference picture list construction for the slice as specified in subclause 8.3.4. This process may result in one or more reference pictures in the DPB being marked as "unused for reference" or "used for long-term reference".

NOTE 1 – The RPS is an absolute description of the reference pictures used in the decoding process of the current and future coded pictures. The RPS signalling is explicit in the sense that all reference pictures included in the RPS are listed explicitly.

A decoded picture in the DPB can be marked as "unused for reference", "used for short-term reference", or "used for long-term reference", but only one among these three at any given moment during the operation of the decoding process. Assigning one of these markings to a picture implicitly removes another of these markings when applicable. When a picture is referred to as being marked as "used for reference", this collectively refers to the picture being marked as "used for short-term reference" or "used for long-term reference" (but not both).

The variable currPicLayerId is set equal to nuh\_layer\_id of the current picture.

When the current picture is an IRAP picture with nuh\_layer\_id equal to 0 and NoClrasOutputFlag is equal to 1, all reference pictures with any value of nuh\_layer\_id currently in the DPB (if any) are marked as "unused for reference".

When the current picture is an IRAP picture with NoRaslOutputFlag equal to 1, all reference pictures with nuh\_layer\_id equal to currPicLayerId currently in the DPB (if any) are marked as "unused for reference".

Short-term reference pictures are identified by their PicOrderCntVal values. Long-term reference pictures are identified either by their PicOrderCntVal values or their slice\_pic\_order\_cnt\_lsb values.

Five lists of picture order count values are constructed to derive the RPS. These five lists are PocStCurrBefore, PocStCurrAfter, PocStFoll, PocLtCurr, and PocLtFoll, with NumPocStCurrBefore, NumPocStCurrAfter, NumPocStFoll, NumPocLtCurr, and NumPocLtFoll number of elements, respectively. The five lists and the five variables are derived as follows:

* If the current picture is an IDR picture, PocStCurrBefore, PocStCurrAfter, PocStFoll, PocLtCurr, and PocLtFoll are all set to be empty, and NumPocStCurrBefore, NumPocStCurrAfter, NumPocStFoll, NumPocLtCurr, and NumPocLtFoll are all set equal to 0.
* Otherwise, the following applies:

for( i = 0, j = 0, k = 0; i < NumNegativePics[ CurrRpsIdx ] ; i++ )  
 if( UsedByCurrPicS0[ CurrRpsIdx ][ i ] )  
 PocStCurrBefore[ j++ ] = PicOrderCntVal + DeltaPocS0[ CurrRpsIdx ][ i ]  
 else  
 PocStFoll[ k++ ] = PicOrderCntVal + DeltaPocS0[ CurrRpsIdx ][ i ]  
NumPocStCurrBefore = j  
  
for( i = 0, j = 0; i < NumPositivePics[ CurrRpsIdx ]; i++ )  
 if( UsedByCurrPicS1[ CurrRpsIdx ][ i ] )  
 PocStCurrAfter[ j++ ] = PicOrderCntVal + DeltaPocS1[ CurrRpsIdx ][ i ]  
 else  
 PocStFoll[ k++ ] = PicOrderCntVal + DeltaPocS1[ CurrRpsIdx ][ i ]  
NumPocStCurrAfter = j  
NumPocStFoll = k (8‑2)  
for( i = 0, j = 0, k = 0; i < num\_long\_term\_sps + num\_long\_term\_pics; i++ ) {  
 pocLt = PocLsbLt[ i ]  
 if( delta\_poc\_msb\_present\_flag[ i ] )  
 pocLt += PicOrderCntVal − DeltaPocMsbCycleLt[ i ] \* MaxPicOrderCntLsb −  
 PicOrderCntVal & ( MaxPicOrderCntLsb − 1 )  
 if( UsedByCurrPicLt[ i ] ) {  
 PocLtCurr[ j ] = pocLt  
 CurrDeltaPocMsbPresentFlag[ j++ ] = delta\_poc\_msb\_present\_flag[ i ]  
 } else {  
 PocLtFoll[ k ] = pocLt  
 FollDeltaPocMsbPresentFlag[ k++ ] = delta\_poc\_msb\_present\_flag[ i ]  
 }  
}  
NumPocLtCurr = j  
NumPocLtFoll = k

where PicOrderCntVal is the picture order count of the current picture as specified in subclause 8.3.1.

NOTE 2 – A value of CurrRpsIdx in the range of 0 to num\_short\_term\_ref\_pic\_sets − 1, inclusive, indicates that a candidate short-term RPS from the active SPS for the current layer is being used, where CurrRpsIdx is the index of the candidate short-term RPS into the list of candidate short-term RPSs signalled in the active SPS for the current layer. CurrRpsIdx equal to num\_short\_term\_ref\_pic\_sets indicates that the short-term RPS of the current picture is directly signalled in the slice header.

For each i in the range of 0 to NumPocLtCurr − 1, inclusive, when CurrDeltaPocMsbPresentFlag[ i ] is equal to 1, it is a requirement of bitstream conformance that the following conditions apply:

* There shall be no j in the range of 0 to NumPocStCurrBefore − 1, inclusive, for which PocLtCurr[ i ] is equal to PocStCurrBefore[ j ].
* There shall be no j in the range of 0 to NumPocStCurrAfter − 1, inclusive, for which PocLtCurr[ i ] is equal to PocStCurrAfter[ j ].
* There shall be no j in the range of 0 to NumPocStFoll − 1, inclusive, for which PocLtCurr[ i ] is equal to PocStFoll[ j ].
* There shall be no j in the range of 0 to NumPocLtCurr − 1, inclusive, where j is not equal to i, for which PocLtCurr[ i ] is equal to PocLtCurr[ j ].

For each i in the range of 0 to NumPocLtFoll − 1, inclusive, when FollDeltaPocMsbPresentFlag[ i ] is equal to 1, it is a requirement of bitstream conformance that the following conditions apply:

* There shall be no j in the range of 0 to NumPocStCurrBefore − 1, inclusive, for which PocLtFoll[ i ] is equal to PocStCurrBefore[ j ].
* There shall be no j in the range of 0 to NumPocStCurrAfter − 1, inclusive, for which PocLtFoll[ i ] is equal to PocStCurrAfter[ j ].
* There shall be no j in the range of 0 to NumPocStFoll − 1, inclusive, for which PocLtFoll[ i ] is equal to PocStFoll[ j ].
* There shall be no j in the range of 0 to NumPocLtFoll − 1, inclusive, where j is not equal to i, for which PocLtFoll[ i ] is equal to PocLtFoll[ j ].
* There shall be no j in the range of 0 to NumPocLtCurr − 1, inclusive, for which PocLtFoll[ i ] is equal to PocLtCurr[ j ].

For each i in the range of 0 to NumPocLtCurr − 1, inclusive, when CurrDeltaPocMsbPresentFlag[ i ] is equal to 0, it is a requirement of bitstream conformance that the following conditions apply:

* There shall be no j in the range of 0 to NumPocStCurrBefore − 1, inclusive, for which PocLtCurr[ i ] is equal to ( PocStCurrBefore[ j ] & ( MaxPicOrderCntLsb − 1 ) ).
* There shall be no j in the range of 0 to NumPocStCurrAfter − 1, inclusive, for which PocLtCurr[ i ] is equal to ( PocStCurrAfter[ j ] & ( MaxPicOrderCntLsb − 1 ) ).
* There shall be no j in the range of 0 to NumPocStFoll − 1, inclusive, for which PocLtCurr[ i ] is equal to ( PocStFoll[ j ] & ( MaxPicOrderCntLsb − 1 ) ).
* There shall be no j in the range of 0 to NumPocLtCurr − 1, inclusive, where j is not equal to i, for which PocLtCurr[ i ] is equal to ( PocLtCurr[ j ] & ( MaxPicOrderCntLsb − 1 ) ).

For each i in the range of 0 to NumPocLtFoll − 1, inclusive, when FollDeltaPocMsbPresentFlag[ i ] is equal to 0, it is a requirement of bitstream conformance that the following conditions apply:

* There shall be no j in the range of 0 to NumPocStCurrBefore − 1, inclusive, for which PocLtFoll[ i ] is equal to ( PocStCurrBefore[ j ] & ( MaxPicOrderCntLsb − 1 ) ).
* There shall be no j in the range of 0 to NumPocStCurrAfter − 1, inclusive, for which PocLtFoll[ i ] is equal to ( PocStCurrAfter[ j ] & ( MaxPicOrderCntLsb − 1 ) ).
* There shall be no j in the range of 0 to NumPocStFoll − 1, inclusive, for which PocLtFoll[ i ] is equal to ( PocStFoll[ j ] & ( MaxPicOrderCntLsb − 1 ) ).
* There shall be no j in the range of 0 to NumPocLtFoll − 1, inclusive, where j is not equal to i, for which PocLtFoll[ i ] is equal to ( PocLtFoll[ j ] & ( MaxPicOrderCntLsb − 1 ) ).
* There shall be no j in the range of 0 to NumPocLtCurr − 1, inclusive, for which PocLtFoll[ i ] is equal to ( PocLtCurr[ j ] & ( MaxPicOrderCntLsb − 1 ) ).

The variable NumPicTotalCurr is derived as specified in subclause 7.4.7.2. It is a requirement of bitstream conformance that the following applies to the value of NumPicTotalCurr:

* If currPicLayerId is equal to 0 and the current picture is a BLA or CRA picture, the value of NumPicTotalCurr shall be equal to 0.
* Otherwise, when the current picture contains a P or B slice, the value of NumPicTotalCurr shall not be equal to 0.

The RPS of the current picture consists of five RPS lists; RefPicSetStCurrBefore, RefPicSetStCurrAfter, RefPicSetStFoll, RefPicSetLtCurr and RefPicSetLtFoll. RefPicSetStCurrBefore, RefPicSetStCurrAfter, and RefPicSetStFoll are collectively referred to as the short-term RPS. RefPicSetLtCurr and RefPicSetLtFoll are collectively referred to as the long-term RPS.

NOTE 3 – RefPicSetStCurrBefore, RefPicSetStCurrAfter, and RefPicSetLtCurr contain all reference pictures that may be used for inter prediction of the current picture and one or more pictures that follow the current picture in decoding order. RefPicSetStFoll and RefPicSetLtFoll consist of all reference pictures that are *not* used for inter prediction of the current picture but may be used in inter prediction for one or more pictures that follow the current picture in decoding order.

The derivation process for the RPS and picture marking are performed according to the following ordered steps:

1. The following applies:

for( i = 0; i < NumPocLtCurr; i++ )  
 if( !CurrDeltaPocMsbPresentFlag[ i ] )  
 if( there is a reference picture picX in the DPB with PicOrderCntVal & ( MaxPicOrderCntLsb − 1 )  
 equal to PocLtCurr[ i ] and nuh\_layer\_id equal to currPicLayerId )  
 RefPicSetLtCurr[ i ] = picX  
 else  
 RefPicSetLtCurr[ i ] = "no reference picture"  
 else  
 if( there is a reference picture picX in the DPB with PicOrderCntVal equal to PocLtCurr[ i ]  
 and nuh\_layer\_id equal to currPicLayerId )  
 RefPicSetLtCurr[ i ] = picX  
 else  
 RefPicSetLtCurr[ i ] = "no reference picture" (8‑3)  
for( i = 0; i < NumPocLtFoll; i++ )  
 if( !FollDeltaPocMsbPresentFlag[ i ] )  
 if( there is a reference picture picX in the DPB with PicOrderCntVal & ( MaxPicOrderCntLsb − 1 )  
 equal to PocLtFoll[ i ] and nuh\_layer\_id equal to currPicLayerId )  
 RefPicSetLtFoll[ i ] = picX  
 else  
 RefPicSetLtFoll[ i ] = "no reference picture"  
 else  
 if( there is a reference picture picX in the DPB with PicOrderCntVal equal to PocLtFoll[ i ]  
 and nuh\_layer\_id equal to currPicLayerId )  
 RefPicSetLtFoll[ i ] = picX  
 else  
 RefPicSetLtFoll[ i ] = "no reference picture"

1. All reference pictures that are included in RefPicSetLtCurr or RefPicSetLtFoll and have nuh\_layer\_id equal to currPicLayerId are marked as "used for long-term reference".
2. The following applies:

for( i = 0; i < NumPocStCurrBefore; i++ )  
 if( there is a short-term reference picture picX in the DPB  
 with PicOrderCntVal equal to PocStCurrBefore[ i ] and nuh\_layer\_id equal to currPicLayerId )  
 RefPicSetStCurrBefore[ i ] = picX  
 else  
 RefPicSetStCurrBefore[ i ] = "no reference picture"

for( i = 0; i < NumPocStCurrAfter; i++ )  
 if( there is a short-term reference picture picX in the DPB  
 with PicOrderCntVal equal to PocStCurrAfter[ i ] and nuh\_layer\_id equal to currPicLayerId )  
 RefPicSetStCurrAfter[ i ] = picX  
 else  
 RefPicSetStCurrAfter[ i ] = "no reference picture" (8‑4)

for( i = 0; i < NumPocStFoll; i++ )  
 if( there is a short-term reference picture picX in the DPB  
 with PicOrderCntVal equal to PocStFoll[ i ] and nuh\_layer\_id equal to currPicLayerId )  
 RefPicSetStFoll[ i ] = picX  
 else  
 RefPicSetStFoll[ i ] = "no reference picture"

1. All reference pictures in the DPB that are not included in RefPicSetLtCurr, RefPicSetLtFoll, RefPicSetStCurrBefore, RefPicSetStCurrAfter, or RefPicSetStFoll and have nuh\_layer\_id equal to currPicLayerId are marked as "unused for reference".

NOTE 4 – There may be one or more entries in the RPS lists that are equal to "no reference picture" because the corresponding pictures are not present in the DPB. Entries in RefPicSetStFoll or RefPicSetLtFoll that are equal to "no reference picture" should be ignored. An unintentional picture loss should be inferred for each entry in RefPicSetStCurrBefore, RefPicSetStCurrAfter, or RefPicSetLtCurr that is equal to "no reference picture".

NOTE 5 – A picture cannot be included in more than one of the five RPS lists.

It is a requirement of bitstream conformance that the RPS is restricted as follows:

* There shall be no entry in RefPicSetStCurrBefore, RefPicSetStCurrAfter, or RefPicSetLtCurr for which one or more of the following are true:
  + The entry is equal to "no reference picture".
  + The entry is a sub-layer non-reference picture and has TemporalId equal to that of the current picture.
  + The entry is a picture that has TemporalId greater than that of the current picture.
* There shall be no entry in RefPicSetLtCurr or RefPicSetLtFoll for which the difference between the picture order count value of the current picture and the picture order count value of the entry is greater than or equal to 224.
* When the current picture is a TSA picture, there shall be no picture included in the RPS with TemporalId greater than or equal to the TemporalId of the current picture.
* When the current picture is an STSA picture, there shall be no picture included in RefPicSetStCurrBefore, RefPicSetStCurrAfter, or RefPicSetLtCurr that has TemporalId equal to that of the current picture.
* When the current picture is a picture that follows, in decoding order, an STSA picture that has TemporalId equal to that of the current picture, there shall be no picture that has TemporalId equal to that of the current picture included in RefPicSetStCurrBefore, RefPicSetStCurrAfter, or RefPicSetLtCurr that precedes the STSA picture in decoding order.
* When the current picture is a CRA picture, there shall be no picture included in the RPS that precedes, in decoding order, any preceding IRAP picture in decoding order (when present).
* When the current picture is a trailing picture, there shall be no picture in RefPicSetStCurrBefore, RefPicSetStCurrAfter, or RefPicSetLtCurr that was generated by the decoding process for generating unavailable reference pictures as specified in clause 8.3.3.
* When the current picture is a trailing picture, there shall be no picture in the RPS that precedes the associated IRAP picture in output order or decoding order.
* When the current picture is a RADL picture, there shall be no picture included in RefPicSetStCurrBefore, RefPicSetStCurrAfter, or RefPicSetLtCurr that is any of the following:
  + A RASL picture
  + A picture that was generated by the decoding process for generating unavailable reference pictures as specified in clause 8.3.3
  + A picture that precedes the associated IRAP picture in decoding order
* When sps\_temporal\_id\_nesting\_flag is equal to 1, the following applies:
  + Let tIdA be the value of TemporalId of the current picture picA.
  + Any picture picB with TemporalId equal to tIdB that is less than or equal to tIdA shall not be included in RefPicSetStCurrBefore, RefPicSetStCurrAfter, or RefPicSetLtCurr of picA when there exists a picture picC that has TemporalId less than tIdB, follows picB in decoding order, and precedes picA in decoding order.

### Decoding process for generating unavailable reference pictures

#### General decoding process for generating unavailable reference pictures

This process is invoked once per coded picture when the current picture is a BLA picture or is a CRA picture with NoRaslOutputFlag equal to 1.

NOTE – This process is primarily specified only for the specification of syntax constraints for RASL pictures. The entire specification of the decoding process for RASL pictures associated with an IRAP picture that has NoRaslOutputFlag equal to 1 is included herein only for purposes of specifying constraints on the allowed syntax content of such RASL pictures. During the decoding process, any RASL pictures associated with an IRAP picture that has NoRaslOutputFlag equal to 1 may be ignored, as these pictures are not specified for output and have no effect on the decoding process of any other pictures that are specified for output. However, in HRD operations as specified in Annex C, RASL access units may need to be taken into consideration in derivation of CPB arrival and removal times.

When this process is invoked, the following applies:

* For each RefPicSetStFoll[ i ], with i in the range of 0 to NumPocStFoll − 1, inclusive, that is equal to "no reference picture", a picture is generated as specified in subclause 8.3.3.2, and the following applies:
* The value of PicOrderCntVal for the generated picture is set equal to PocStFoll[ i ].
* The value of PicOutputFlag for the generated picture is set equal to 0.
* The generated picture is marked as "used for short-term reference".
* RefPicSetStFoll[ i ] is set to be the generated reference picture.
* The value of nuh\_layer\_id for the generated picture is inferred to be equal to nuh\_layer\_id.
* For each RefPicSetLtFoll[ i ], with i in the range of 0 to NumPocLtFoll − 1, inclusive, that is equal to "no reference picture", a picture is generated as specified in subclause 8.3.3.2, and the following applies:
* The value of PicOrderCntVal for the generated picture is set equal to PocLtFoll[ i ].
* The value of slice\_pic\_order\_cnt\_lsb for the generated picture is inferred to be equal to ( PocLtFoll[ i ] & ( MaxPicOrderCntLsb – 1 ) ).
* The value of PicOutputFlag for the generated picture is set equal to 0.
* The generated picture is marked as "used for long-term reference".
* RefPicSetLtFoll[ i ] is set to be the generated reference picture.
* The value of nuh\_layer\_id for the generated picture is inferred to be equal to nuh\_layer\_id.

*Replace Annex C with the following (with differences indicated in turquois):*

1. Annex C  
     
   Hypothetical reference decoder

(This annex forms an integral part of this Recommendation | International Standard)

* 1. **General**

This annex specifies the hypothetical reference decoder (HRD) and its use to check bitstream and decoder conformance.

Two types of bitstreams or bitstream subsets are subject to HRD conformance checking for this Specification. The first type, called a Type I bitstream, is a NAL unit stream containing only the VCL NAL units and NAL units with nal\_unit\_type equal to FD\_NUT (filler data NAL units) for all access units in the bitstream. The second type, called a Type II bitstream, contains, in addition to the VCL NAL units and filler data NAL units for all access units in the bitstream, at least one of the following:

– additional non-VCL NAL units other than filler data NAL units,

– all leading\_zero\_8bits, zero\_byte, start\_code\_prefix\_one\_3bytes, and trailing\_zero\_8bits syntax elements that form a byte stream from the NAL unit stream (as specified in Annex B).

Figure C‑1 shows the types of bitstream conformance points checked by the HRD.



**Figure C‑1 – Structure of byte streams and NAL unit streams for HRD conformance checks**

The syntax elements of non-VCL NAL units (or their default values for some of the syntax elements), required for the HRD, are specified in the semantic subclauses of clause 7, Annexes D and E.

Two types of HRD parameter sets (NAL HRD parameters and VCL HRD parameters) are used. The HRD parameter sets are signalled through the hrd\_parameters( ) syntax structure, which may be part of the SPS syntax structure or the VPS syntax structure.

Multiple tests may be needed for checking the conformance of a bitstream, which is referred to as the bitstream under test. For each test, the following steps apply in the order listed:

1. An operation point under test, denoted as TargetOp, is selected by selecting a target output layer set identified by TargetOptLayerSetIdx and selecting a target highest TemporalId value HighestTid. The value of TargetOptLayerSetIdx shall be in the range of 0 to NumOutputLayerSets − 1, inclusive. The value of HighestTid shall be in the range of 0 to vps\_max\_sub\_layers\_minus1, inclusive. The variables TargetDecLayerSetIdx, TargetOptLayerIdList, and TargetDecLayerIdList are then derived as specified by Equation 8‑1. The operation point under test has OptLayerIdList equal to TargetOptLayerIdList, OpLayerIdList equal to TargetDecLayerIdList, and OpTid equal to HighestTid.
2. The sub-bitstream extraction process as specified in clause 10 is invoked with the bitstream under test, HighestTid, and TargetDecLayerIdList as inputs, and the output is assigned to BitstreamToDecode.
3. When both the vps\_vui\_bsp\_hrd\_parameters( ) syntax structure is present in the active VPS and num\_bitstream\_partitions[ TargetDecLayerSetIdx ] is greater than 1 or both a bitstream partition HRD parameters SEI message is present and the SEI message contains syntax element sei\_num\_bitstream\_partitions\_minus1[ TargetDecLayerSetIdx ] greater than 0, either the bitstream-specific CPB operation or the bitstream-partition-specific CPB operation is selected for a conformance test, and both CPB operations shall be tested for checking the conformance of a bitstream. When the bitstream-specific CPB operation is tested, the subsequent steps apply for the bitstream under test. When the bitstream-partition-specific CPB operation is tested, the subsequent steps apply to each bitstream partition of the bitstream under test, referred to as the bitstream partition under test. When the bitstream-partition-specific CPB operation is tested and the input to the HRD is a bitstream, the bitstream partitions are derived with the demultiplexing process for deriving a bitstream partition in subclause C.6.
4. The hrd\_parameters( ) syntax structure and the sub\_layer\_hrd\_parameters( ) syntax structure applicable to TargetOp are selected as follows:

– If the bitstream-specific CPB operation is tested, the following applies:

– If TargetDecLayerIdList contains all nuh\_layer\_id values present in the bitstream under test, the hrd\_parameters( ) syntax structure in the active SPS for the base layer (or provided through an external means not specified in this Specification) is selected.

– Otherwise, the hrd\_parameters( ) syntax structure in the active VPS (or provided through some external means not specified in this Specification) that applies to TargetOp is selected.

– Otherwise, the hrd\_parameters( ) syntax structure is selected as follows:

– Either one of the hrd\_parameters( ) syntax structures in the following conditions can be selected, if both of the following conditions are true:

– The vps\_vui\_bsp\_hrd\_parameters( ) syntax structure is present in the active VPS (or is available through some external means not specified in this Specification) and contains a hrd\_parameters( ) syntax structure that applies to TargetOp and to the bitstream partition under test.

– A bitstream partition HRD parameters SEI message that is included in a scalable nesting SEI message that applies to TargetOp and contains a hrd\_parameters( ) syntax structure that applies to TargetOp and to the bitstream partition under test is present (or is available through some external means not specified in this Specification).

– Otherwise, if the vps\_vui\_bsp\_hrd\_parameters( ) syntax structure is present in the active VPS (or is available through some external means not specified in this Specification) and contains a hrd\_parameters( ) syntax structure that applies to TargetOp and the bitstream partition under test, that hrd\_parameters( ) syntax structure is selected.

– Otherwise, a hrd\_parameters( ) syntax structure that applies to the bitstream partition under test in the bitstream partition HRD parameters SEI message that is included in a scalable nesting SEI message that applies to TargetOp shall be present (or shall be available through some external means not specified in this Specification) and is selected.

Within the selected hrd\_parameters( ) syntax structure, if BitstreamToDecode is a Type I bitstream, the sub\_layer\_hrd\_parameters( HighestTid ) syntax structure that immediately follows the condition "if( vcl\_hrd\_parameters\_present\_flag )" is selected and the variable NalHrdModeFlag is set equal to 0; otherwise (BitstreamToDecode is a Type II bitstream), the sub\_layer\_hrd\_parameters( HighestTid ) syntax structure that immediately follows either the condition "if( vcl\_hrd\_parameters\_present\_flag )" (in this case the variable NalHrdModeFlag is set equal to 0) or the condition "if( nal\_hrd\_parameters\_present\_flag )" (in this case the variable NalHrdModeFlag is set equal to 1) is selected. When BitstreamToDecode is a Type II bitstream and NalHrdModeFlag is equal to 0, all non-VCL NAL units except filler data NAL units, and all leading\_zero\_8bits, zero\_byte, start\_code\_prefix\_one\_3bytes, and trailing\_zero\_8bits syntax elements that form a byte stream from the NAL unit stream (as specified in Annex B), when present, are discarded from BitstreamToDecode, and the remaining bitstream is assigned to BitstreamToDecode.

1. An access unit associated with a buffering period SEI message (present in BitstreamToDecode or available through external means not specified in this Specification) applicable to TargetOp is selected as the HRD initialization point and referred to as access unit 0. An applicable buffering period SEI message is available through external means not specified in this Specification or is selected from access unit 0 as follows:

– If the bitstream-specific CPB operation is tested, the following applies:

– If TargetDecLayerIdList contains all nuh\_layer\_id values present in the bitstream under test, a non-nested buffering period SEI message is selected.

– Otherwise, a buffering period SEI message included in the scalable nesting SEI message with bitstream\_subset\_flag equal to 1 and applicable to TargetOp is selected.

– Otherwise, a buffering period SEI message included in the bitstream partition nesting SEI message applicable to the bitstream partition under test is selected.

1. For each access unit in BitstreamToDecode starting from access unit 0, the buffering period SEI message (present in BitstreamToDecode or available through external means not specified in this Specification) that is associated with the access unit and applies to TargetOp is selected, the picture timing SEI message (present in BitstreamToDecode or available through external means not specified in this Specification) that is associated with the access unit and applies to TargetOp is selected, and when SubPicHrdFlag is equal to 1 and sub\_pic\_cpb\_params\_in\_pic\_timing\_sei\_flag is equal to 0, the decoding unit information SEI messages (present in BitstreamToDecode or available through external means not specified in this Specification) that are associated with decoding units in the access unit and apply to TargetOp are selected as follows:

– If the bitstream-specific CPB operation is tested, the following applies:

– If TargetDecLayerIdList contains all nuh\_layer\_id values present in the bitstream under test, non-nested buffering period, picture timing and decoding unit information SEI messages are selected.

– Otherwise, buffering period, picture timing and decoding unit information SEI messages included in the scalable nesting SEI message with bitstream\_subset\_flag equal to 1 and applicable to TargetOp are selected.

– Otherwise, buffering period, picture timing and decoding unit information SEI messages included in the bitstream partition nesting SEI message and applicable to the bitstream partition under test are selected.

1. A value of SchedSelIdx is selected as follows:

– If the bitstream-specific CPB operation is tested, the selected SchedSelIdx shall be in the range of 0 to cpb\_cnt\_minus1[ HighestTid ], inclusive, where cpb\_cnt\_minus1[ HighestTid ] is found in the sub\_layer\_hrd\_parameters( HighestTid ) syntax structure as selected above.

– Otherwise (the bitstream-partition-specific CPB operation is tested), a SchedSelCombIdx is selected for the bitstream under test and used for each bitstream partition under test. The following applies:

– If the vps\_vui\_bsp\_hrd\_parameters( ) syntax structure is present in the active VPS (or made available through external means not specified in this Specification) and contains the selected hrd\_parameters( ) syntax structure that applies to TargetOp and the bitstream partition under test, the selected SchedSelCombIdx shall be in the range of 0 to num\_bsp\_sched\_combinations[ TargetDecLayerSetIdx ] – 1, inclusive. and the selected SchedSelIdx shall be equal to bsp\_comb\_sched\_idx[ TargetDecLayerSetIdx ][ SchedSelCombIdx ][ j ] where j is the index of the bitstream partition under test.

– Otherwise, the selected SchedSelCombIdx shall be in the range of 0 to sei\_num\_bsp\_sched\_combinations\_minus1[ TargetDecLayerSetIdx ], inclusive. and the selected SchedSelIdx shall be equal to sei\_bsp\_comb\_sched\_idx[ TargetDecLayerSetIdx ][ SchedSelCombIdx ][ j ] of the bitstream partition HRD parameters SEI message applicable to TargetOp where j is the index of the bitstream partition under test.

1. When the coded picture in access unit 0 has nal\_unit\_type equal to CRA\_NUT or BLA\_W\_LP, and irap\_cpb\_params\_present\_flag in the selected buffering period SEI message is equal to 1, either of the following applies for selection of the initial CPB removal delay and delay offset:

– If NalHrdModeFlag is equal to 1, the default initial CPB removal delay and delay offset represented by nal\_initial\_cpb\_removal\_delay[ SchedSelIdx ] and nal\_initial\_cpb\_removal\_offset[ SchedSelIdx ], respectively, in the selected buffering period SEI message are selected. Otherwise, the default initial CPB removal delay and delay offset represented by vcl\_initial\_cpb\_removal\_delay[ SchedSelIdx ] and vcl\_initial\_cpb\_removal\_offset[ SchedSelIdx ], respectively, in the selected buffering period SEI message are selected. The variable DefaultInitCpbParamsFlag is set equal to 1.

– If NalHrdModeFlag is equal to 1, the alternative initial CPB removal delay and delay offset represented by nal\_initial\_alt\_cpb\_removal\_delay[ SchedSelIdx ] and nal\_initial\_alt\_cpb\_removal\_offset[ SchedSelIdx ], respectively, in the selected buffering period SEI message are selected. Otherwise, the alternative initial CPB removal delay and delay offset represented by vcl\_initial\_alt\_cpb\_removal\_delay[ SchedSelIdx ] and vcl\_initial\_alt\_cpb\_removal\_offset[ SchedSelIdx ], respectively, in the selected buffering period SEI message are selected. The variable DefaultInitCpbParamsFlag is set equal to 0, and the RASL access units associated with access unit 0 are discarded from BitstreamToDecode and the remaining bitstream is assigned to BitstreamToDecode.

1. For the bitstream-partition-specific CPB operation, SubPicHrdFlag is set equal to 1. For the bitstream-specific CPB operation, when sub\_pic\_hrd\_params\_present\_flag in the selected hrd\_parameters( ) syntax structure is equal to 1, the CPB is scheduled to operate either at the access unit level (in which case the variable SubPicHrdFlag is set equal to 0) or at the sub-picture level (in which case the variable SubPicHrdFlag is set equal to 1).

For each operation point under test when the bitstream-specific CPB operation is tested, the number of bitstream conformance tests to be performed is equal to n0 \* n1 \* ( n2 \* 2 + n3 ) \* n4, where the values of n0, n1, n2, n3, and n4 are specified as follows:

– n0 is derived as follows:

– If BitstreamToDecode is a Type I bitstream, n0 is equal to 1.

– Otherwise (BitstreamToDecode is a Type II bitstream), n0 is equal to 2.

– n1 is equal to cpb\_cnt\_minus1[ HighestTid ] + 1.

– n2 is the number of access units in BitstreamToDecode that each is associated with a buffering period SEI message applicable to TargetOp and for each of which both of the following conditions are true:

– nal\_unit\_type is equal to CRA\_NUT or BLA\_W\_LP for the VCL NAL units;

– The associated buffering period SEI message applicable to TargetOp has irap\_cpb\_params\_present\_flag equal to 1.

– n3 is the number of access units in BitstreamToDecode BitstreamToDecode that each is associated with a buffering period SEI message applicable to TargetOp and for each of which one or both of the following conditions are true:

– nal\_unit\_type is equal to neither CRA\_NUT nor BLA\_W\_LP for the VCL NAL units;

– The associated buffering period SEI message applicable to TargetOp has irap\_cpb\_params\_present\_flag equal to 0.

– n4 is derived as follows:

– If sub\_pic\_hrd\_params\_present\_flag in the selected hrd\_parameters( ) syntax structure is equal to 0, n4 is equal to 1;

– Otherwise, n4 is equal to 2.

When BitstreamToDecode is a Type II bitstream, the following applies:

– If the sub\_layer\_hrd\_parameters( HighestTid ) syntax structure that immediately follows the condition "if( vcl\_hrd\_parameters\_present\_flag )" is selected, the test is conducted at the Type I conformance point shown in Figure C‑1, and only VCL and filler data NAL units are counted for the input bit rate and CPB storage.

– Otherwise (the sub\_layer\_hrd\_parameters( HighestTid ) syntax structure that immediately follows the condition "if( nal\_hrd\_parameters\_present\_flag )" is selected), the test is conducted at the Type II conformance point shown in Figure C‑1, and all bytes of the Type II bitstream, which may be a NAL unit stream or a byte stream, are counted for the input bit rate and CPB storage.

NOTE 1 – NAL HRD parameters established by a value of SchedSelIdx for the Type II conformance point shown in Figure C‑1 are sufficient to also establish VCL HRD conformance for the Type I conformance point shown in Figure C‑1 for the same values of InitCpbRemovalDelay[ SchedSelIdx ], BitRate[ SchedSelIdx ], and CpbSize[ SchedSelIdx ] for the VBR case (cbr\_flag[ SchedSelIdx ] equal to 0). This is because the data flow into the Type I conformance point is a subset of the data flow into the Type II conformance point and because, for the VBR case, the CPB is allowed to become empty and stay empty until the time a next picture is scheduled to begin to arrive. For example, when decoding a CVS conforming to one or more of the profiles specified in Annex A using the decoding process specified in clauses 2 through 10, when NAL HRD parameters are provided for the Type II conformance point that not only fall within the bounds set for NAL HRD parameters for profile conformance in item f) of subclause A.4.2 but also fall within the bounds set for VCL HRD parameters for profile conformance in item e) of subclause A.4.2, conformance of the VCL HRD for the Type I conformance point is also assured to fall within the bounds of item e) of subclause A.4.2.

All VPSs, SPSs and PPSs referred to in the VCL NAL units, and the corresponding buffering period, picture timing and decoding unit information SEI messages shall be conveyed to the HRD, in a timely manner, either in the bitstream (by non-VCL NAL units), or by other means not specified in this Specification.

In Annexes C, D, and E, the specification for "presence" of non-VCL NAL units that contain VPSs, SPSs, PPSs, buffering period SEI messages, picture timing SEI messages, or decoding unit information SEI messages is also satisfied when those NAL units (or just some of them) are conveyed to decoders (or to the HRD) by other means not specified in this Specification. For the purpose of counting bits, only the appropriate bits that are actually present in the bitstream are counted.

NOTE 2 – As an example, synchronization of such a non-VCL NAL unit, conveyed by means other than presence in the bitstream, with the NAL units that are present in the bitstream, can be achieved by indicating two points in the bitstream, between which the non‑VCL NAL unit would have been present in the bitstream, had the encoder decided to convey it in the bitstream.

When the content of such a non-VCL NAL unit is conveyed for the application by some means other than presence within the bitstream, the representation of the content of the non-VCL NAL unit is not required to use the same syntax as specified in this Specification.

NOTE 3 – When HRD information is contained within the bitstream, it is possible to verify the conformance of a bitstream to the requirements of this subclause based solely on information contained in the bitstream. When the HRD information is not present in the bitstream, as is the case for all "stand-alone" Type I bitstreams, conformance can only be verified when the HRD data are supplied by some other means not specified in this Specification.

For the bitstream-specific CPB operation, the HRD contains a coded picture buffer (CPB), an instantaneous decoding process, a decoded picture buffer (DPB) that contains a sub-DPB for each layer, and output cropping as shown in Figure C‑2.



**Figure C‑2 – Bitstream-specific HRD buffer model**

For the bitstream-partition-specific CPB operation, the HRD contains a bitstream demultiplexer (optionally present), two or more bitstream partition buffers (BPB), two or more instantaneous decoding processes, a decoded picture buffer (DPB) that contains a sub-DPB for each layer, and output cropping as shown in Figure C‑3.



**Figure C‑3 – Bitstream-partition-specific HRD buffer model**

For each bitstream conformance test, the CPB size (number of bits) for the bitstream-specific CPB operation and the BPB size for the bitstream-partition-specific CPB operation is CpbSize[ SchedSelIdx ] as specified in subclause E.3.3, where SchedSelIdx and the HRD parameters are specified above in this subclause. When a CVS conforming to one or more of the profiles specified in Annex A is decoded by applying the decoding process specified in clauses 2−10, the sub-DPB size (number of picture storage buffers) of the sub-DPB for the base layer is sps\_max\_dec\_pic\_buffering\_minus1[ HighestTid ] + 1, where sps\_max\_dec\_pic\_buffering\_minus1[ HighestTid ] is from the active SPS for the base layer. When a CVS conforming to one or more of the profiles specified in Annex G or H is decoded by applying the decoding process specified in clauses 2−10, Annex F, and Annex G or H, the sub-DPB size of the sub-DPB for a layer with nuh\_layer\_id equal to currLayerId is max\_vps\_dec\_pic\_buffering\_minus1[ TargetOptLayerSetIdx ][ currLayerId ][ HighestTid ] + 1, where max\_vps\_dec\_pic\_buffering\_minus1[ TargetOptLayerSetIdx ][ currLayerId ][ HighestTid ] + 1 is from the active VPS.

The variable SubPicHrdPreferredFlag is either specified by external means, or when not specified by external means, set equal to 0.

When the value of the variable SubPicHrdFlag has not been set by step 9 above in this subclause, it is derived as follows:

SubPicHrdFlag = SubPicHrdPreferredFlag && sub\_pic\_hrd\_params\_present\_flag (C‑1)

If SubPicHrdFlag is equal to 0, the HRD operates at access unit level and each decoding unit is an access unit. Otherwise the HRD operates at sub-picture level and each decoding unit is a subset of an access unit.

NOTE 4 – If the HRD operates at access unit level, each time a decoding unit that is an entire access unit is removed from the CPB. Otherwise (the HRD operates at sub-picture level), each time a decoding unit that is a subset of an access unit is removed from the CPB. In both cases, each time an entire decoded picture is output from the DPB, though the picture output time is derived based on the differently derived CPB removal times and the differently signalled DPB output delays.

The following is specified for expressing the constraints in this annex:

– Each access unit is referred to as access unit n, where the number n identifies the particular access unit. Access unit 0 is selected per step 5 above. The value of n is incremented by 1 for each subsequent access unit in decoding order.

– Each decoding unit is referred to as decoding unit m, where the number m identifies the particular decoding unit. The first decoding unit in decoding order in access unit 0 is referred to as decoding unit 0. The value of m is incremented by 1 for each subsequent decoding unit in decoding order.

NOTE 5 – The numbering of decoding units is relative to the first decoding unit in access unit 0.

– Picture n refers to the coded picture or the decoded picture of access unit n.

The HRD operates as follows:

– The HRD is initialized at decoding unit 0, with the CPB, each sub-DPB of the DPB, and each BPB being set to be empty (the sub-DPB fullness for each sub-DPB is set equal to 0).

NOTE 6 – After initialization, the HRD is not initialized again by subsequent buffering period SEI messages.

– For the bitstream-specific CPB operation, data associated with decoding units that flow into the CPB according to a specified arrival schedule are delivered by the HSS. For the bitstream-partition-specific CPB operation, data associated with decoding units that flow into the BPB according to a specified arrival schedule are delivered by an HBPS.

– When the bitstream-partition-specific CPB operation is used, each bitstream partition with index j is processed as specified in clause C.2 with the HSS replaced by the HPBS and with SchedSelIdx equal to bsp\_comb\_sched\_idx[ TargetDecLayerSetIdx ][ SchedSelCombIdx ][ j ], if vps\_vui\_bsp\_hrd\_parameters( ) syntax structure is present in the active VPS or is available through some external means not specified in this Specification), or equal to sei\_bsp\_comb\_sched\_idx[ TargetDecLayerSetIdx ][ SchedSelCombIdx ][ j ] of the bitstream partition HRD parameters SEI message applicable to TargetOp, otherwise.

– The data associated with each decoding unit are removed and decoded instantaneously by the instantaneous decoding process at the CPB removal time of the decoding unit.

– Each decoded picture is placed in the DPB.

– A decoded picture is removed from the DPB when it becomes no longer needed for inter prediction reference and no longer needed for output.

For each bitstream conformance test, the operation of the CPB and the BPB is specified in subclause C.2, the instantaneous decoder operation is specified in clauses 2 through 10, the operation of the DPB is specified in subclause C.3, and the output cropping is specified in subclause C.3.3 and subclause C.5.2.2.

HSS, HBPS and HRD information concerning the number of enumerated delivery schedules and their associated bit rates and buffer sizes is specified in subclauses E.2.2 and E.3.2. The HRD is initialized as specified by the buffering period SEI message specified in subclauses D.2.2 and D.3.2. The removal timing of decoding units from the CPB and output timing of decoded pictures from the DPB is specified using information in picture timing SEI messages (specified in subclauses D.2.3 and D.3.3) or in decoding unit information SEI messages (specified in subclauses D.2.21 and D.3.21). All timing information relating to a specific decoding unit shall arrive prior to the CPB removal time of the decoding unit.

The requirements for bitstream conformance are specified in subclause C.4, and the HRD is used to check conformance of bitstreams as specified above in this subclause and to check conformance of decoders as specified in subclause 11.

NOTE 7 – While conformance is guaranteed under the assumption that all picture-rates and clocks used to generate the bitstream match exactly the values signalled in the bitstream, in a real system each of these may vary from the signalled or specified value.

All the arithmetic in this annex is performed with real values, so that no rounding errors can propagate. For example, the number of bits in a CPB just prior to or after removal of a decoding unit is not necessarily an integer.

The variable ClockTick is derived as follows and is called a clock tick:

ClockTick = vui\_num\_units\_in\_tick  vui\_time\_scale (C‑2)

The variable ClockSubTick is derived as follows and is called a clock sub-tick:

ClockSubTick = ClockTick  ( tick\_divisor\_minus2 + 2 ) (C‑3)

* 1. **Operation of coded picture buffer (CPB) and bitstream partition buffer (BPB)**
     1. **General**

The specifications in this subclause apply independently to each set of CPB parameters that is present and to both the Type I and Type II conformance points shown in Figure C‑1, and the set of CPB parameters is selected as specified in subclause C.1.

* + 1. **Timing of decoding unit arrival**

If SubPicHrdFlag is equal to 0, the variable subPicParamsFlag is set equal to 0, and the process in specified in the remainder of this subclause is invoked with a decoding unit being considered as an access unit, for derivation of the initial and final CPB arrival times for access unit n.

Otherwise (SubPicHrdFlag is equal to 1), the process in specified in the remainder of this subclause is first invoked with the variable subPicParamsFlag set equal to 0 and a decoding unit being considered as an access unit, for derivation of the initial and final CPB arrival times for access unit n, and then invoked with subPicParamsFlag set equal to 1 and a decoding unit being considered as a subset of an access unit, for derivation of the initial and final CPB arrival times for the decoding units in access unit n.

The variables InitCpbRemovalDelay[ SchedSelIdx ] and InitCpbRemovalDelayOffset[ SchedSelIdx ] are derived as follows:

– If one or more of the following conditions are true, InitCpbRemovalDelay[ SchedSelIdx ] and InitCpbRemovalDelayOffset[ SchedSelIdx ] are set equal to the values of the buffering period SEI message syntax elements nal\_initial\_alt\_cpb\_removal\_delay[ SchedSelIdx ] and nal\_initial\_alt\_cpb\_removal\_offset[ SchedSelIdx ], respectively, when NalHrdModeFlag is equal to 1, or vcl\_initial\_alt\_cpb\_removal\_delay[ SchedSelIdx ] and vcl\_initial\_alt\_cpb\_removal\_offset[ SchedSelIdx ], respectively, when NalHrdModeFlag is equal to 0, where the buffering period SEI message syntax elements are selected as specified in subclause C.1:

– Access unit 0 is a BLA access unit for which each coded picture has nal\_unit\_type equal to BLA\_W\_RADL or BLA\_N\_LP, and the value of irap\_cpb\_params\_present\_flag of the buffering period SEI message is equal to 1.

– Access unit 0 is a BLA access unit for which each coded picture has nal\_unit\_type equal to BLA\_W\_LP or is a CRA access unit, and the value of irap\_cpb\_params\_present\_flag of the buffering period SEI message is equal to 1, and one or more of the following conditions are true:

– UseAltCpbParamsFlag for access unit 0 is equal to 1.

– DefaultInitCpbParamsFlag is equal to 0.

– The value of subPicParamsFlag is equal to 1.

– Otherwise, InitCpbRemovalDelay[ SchedSelIdx ] and InitCpbRemovalDelayOffset[ SchedSelIdx ] are set equal to the values of the buffering period SEI message syntax elements nal\_initial\_cpb\_removal\_delay[ SchedSelIdx ] and nal\_initial\_cpb\_removal\_offset[ SchedSelIdx ], respectively, when NalHrdModeFlag is equal to 1, or vcl\_initial\_cpb\_removal\_delay[ SchedSelIdx ] and vcl\_initial\_cpb\_removal\_offset[ SchedSelIdx ], respectively, when NalHrdModeFlag is equal to 0, where the buffering period SEI message syntax elements are selected as specified in subclause C.1.

The time at which the first bit of decoding unit m begins to enter the CPB is referred to as the initial arrival time initArrivalTime[ m ].

If the bitstream-specific CPB operation is used, decoding units are indexed in decoding order within the bitstream. Otherwise (the bitstream-partition-specific CPB operation is used), decoding units are indexed in decoding order with each bitstream partition.

The initial arrival time of decoding unit m is derived as follows:

– If the decoding unit is decoding unit 0 (i.e. m = 0) and either the bitstream-specific CPB operation is used or the decoding unit belongs to the base bitstream partition, initArrivalTime[ 0 ] = 0.

– Otherwise, if the decoding unit is decoding unit 0, the bitstream-partition-specific CPB operation is used, and the decoding unit does not belong to the base bitstream partition, initArrivalTime[ 0 ] is obtained from the bitstream partition initial arrival time SEI message applicable to TargetOp.

– Otherwise, the following applies:

– If cbr\_flag[ SchedSelIdx ] is equal to 1, the initial arrival time for decoding unit m is equal to the final arrival time (which is derived below) of decoding unit m − 1, i.e.

if( !subPicParamsFlag )  
 initArrivalTime[ m ] = AuFinalArrivalTime[ m − 1 ] (C‑4)  
else  
 initArrivalTime[ m ] = DuFinalArrivalTime[ m − 1 ]

– Otherwise (cbr\_flag[ SchedSelIdx ] is equal to 0), the initial arrival time for decoding unit m is derived as follows:

if( !subPicParamsFlag )  
 initArrivalTime[ m ] = Max( AuFinalArrivalTime[ m − 1 ], initArrivalEarliestTime[ m ] ) (C‑5)  
else  
 initArrivalTime[ m ] = Max( DuFinalArrivalTime[ m − 1 ], initArrivalEarliestTime[ m ] )

where initArrivalEarliestTime[ m ] is derived as follows:

– The variable tmpNominalRemovalTime is derived as follows:

if( !subPicParamsFlag )  
 tmpNominalRemovalTime = AuNominalRemovalTime[ m ] (C‑6)  
else  
 tmpNominalRemovalTime = DuNominalRemovalTime[ m ]

where AuNominalRemovalTime[ m ] and DuNominalRemovalTime[ m ] are the nominal CPB removal time of access unit m and decoding unit m, respectively, as specified in subclause C.2.3.

– If decoding unit m is not the first decoding unit of a subsequent buffering period, initArrivalEarliestTime[ m ] is derived as follows:

initArrivalEarliestTime[ m ] = tmpNominalRemovalTime − ( InitCpbRemovalDelay[ SchedSelIdx ]  
 + InitCpbRemovalDelayOffset[ SchedSelIdx ] )  90000 (C‑7)

– Otherwise (decoding unit m is the first decoding unit of a subsequent buffering period), initArrivalEarliestTime[ m ] is derived as follows:

initArrivalEarliestTime[ m ] = tmpNominalRemovalTime −  
 ( InitCpbRemovalDelay[ SchedSelIdx ]  90000 ) (C‑8)

The final arrival time for decoding unit m is derived as follows:

if( !subPicParamsFlag )  
 AuFinalArrivalTime[ m ] = initArrivalTime[ m ] + sizeInbits[ m ]  BitRate[ SchedSelIdx ] (C‑9)  
else  
 DuFinalArrivalTime[ m ] = initArrivalTime[ m ] + sizeInbits[ m ]  BitRate[ SchedSelIdx ]

where sizeInbits[ m ] is the size in bits of decoding unit m, counting the bits of the VCL NAL units and the filler data NAL units for the Type I conformance point or all bits of the Type II bitstream for the Type II conformance point, where the Type I and Type II conformance points are as shown in Figure C‑1.

The values of SchedSelIdx, BitRate[ SchedSelIdx ], and CpbSize[ SchedSelIdx ] are constrained as follows:

– If the content of the selected hrd\_parameters( ) syntax structures for the access unit containing decoding unit m and the previous access unit differ, the HSS selects a value SchedSelIdx1 of SchedSelIdx from among the values of SchedSelIdx provided in the selected hrd\_parameters( ) syntax structures for the access unit containing decoding unit m that results in a BitRate[ SchedSelIdx1 ] or CpbSize[ SchedSelIdx1 ] for the access unit containing decoding unit m. The value of BitRate[ SchedSelIdx1 ] or CpbSize[ SchedSelIdx1 ] may differ from the value of BitRate[ SchedSelIdx0 ] or CpbSize[ SchedSelIdx0 ] for the value SchedSelIdx0 of SchedSelIdx that was in use for the previous access unit.

– Otherwise, the HSS continues to operate with the previous values of SchedSelIdx, BitRate[ SchedSelIdx ] and CpbSize[ SchedSelIdx ].

When the HSS selects values of BitRate[ SchedSelIdx ] or CpbSize[ SchedSelIdx ] that differ from those of the previous access unit, the following applies:

– The variable BitRate[ SchedSelIdx ] comes into effect at the initial CPB arrival time of the current access unit.

– The variable CpbSize[ SchedSelIdx ] comes into effect as follows:

– If the new value of CpbSize[ SchedSelIdx ] is greater than the old CPB size, it comes into effect at the initial CPB arrival time of the current access unit.

– Otherwise, the new value of CpbSize[ SchedSelIdx ] comes into effect at the CPB removal time of the current access unit.

* + 1. **Timing of decoding unit removal and decoding of decoding unit**

The variables InitCpbRemovalDelay[ SchedSelIdx ], InitCpbRemovalDelayOffset[ SchedSelIdx ], CpbDelayOffset, and DpbDelayOffset are derived as follows:

– If one or more of the following conditions are true, CpbDelayOffset is set equal to the value of the buffering period SEI message syntax element cpb\_delay\_offset, DpbDelayOffset is set equal to the value of the buffering period SEI message syntax element dpb\_delay\_offset, and InitCpbRemovalDelay[ SchedSelIdx ] and InitCpbRemovalDelayOffset[ SchedSelIdx ] are set equal to the values of the buffering period SEI message syntax elements nal\_initial\_alt\_cpb\_removal\_delay[ SchedSelIdx ] and nal\_initial\_alt\_cpb\_removal\_offset[ SchedSelIdx ], respectively, when NalHrdModeFlag is equal to 1, or vcl\_initial\_alt\_cpb\_removal\_delay[ SchedSelIdx ] and vcl\_initial\_alt\_cpb\_removal\_offset[ SchedSelIdx ], respectively, when NalHrdModeFlag is equal to 0, where the buffering period SEI message containing the syntax elements is selected as specified in subclause C.1:

– Access unit 0 is a BLA access unit for which each coded picture has nal\_unit\_type equal to BLA\_W\_RADL or BLA\_N\_LP, and the value of irap\_cpb\_params\_present\_flag of the buffering period SEI message is equal to 1.

– Access unit 0 is a BLA access unit for which each coded picture has nal\_unit\_type equal to BLA\_W\_LP or is a CRA access unit, and the value of irap\_cpb\_params\_present\_flag of the buffering period SEI message is equal to 1, and one or more of the following conditions are true:

– UseAltCpbParamsFlag for access unit 0 is equal to 1.

– DefaultInitCpbParamsFlag is equal to 0.

– Otherwise, InitCpbRemovalDelay[ SchedSelIdx ] and InitCpbRemovalDelayOffset[ SchedSelIdx ] are set equal to the values of the buffering period SEI message syntax elements nal\_initial\_cpb\_removal\_delay[ SchedSelIdx ] and nal\_initial\_cpb\_removal\_offset[ SchedSelIdx ], respectively, when NalHrdModeFlag is equal to 1, or vcl\_initial\_cpb\_removal\_delay[ SchedSelIdx ] and vcl\_initial\_cpb\_removal\_offset[ SchedSelIdx ], respectively, when NalHrdModeFlag is equal to 0, where the buffering period SEI message containing the syntax elements is selected as specified in subclause C.1, CpbDelayOffset and DpbDelayOffset are both set equal to 0.

The nominal removal time of the access unit n from the CPB is specified as follows:

– If access unit n is the access unit with n equal to 0 (the access unit that initializes the HRD), the nominal removal time of the access unit from the CPB is specified by:

AuNominalRemovalTime[ 0 ] = InitCpbRemovalDelay[ SchedSelIdx ]  90000 (C‑10)

– Otherwise, the following applies:

– When access unit n is the first access unit of a buffering period that does not initialize the HRD, the following applies:

The nominal removal time of the access unit n from the CPB is specified by:

if( !concatenationFlag ) {  
 baseTime = AuNominalRemovalTime[ firstPicInPrevBuffPeriod ]  
 tmpCpbRemovalDelay = AuCpbRemovalDelayVal  
} else {  
 baseTime = AuNominalRemovalTime[ prevNonDiscardablePic ]  
 tmpCpbRemovalDelay =  
 Max( ( auCpbRemovalDelayDeltaMinus1 + 1 ), (C‑11)  
 Ceil( ( InitCpbRemovalDelay[ SchedSelIdx ]  90000 +  
 AuFinalArrivalTime[ n − 1 ] − AuNominalRemovalTime[ n − 1 ] )  ClockTick ) )  
}  
AuNominalRemovalTime[ n ] = baseTime + ClockTick \* ( tmpCpbRemovalDelay − CpbDelayOffset )

where AuNominalRemovalTime[ firstPicInPrevBuffPeriod ] is the nominal removal time of the first access unit of the previous buffering period, AuNominalRemovalTime[ prevNonDiscardablePic ] is the nominal removal time of the preceding access unit in decoding order, each picture of which is with TemporalId equal to 0 that is not a RASL, RADL or sub-layer non-reference picture, AuCpbRemovalDelayVal is the value of AuCpbRemovalDelayVal derived according to au\_cpb\_removal\_delay\_minus1 in the picture timing SEI message, selected as specified in subclause C.1, associated with access unit n, and concatenationFlag and auCpbRemovalDelayDeltaMinus1 are the values of the syntax elements concatenation\_flag and au\_cpb\_removal\_delay\_delta\_minus1, respectively, in the buffering period SEI message, selected as specified in subclause C.1, associated with access unit n.

After the derivation of the nominal CPB removal time and before the derivation of the DPB output time of access unit n, the values of CpbDelayOffset and DpbDelayOffset are updated as follows:

– If one or more of the following conditions are true, CpbDelayOffset is set equal to the value of the buffering period SEI message syntax element cpb\_delay\_offset, and DpbDelayOffset is set equal to the value of the buffering period SEI message syntax element dpb\_delay\_offset, where the buffering period SEI message containing the syntax elements is selected as specified in subclause C.1:

– Access unit n is a BLA access unit for which each coded picture has nal\_unit\_type equal to BLA\_W\_RADL or BLA\_N\_LP, and the value of irap\_cpb\_params\_present\_flag of the buffering period SEI message is equal to 1.

– Access unit n is a BLA access unit for which each coded picture has nal\_unit\_type equal to BLA\_W\_LP or is a CRA access unit, and the value of irap\_cpb\_params\_present\_flag of the buffering period SEI message is equal to 1, and UseAltCpbParamsFlag for access unit n is equal to 1.

– Otherwise, CpbDelayOffset and DpbDelayOffset are both set equal to 0.

– When access unit n is not the first access unit of a buffering period, the nominal removal time of the access unit n from the CPB is specified by:

AuNominalRemovalTime[ n ] = AuNominalRemovalTime[ firstPicInCurrBuffPeriod ] +  
 ClockTick \* ( AuCpbRemovalDelayVal − CpbDelayOffset ) (C‑12)

where AuNominalRemovalTime[ firstPicInCurrBuffPeriod ] is the nominal removal time of the first access unit of the current buffering period, and AuCpbRemovalDelayVal is the value of AuCpbRemovalDelayVal derived according to au\_cpb\_removal\_delay\_minus1 in the picture timing SEI message, selected as specified in subclause C.1, associated with access unit n.

When SubPicHrdFlag is equal to 1, the following applies:

– The variable duCpbRemovalDelayInc is derived as follows:

– If sub\_pic\_cpb\_params\_in\_pic\_timing\_sei\_flag is equal to 0, duCpbRemovalDelayInc is set equal to the value of du\_spt\_cpb\_removal\_delay\_increment in the decoding unit information SEI message, selected as specified in subclause C.1, associated with decoding unit m.

– Otherwise, if du\_common\_cpb\_removal\_delay\_flag is equal to 0, duCpbRemovalDelayInc is set equal to the value of du\_cpb\_removal\_delay\_increment\_minus1[ i ] + 1 for decoding unit m in the picture timing SEI message, selected as specified in subclause C.1, associated with access unit n, where the value of i is 0 for the first num\_nalus\_in\_du\_minus1[ 0 ] + 1 consecutive NAL units in the access unit that contains decoding unit m, 1 for the subsequent num\_nalus\_in\_du\_minus1[ 1 ] + 1 NAL units in the same access unit, 2 for the subsequent num\_nalus\_in\_du\_minus1[ 2 ] + 1 NAL units in the same access unit, etc.

– Otherwise, duCpbRemovalDelayInc is set equal to the value of du\_common\_cpb\_removal\_delay\_increment\_minus1 + 1 in the picture timing SEI message, selected as specified in subclause C.1, associated with access unit n.

– The nominal removal time of decoding unit m from the CPB is specified as follows, where AuNominalRemovalTime[ n ] is the nominal removal time of access unit n:

– If decoding unit m is the last decoding unit in access unit n, the nominal removal time of decoding unit m DuNominalRemovalTime[ m ] is set equal to AuNominalRemovalTime[ n ].

– Otherwise (decoding unit m is not the last decoding unit in access unit n), the nominal removal time of decoding unit m DuNominalRemovalTime[ m ] is derived as follows:

if( sub\_pic\_cpb\_params\_in\_pic\_timing\_sei\_flag )  
 DuNominalRemovalTime[ m ] = DuNominalRemovalTime[ m + 1 ] −  
 ClockSubTick \* duCpbRemovalDelayInc (C‑13)  
else  
 DuNominalRemovalTime[ m ] = AuNominalRemovalTime[ n ] −  
 ClockSubTick \* duCpbRemovalDelayInc

If SubPicHrdFlag is equal to 0, the removal time of access unit n from the CPB is specified as follows, where AuFinalArrivalTime[ n ] and AuNominalRemovalTime[ n ] are the final CPB arrival time and nominal CPB removal time, respectively, of access unit n:

if( !low\_delay\_hrd\_flag[ HighestTid ] | | AuNominalRemovalTime[ n ] >= AuFinalArrivalTime[ n ] )  
 AuCpbRemovalTime[ n ] = AuNominalRemovalTime[ n ]  
else (C‑14)  
 AuCpbRemovalTime[ n ] = AuNominalRemovalTime[ n ] + ClockTick \*  
 Ceil( ( AuFinalArrivalTime[ n ] − AuNominalRemovalTime[ n ] )  ClockTick )

NOTE 1 – When low\_delay\_hrd\_flag[ HighestTid ] is equal to 1 and AuNominalRemovalTime[ n ] is less than AuFinalArrivalTime[ n ], the size of access unit n is so large that it prevents removal at the nominal removal time.

Otherwise (SubPicHrdFlag is equal to 1), the removal time of decoding unit m from the CPB is specified as follows:

– When the bitstream-specific CPB operation is used or when the current DU belongs to the base bitstream partition, the following applies:

if( !low\_delay\_hrd\_flag[ HighestTid ] | | DuNominalRemovalTime[ m ] >= DuFinalArrivalTime[ m ] )  
 DuCpbRemovalTime[ m ] = DuNominalRemovalTime[ m ]  
else (C‑15)  
 DuCpbRemovalTime[ m ] = DuFinalArrivalTime[ m ]

NOTE 2 – When low\_delay\_hrd\_flag[ HighestTid ] is equal to 1 and DuNominalRemovalTime[ m ] is less than DuFinalArrivalTime[ m ], the size of decoding unit m is so large that it prevents removal at the nominal removal time.

– When the bitstream-partition-specific CPB operation is used and cbr\_flag[ SchedSelIdx ] is equal to 0, the following applies:

– Let refDuCpbRemovalTime be equal to the CPB removal time of the previous DU preceding the current DU in decoding order (regardless of the bitstream partitions to which the previous DU and the current DU belong).

– The variable DuCpbRemovalTime[ m ] is modified as follows:

DuCpbRemovalTime[ m ] = Max( DuCpbRemovalTime[ m ], refDuCpbRemovalTime ) (C‑16)

If SubPicHrdFlag is equal to 0, at the CPB removal time of access unit n, the access unit is instantaneously decoded.

Otherwise (SubPicHrdFlag is equal to 1), at the CPB removal time of decoding unit m, the decoding unit is instantaneously decoded, and when decoding unit m is the last decoding unit of access unit n, the following applies:

– Access unit n is considered as decoded.

– The final CPB arrival time of access unit n, i.e. AuFinalArrivalTime[ n ], is set equal to the final CPB arrival time of the last decoding unit in access unit n, i.e. DuFinalArrivalTime[ m ].

– The nominal CPB removal time of access unit n, i.e. AuNominalRemovalTime[ n ], is set equal to the nominal CPB removal time of the last decoding unit in access unit n, i.e. DuNominalRemovalTime[ m ].

– The CPB removal time of access unit n, i.e. AuCpbRemovalTime[ m ], is set equal to the CPB removal time of the last decoding unit in access unit n, i.e. DuCpbRemovalTime[ m ].

* 1. **Operation of the decoded picture buffer (DPB)**
     1. **General**

The specifications in this subclause apply independently to each set of DPB parameters selected as specified in subclause C.1.

The decoded picture buffer consists of sub-DPBs, and each sub-DPB contains picture storage buffers for storage of decoded pictures of one layer only. Each of the picture storage buffers of a sub-DPB may contain a decoded picture that is marked as "used for reference" or is held for future output.

The following applies for all decoded access units:

– If an access unit does not contain a picture at a target output layer and alt\_output\_layer\_flag is equal to 1, the following ordered steps apply:

– The list nonOutputLayerPictures is the list of pictures of the access unit with PicOutputFlag equal to 1 and with nuh\_layer\_id values that are included in the TargetDecLayerIdList and that are not on target output layers.

– The picture with the highest nuh\_layer\_id value among the list nonOutputLayerPictures is removed from the list nonOutputLayerPictures.

– PicOutputFlag for each picture that is included in the list nonOutputLayerPictures is set equal to 0.

– Otherwise, PicOutputFlag for pictures that are not included in a target output layer is set equal to 0.

The processes specified in subclauses C.3.2, C.3.3 and C.3.4 are sequentially applied as specified below, and are applied independently for each layer, starting from the base layer, in increasing order of nuh\_layer\_id values of the layers in the bitstream. When these processes are applied for a particular layer, only the sub-DPB for the particular layer is affected. In the descriptions of these processes, the DPB refers to the sub-DPB for the particular layer, and the particular layer is referred to as the current layer.

NOTE – In the operation of output timing DPB, decoded pictures with PicOutputFlag equal to 1 in the same access unit are output consecutively in ascending order of the nuh\_layer\_id values of the decoded pictures.

Let picture n and the current picture be the coded picture or decoded picture of the access unit n for a particular value of nuh\_layer\_id, wherein n is a non-negative integer number. [Ed. (CY&YK): This probably is not a good definition of picture n especially if each picture is a DU. It is a temporary term defined only for DPB operations, further improvements are needed.]

* + 1. **Removal of pictures from the DPB**

When the current picture is not picture 0 in the current layer, the removal of pictures in the current layer from the DPB before decoding of the current picture, i.e. picture n, but after parsing the slice header of the first slice of the current picture, happens instantaneously at the CPB removal time of the first decoding unit of the current picture and proceeds as follows:

– The decoding process for RPS as specified in subclause 8.3.2 is invoked.

– When the current picture is an IRAP picture with NoRaslOutputFlag equal to 1, or the base layer picture in the current access unit is an IRAP picture with NoRaslOutputFlag equal to 1 and NoClrasOutputFlag is equal to 1, the following ordered steps are applied:

1. The variable NoOutputOfPriorPicsFlag is derived for the decoder under test as follows:

– If the current picture is a CRA picture with NoRaslOutputFlag equal to 1, NoOutputOfPriorPicsFlag is set equal to 1 (regardless of the value of no\_output\_of\_prior\_pics\_flag).

– Otherwise, if the current picture is an IRAP picture with NoRaslOutputFlag equal to 1 and the value of pic\_width\_in\_luma\_samples, pic\_height\_in\_luma\_samples, or sps\_max\_dec\_pic\_buffering\_minus1[ HighestTid ] derived from the active SPS for the current layer is different from the value of pic\_width\_in\_luma\_samples, pic\_height\_in\_luma\_samples, or sps\_max\_dec\_pic\_buffering\_minus1[ HighestTid ], respectively, derived from the SPS that was active for the current layer when decoding the preceding picture in the current layer, NoOutputOfPriorPicsFlag may (but should not) be set to 1 by the decoder under test, regardless of the value of no\_output\_of\_prior\_pics\_flag.

NOTE – Although setting NoOutputOfPriorPicsFlag equal to no\_output\_of\_prior\_pics\_flag is preferred under these conditions, the decoder under test is allowed to set NoOutputOfPriorPicsFlag to 1 in this case.

– Otherwise, if the current picture is an IRAP picture with NoRaslOutputFlag equal to 1, NoOutputOfPriorPicsFlag is set equal to no\_output\_of\_prior\_pics\_flag.

– Otherwise (the current picture is not an IRAP picture with NoRaslOutputFlag equal to 1, the base layer picture in the current access unit is an IRAP picture with NoRaslOutputFlag equal to 1, and NoClrasOutputFlag is equal to 1), NoOutputOfPriorPicsFlag is set equal to 1.

2. The value of NoOutputOfPriorPicsFlag derived for the decoder under test is applied for the HRD, such that when the value of NoOutputOfPriorPicsFlag is equal to 1, all picture storage buffers in the DPB are emptied without output of the pictures they contain, and the DPB fullness is set equal to 0.

– When both of the following conditions are true for any pictures k in the DPB, all such pictures k in the DPB are removed from the DPB:

– picture k is marked as "unused for reference"

– picture k has PicOutputFlag equal to 0 or its DPB output time is less than or equal to the CPB removal time of the first decoding unit (denoted as decoding unit m) of the current picture n; i.e. DpbOutputTime[ k ] is less than or equal to CpbRemovalTime( m )

– For each picture that is removed from the DPB, the DPB fullness is decremented by one.

* + 1. **Picture output**

The processes specified in this subclause happen instantaneously at the CPB removal time of access unit n, AuCpbRemovalTime[ n ].

When picture n has PicOutputFlag equal to 1, its DPB output time DpbOutputTime[ n ] is derived as follows, where the variable firstPicInBufferingPeriodFlag is equal to 1 if access unit n is the first access unit of a buffering period and 0 otherwise:

if( !SubPicHrdFlag ) {  
 DpbOutputTime[ n ] = AuCpbRemovalTime[ n ] + ClockTick \* picDpbOutputDelay (C‑17)  
 if( firstPicInBufferingPeriodFlag )  
 DpbOutputTime[ n ] −= ClockTick \* DpbDelayOffset  
} else  
 DpbOutputTime[ n ] = AuCpbRemovalTime[ n ] + ClockSubTick \* picSptDpbOutputDuDelay

where picDpbOutputDelay is the value of pic\_dpb\_output\_delay in the picture timing SEI message associated with access unit n, and picSptDpbOutputDuDelay is the value of pic\_spt\_dpb\_output\_du\_delay, when present, in the decoding unit information SEI messages associated with access unit n, or the value of pic\_dpb\_output\_du\_delay in the picture timing SEI message associated with access unit n when there is no decoding unit information SEI message associated with access unit n or no decoding unit information SEI message associated with access unit n has pic\_spt\_dpb\_output\_du\_delay present.

NOTE – When the syntax element pic\_spt\_dpb\_output\_du\_delay is not present in any decoding unit information SEI message associated with access unit n, the value is inferred to be equal to pic\_dpb\_output\_du\_delay in the picture timing SEI message associated with access unit n.

The output of the current picture is specified as follows:

– If PicOutputFlag is equal to 1 and DpbOutputTime[ n ] is equal to AuCpbRemovalTime[ n ], the current picture is output.

– Otherwise, if PicOutputFlag is equal to 0, the current picture is not output, but will be stored in the DPB as specified in subclause C.3.4.

– Otherwise (PicOutputFlag is equal to 1 and DpbOutputTime[ n ] is greater than AuCpbRemovalTime[ n ] ), the current picture is output later and will be stored in the DPB (as specified in subclause C.3.4) and is output at time DpbOutputTime[ n ] unless indicated not to be output by the decoding or inference of no\_output\_of\_prior\_pics\_flag equal to 1 at a time that precedes DpbOutputTime[ n ].

When output, the picture is cropped, using the conformance cropping window specified in the active SPS for the layer containing the picture.

When picture n is a picture that is output and is not the last picture of the bitstream that is output, the value of the variable DpbOutputInterval[ n ] is derived as follows:

DpbOutputInterval[ n ] = DpbOutputTime[ nextPicInOutputOrder ] − DpbOutputTime[ n ] (C‑18)

where nextPicInOutputOrder is the picture that follows picture n in output order and has PicOutputFlag equal to 1.

* + 1. **Current decoded picture marking and storage**

The process specified in this subclause happens instantaneously at the CPB removal time of the last decoding unit of the current picture. [Ed. (MH): This change might not comply with version 1, because version 1 decoders would mark and store the base-layer picture at the CPB removal time of the AU, which can be later than the CPB removal time of the base-layer picture.]

The current decoded picture is stored in the DPB in an empty picture storage buffer, the DPB fullness is incremented by one, and the current picture is marked as "used for short-term reference".

* 1. **Bitstream conformance**

A bitstream of coded data conforming to this Specification shall fulfil all requirements specified in this subclause.

The bitstream shall be constructed according to the syntax, semantics, and constraints specified in this Specification outside of this annex.

The first access unit in a bitstream shall be an IRAP access unit.

The bitstream is tested by the HRD for conformance as specified in subclause C.1.

Let currPicLayerId be equal to the nuh\_layer\_id of the current picture.

For each current picture, let the variables maxPicOrderCnt and minPicOrderCnt be set equal to the maximum and the minimum, respectively, of the PicOrderCntVal values of the following pictures with nuh\_layer\_id equal to currPicLayerId:

– The current picture.

– The previous picture in decoding order that has TemporalId equal to 0 and that is not a RASL picture, a RADL picture, or a sub-layer non-reference picture.

– The short-term reference pictures in the RPS of the current picture.

– All pictures n that have PicOutputFlag equal to 1, AuCpbRemovalTime[ n ] less than AuCpbRemovalTime[ currPic ], and DpbOutputTime[ n ] greater than or equal to AuCpbRemovalTime[ currPic ], where currPic is the current picture. [Ed. (CY): clarify the AuCpbRemovalTime of a picture to be that of the containing AU.]

All of the following conditions shall be fulfilled for each of the bitstream conformance tests:

1. For each access unit n, with n greater than 0, associated with a buffering period SEI message, let the variable deltaTime90k[ n ] be specified as follows:

deltaTime90k[ n ] = 90000 \* ( AuNominalRemovalTime[ n ] − AuFinalArrivalTime[ n − 1 ] ) (C‑19)

The value of InitCpbRemovalDelay[ SchedSelIdx ] is constrained as follows:

– If cbr\_flag[ SchedSelIdx ] is equal to 0, the following condition shall be true:

InitCpbRemovalDelay[ SchedSelIdx ] <= Ceil( deltaTime90k[ n ] ) (C‑20)

– Otherwise (cbr\_flag[ SchedSelIdx ] is equal to 1), the following condition shall be true:

Floor( deltaTime90k[ n ] ) <= InitCpbRemovalDelay[ SchedSelIdx ] <= Ceil( deltaTime90k[ n ] ) (C‑21)

NOTE 1 – The exact number of bits in the CPB at the removal time of each picture may depend on which buffering period SEI message is selected to initialize the HRD. Encoders must take this into account to ensure that all specified constraints must be obeyed regardless of which buffering period SEI message is selected to initialize the HRD, as the HRD may be initialized at any one of the buffering period SEI messages.

1. A CPB overflow is specified as the condition in which the total number of bits in the CPB is greater than the CPB size. The CPB shall never overflow.
2. A CPB underflow is specified as the condition in which the nominal CPB removal time of decoding unit m DuNominalRemovalTime( m ) is less than the final CPB arrival time of decoding unit m DuFinalArrivalTime( m ) for at least one value of m. When low\_delay\_hrd\_flag[ HighestTid ] is equal to 0, the CPB shall never underflow.
3. When SubPicHrdFlag is equal to 1, low\_delay\_hrd\_flag[ HighestTid ] is equal to 1, and the nominal removal time of a decoding unit m of access unit n is less than the final CPB arrival time of decoding unit m (i.e. DuNominalRemovalTime[ m ] < DuFinalArrivalTime[ m ]), the nominal removal time of access unit n shall be less than the final CPB arrival time of access unit n (i.e. AuNominalRemovalTime[ n ] < AuFinalArrivalTime[ n ]).
4. When the bitstream-partition-specific CPB operation is used and cbr\_flag[ SchedSelIdx ] is equal to 1, DuCpbRemovalTime[ m ] shall be greater than or equal to the CPB removal time of the previous DU preceding the current DU in decoding order (regardless of the bitstream partitions to which the previous DU and the current DU belong) for any decoding unit m in bitstream partitions with index greater than 0.
5. The nominal removal times of access units from the CPB (starting from the second access unit in decoding order) shall satisfy the constraints on AuNominalRemovalTime[ n ] and AuCpbRemovalTime[ n ] expressed in subclauses A.4.1 through A.4.2.
6. For each current picture, after invocation of the process for removal of pictures from the sub-DPB as specified in subclause C.3.2, the number of decoded pictures in the sub-DPB for the current layer, including all pictures n in the current layer that are marked as "used for reference", or that have PicOutputFlag equal to 1 and AuCpbRemovalTime[ n ] less than AuCpbRemovalTime[ currPic ], where currPic is the current picture, shall be less than or equal to sps\_max\_dec\_pic\_buffering\_minus1[ HighestTid ] when currPicLayerId is equal to 0 or max\_vps\_dec\_pic\_buffering\_minus1[ TargetOptLayerSetIdx ][ currPicLayerId ][ HighestTid ] when currPicLayerId is greater than 0.
7. All reference pictures shall be present in the DPB when needed for prediction. Each picture that has PicOutputFlag equal to 1 shall be present in the DPB at its DPB output time unless it is removed from the DPB before its output time by one of the processes specified in subclause C.3.
8. For each current picture, the value of maxPicOrderCnt − minPicOrderCnt shall be less than MaxPicOrderCntLsb / 2.
9. The value of DpbOutputInterval[ n ] as given by Equation C‑18, which is the difference between the output time of an access unit and that of the first access unit following it in output order and having PicOutputFlag equal to 1, shall satisfy the constraint expressed in subclause A.4.1 for the profile, tier and level specified in the bitstream using the decoding process specified in clauses 2 through 10. [Ed. (MH): This constraint has to be updated, since 1) it assumes a single profile-tier-level combination for a bitstream (as if the bitstream were a single-layer bitstream), and 2) it refers to the decoding process in clauses 2 to 10 (while now also the decoding process of extensions should somehow be referred to).]
10. For each current picture, when sub\_pic\_cpb\_params\_in\_pic\_timing\_sei\_flag is equal to 1, let tmpCpbRemovalDelaySum be derived as follows:

tmpCpbRemovalDelaySum = 0  
for( i = 0; i < num\_decoding\_units\_minus1; i++ ) (C‑22)  
 tmpCpbRemovalDelaySum += du\_cpb\_removal\_delay\_increment\_minus1[ i ] + 1

The value of ClockSubTick \* tmpCpbRemovalDelaySum shall be equal to the difference between the nominal CPB removal time of the current access unit and the nominal CPB removal time of the first decoding unit in the current access unit in decoding order.

1. For any two pictures m and n in the same CVS, when DpbOutputTime[ m ] is greater than DpbOutputTime[ n ], the PicOrderCntVal of picture m shall be greater than the PicOrderCntVal of picture n.

NOTE 2 – All pictures of an earlier CVS in decoding order that are output are output before any pictures of a later CVS in decoding order. Within any particular CVS, the pictures that are output are output in increasing PicOrderCntVal order.

* 1. **Decoder conformance**
     1. **General**

A decoder conforming to this Specification shall fulfil all requirements specified in this subclause.

A decoder claiming conformance to a specific profile, tier and level shall be able to successfully decode all bitstreams that conform to the bitstream conformance requirements specified in subclause C.4, in the manner specified in Annex A, provided that all VPSs, SPSs and PPSs referred to in the VCL NAL units, and appropriate buffering period and picture timing SEI messages are conveyed to the decoder, in a timely manner, either in the bitstream (by non-VCL NAL units), or by external means not specified in this Specification.

When a bitstream contains syntax elements that have values that are specified as reserved and it is specified that decoders shall ignore values of the syntax elements or NAL units containing the syntax elements having the reserved values, and the bitstream is otherwise conforming to this Specification, a conforming decoder shall decode the bitstream in the same manner as it would decode a conforming bitstream and shall ignore the syntax elements or the NAL units containing the syntax elements having the reserved values as specified.

There are two types of conformance that can be claimed by a decoder: output timing conformance and output order conformance.

To check conformance of a decoder, test bitstreams conforming to the claimed profile, tier and level, as specified in subclause C.4 are delivered by a hypothetical stream scheduler (HSS) both to the HRD and to the decoder under test (DUT). All cropped decoded pictures output by the HRD shall also be output by the DUT, each cropped decoded picture output by the DUT shall be a picture with PicOutputFlag equal to 1, and, for each such cropped decoded picture output by the DUT, the values of all samples that are output shall be equal to the values of the samples produced by the specified decoding process.

For output timing decoder conformance, the HSS operates as described above, with delivery schedules selected only from the subset of values of SchedSelIdx for which the bit rate and CPB size are restricted as specified in Annex A for the specified profile, tier and level, or with "interpolated" delivery schedules as specified below for which the bit rate and CPB size are restricted as specified in Annex A. The same delivery schedule is used for both the HRD and the DUT.

When the HRD parameters and the buffering period SEI messages are present with cpb\_cnt\_minus1[ HighestTid ] greater than 0, the decoder shall be capable of decoding the bitstream as delivered from the HSS operating using an "interpolated" delivery schedule specified as having peak bit rate r, CPB size c( r ), and initial CPB removal delay ( f( r )r ) as follows:

 = ( r − BitRate[ SchedSelIdx − 1 ] )  ( BitRate[ SchedSelIdx ] − BitRate[ SchedSelIdx − 1 ] ), (C‑23)

c( r ) =  \* CpbSize[ SchedSelIdx ] + ( 1 −   \* CpbSize[ SchedSelIdx − 1 ], (C‑24)

f( r ) = InitCpbRemovalDelay[ SchedSelIdx ] \* BitRate[ SchedSelIdx ] +   
 ( 1 −  InitCpbRemovalDelay[ SchedSelIdx − 1 ] \* BitRate[ SchedSelIdx − 1 ] (C‑25)

for any SchedSelIdx > 0 and r such that BitRate[ SchedSelIdx − 1 ] <= r <= BitRate[ SchedSelIdx ] such that r and c( r ) are within the limits as specified in Annex A for the maximum bit rate and buffer size for the specified profile, tier and level.

NOTE 1 – InitCpbRemovalDelay[ SchedSelIdx ] can be different from one buffering period to another and have to be re-calculated.

For output timing decoder conformance, an HRD as described above is used and the timing (relative to the delivery time of the first bit) of picture output is the same for both the HRD and the DUT up to a fixed delay.

For output order decoder conformance, the following applies:

– The HSS delivers the bitstream BitstreamToDecode to the DUT "by demand" from the DUT, meaning that the HSS delivers bits (in decoding order) only when the DUT requires more bits to proceed with its processing.

NOTE 2 – This means that for this test, the coded picture buffer of the DUT could be as small as the size of the largest decoding unit.

– A modified HRD as described below is used, and the HSS delivers the bitstream to the HRD by one of the schedules specified in the bitstream BitstreamToDecode such that the bit rate and CPB size are restricted as specified in Annex A. The order of pictures output shall be the same for both the HRD and the DUT.

– The HRD CPB size is given by CpbSize[ SchedSelIdx ] as specified in subclause E.3.3, where SchedSelIdx and the HRD parameters are selected as specified in subclause C.1. The DPB size is given by sps\_max\_dec\_pic\_buffering\_minus1[ HighestTid ] + 1. Removal time from the CPB for the HRD is the final bit arrival time and decoding is immediate. The operation of the DPB of this HRD is as described in subclauses C.5.2 through C.5.2.3.

* + 1. **Operation of the output order DPB**
       1. **General**

The decoded picture buffer consists of sub-DPBs, and each sub-DPB contains picture storage buffers for storage of decoded pictures of one layer only. Each of the picture storage buffers of a sub-DPB contains a decoded picture that is marked as "used for reference" or is held for future output.

The process for output and removal of pictures from the DPB as specified in subclause C.5.2.2 is invoked, followed by the invocation of the process for picture decoding, marking, additional bumping, and storage as specified in subclause C.5.2.3. The "bumping" process is specified in subclause C.5.2.4 and is invoked as specified in subclauses C.5.2.2 and C.5.2.3.

These processes are applied independently for each layer, starting from the base layer, in increasing order of the nuh\_layer\_id values of the layers in the bitstream. When these processes are applied for a particular layer, only the sub-DPB for the particular layer is affected except for the "bumping" process, which may crop and output pictures, mark pictures as "not needed for output" and empty picture storage buffers for any layer.

NOTE – In the operation of output order DPB, same as in the operation of output timing DPB, decoded pictures with PicOutputFlag equal to 1 in the same access unit are also output consecutively in ascending order of the nuh\_layer\_id values of the decoded pictures.

Let picture n and the current picture be the coded picture or decoded picture of the access unit n for a particular value of nuh\_layer\_id, wherein n is a non-negative integer number.

When these processes are applied for a layer with nuh\_layer\_id equal to currLayerId, the variables MaxNumReorderPics, MaxLatencyIncreasePlus1, MaxLatencyPictures, and MaxDecPicBufferingMinus1 are derived as follows:

– If a CVS conforming to one or more of the profiles specified in Annex G or H is decoded by applying the decoding process specified in clauses 2−10, Annex F, and Annex G or H, the following applies:

* MaxNumReorderPics is set equal to max\_vps\_num\_reorder\_pics[ TargetOptLayerSetIdx ][ HighestTid ] of the active VPS.
* MaxLatencyIncreasePlus1 is set equal to the value of the syntax element max\_vps\_latency\_increase\_plus1[ TargetOptLayerSetIdx ][ HighestTid ] of the active VPS,.
* MaxLatencyPictures is set equal to VpsMaxLatencyPictures[ TargetOptLayerSetIdx ][ HighestTid ] of the active VPS.
* MaxDecPicBufferingMinus1 is set equal to the value of the syntax element max\_vps\_dec\_pic\_buffering\_minus1[ TargetOptLayerSetIdx ][ currLayerId ][ HighestTid ] of the active VPS.

– Otherwise (a CVS conforming to one or more of the profiles specified in Annex A is decoded by applying the decoding process specified in clauses 2−10), the following applies:

* MaxNumReorderPics is set equal to sps\_max\_num\_reorder\_pics[ HighestTid ] of the active SPS for the base layer.
* MaxLatencyIncreasePlus1 is set equal to sps\_max\_latency\_increase\_plus1[ HighestTid ] of the active SPS for the base layer.
* MaxLatencyPictures is set equal to SpsMaxLatencyPictures[ HighestTid ] of the active SPS for the base layer.
* MaxDecPicBufferingMinus1 is set equal to sps\_max\_dec\_pic\_buffering\_minus1[ HighestTid ] of the active SPS for the base layer.
  + - 1. **Output and removal of pictures from the DPB**

When the current picture is not picture 0 in the current layer, the output and removal of pictures in the current layer from the DPB before the decoding of the current picture , i.e. picture n, but after parsing the slice header of the first slice of the current picture, happens instantaneously when the first decoding unit of the current picture is removed from the CPB and proceeds as follows:

– The decoding process for RPS as specified in subclause 8.3.2 is invoked.

– If the current picture is an IRAP picture with NoRaslOutputFlag equal to 1, or the base layer picture in the current access unit is an IRAP picture with NoRaslOutputFlag equal to 1 and NoClrasOutputFlag is equal to 1, the following ordered steps are applied:

1. The variable NoOutputOfPriorPicsFlag is derived for the decoder under test as follows:

– If the current picture is a CRA picture with NoRaslOutputFlag equal to 1, NoOutputOfPriorPicsFlag is set equal to 1 (regardless of the value of no\_output\_of\_prior\_pics\_flag).

– Otherwise, if the current picture is an IRAP picture with NoRaslOutputFlag equal to 1 and the value of pic\_width\_in\_luma\_samples, pic\_height\_in\_luma\_samples, or sps\_max\_dec\_pic\_buffering\_minus1[ HighestTid ] derived from the active SPS for the current layer is different from the value of pic\_width\_in\_luma\_samples, pic\_height\_in\_luma\_samples, or sps\_max\_dec\_pic\_buffering\_minus1[ HighestTid ], respectively, derived from the SPS that was active for the current layer when decoding the preceding picture in the current layer, NoOutputOfPriorPicsFlag may (but should not) be set to 1 by the decoder under test, regardless of the value of no\_output\_of\_prior\_pics\_flag.

NOTE – Although setting NoOutputOfPriorPicsFlag equal to no\_output\_of\_prior\_pics\_flag is preferred under these conditions, the decoder under test is allowed to set NoOutputOfPriorPicsFlag to 1 in this case.

– Otherwise, if the current picture is an IRAP picture with NoRaslOutputFlag equal to 1, NoOutputOfPriorPicsFlag is set equal to no\_output\_of\_prior\_pics\_flag.

– Otherwise (the current picture is not an IRAP picture with NoRaslOutputFlag equal to 1, the base layer picture in the current access unit is an IRAP picture with NoRaslOutputFlag equal to 1, and NoClrasOutputFlag is equal to 1), NoOutputOfPriorPicsFlag is set equal to 1.

2. The value of NoOutputOfPriorPicsFlag derived for the decoder under test is applied for the HRD as follows:

– If NoOutputOfPriorPicsFlag is equal to 1, all picture storage buffers in the sub-DPB are emptied without output of the pictures they contain, and the sub-DPB fullness is set equal to 0.

– Otherwise (NoOutputOfPriorPicsFlag is equal to 0), all picture storage buffers containing a picture that is marked as "not needed for output" and "unused for reference" are emptied (without output), and all non-empty picture storage buffers in the sub-DPB are emptied by repeatedly invoking the "bumping" process specified in subclause C.5.2.4, and the sub-DPB fullness is set equal to 0.

– Otherwise, all picture storage buffers that contain a picture in the current layer and that are marked as "not needed for output" and "unused for reference" are emptied (without output). For each picture storage buffer that is emptied, the sub-DPB fullness is decremented by one. When one or more of the following conditions are true, the "bumping" process specified in subclause C.5.2.4 is invoked repeatedly while further decrementing the sub-DPB fullness by one for each additional picture storage buffer that is emptied, until none of the following conditions are true:

* The number of access units that contain at least one decoded picture in the DPB marked as "needed for output" is greater than MaxNumReorderPics.
* MaxLatencyIncreasePlus1 is not equal to 0 and there is at least one access units that contain at least one decoded picture in the DPB marked as "needed for output" for which the associated variable PicLatencyCount is greater than or equal to MaxLatencyPictures.
* The number of pictures in the current layer in the sub-DPB is greater than or equal to MaxDecPicBufferingMinus1 + 1.
  + - 1. **Picture decoding, marking, additional bumping, and storage**

The processes specified in this subclause happen instantaneously when the last decoding unit of picture n is removed from the CPB. [Ed. (MH): This change might not comply with version 1, because version 1 decoders would mark and store the base-layer picture at the CPB removal time of the AU, which can be later than the CPB removal time of the base-layer picture.]

PicOutputFlag is updated as follows:

– If the current access unit does not contain a picture at a target output layer and alt\_output\_layer\_flag is equal to 1, the following ordered steps apply:

– The list nonOutputLayerPictures is the list of pictures of the access unit with PicOutputFlag equal to 1 and with nuh\_layer\_id values that are included in the TargetDecLayerIdList and that are not on target output layers.

– The picture with the highest nuh\_layer\_id value among the list nonOutputLayerPictures is removed from the list nonOutputLayerPictures.

– PicOutputFlag for each picture that is included in the list nonOutputLayerPictures is set equal to 0.

– Otherwise, PicOutputFlag for pictures that are not included in a target output layer is set equal to 0.

When the current picture has PicOutputFlag equal to 1, for each picture in the current layer in the sub-DPB that is marked as "needed for output" and follows the current picture in output order, the associated variable PicLatencyCount is set equal to PicLatencyCount + 1.

The current picture is considered as decoded after the last decoding unit of the picture is decoded. The current decoded picture is stored in an empty picture storage buffer in the sub-DPB, and the following applies:

– If the current decoded picture has PicOutputFlag equal to 1, it is marked as "needed for output" and its associated variable PicLatencyCount is set equal to 0.

– Otherwise (the current decoded picture has PicOutputFlag equal to 0), it is marked as "not needed for output".

The current decoded picture is marked as "used for short-term reference".

When one or more of the following conditions are true, the "bumping" process specified in subclause C.5.2.4 is invoked repeatedly until none of the following conditions are true:

– The number of access units that contain at least one decoded picture in the DPB marked as "needed for output" is greater than MaxNumReorderPics.

– MaxLatencyIncreasePlus1 is not equal to 0 and there is at least one access units that contain at least one decoded picture in the DPB marked as "needed for output" for which the associated variable PicLatencyCount is greater than or equal to MaxLatencyPictures.

* + - 1. **"Bumping" process**

The "bumping" process consists of the following ordered steps:

1. The picture or pictures that are first for output are selected as the ones having the smallest value of PicOrderCntVal of all pictures in the DPB marked as "needed for output".
2. Each of these pictures is, in ascending nuh\_layer\_id order, cropped, using the conformance cropping window specified in the active SPS for the picture, the cropped picture is output, and the picture is marked as "not needed for output".
3. Each picture storage buffer that contains a picture marked as "unused for reference" and that was one of the pictures cropped and output is emptied.
   1. **Demultiplexing process for deriving a bitstream partition**

Inputs to this process are a bitstream, a layer identifier list bspLayerId[ bspIdx ] and the number of layer identifiers numBspLayerId in the layer index list bspLayerId[ bspIdx ].

Output of this process is a bitstream partition.

Let variable minBspLayerId be the smallest value of bspLayerId[ bspIdx ] with any value of bspIdx in the range of 0 to numBspLayerId – 1, inclusive.

The output bitstream partition consists of selected NAL units of the input bitstream in the same order as they appear in the input bitstream. The following NAL units of the input bitstream are omitted from the output bitstream partition, while the remaining NAL units of the input bitstream are included in the output bitstream partition:

– Omit all NAL units that have a nuh\_layer\_id value other than bspLayerId[ bspIdx ] with any value of bspIdx in the range of 0 to numBspLayerId – 1, inclusive.

– Omit all SEI NAL units containing a scalable nesting SEI message for which no derived nestingLayerIdList[ i ] contains any layer identifier value equal to bspLayerId[ bspIdx ] with any value of bspIdx in the range of 0 to numBspLayerId – 1, inclusive.

– Omit all SEI NAL units containing a scalable nesting SEI message for which a derived nestingLayerIdList[ i ] contains a layer identifier value less than minBspLayerId.

*Modify subclause D.1.1 as follows:*

1. Annex D  
     
   Supplemental enhancement information

(This annex forms an integral part of this Recommendation | International Standard)

* 1. SEI payload syntax
     1. General SEI message syntax

*Add rows enclosed by "...".*

|  |  |
| --- | --- |
| sei\_payload( payloadType, payloadSize ) { | Descriptor |
| if( nal\_unit\_type = = PREFIX\_SEI\_NUT ) |  |
| if( payloadType = = 0 ) |  |
| **...** |  |
| else if( payloadType = = XXX ) |  |
| layers\_not\_present( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| inter\_layer\_constrained\_tile\_sets( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| bsp\_nesting( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| bsp\_initial\_arrival\_time( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| bsp\_hrd( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| three\_dimensional\_reference\_displays\_info( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| depth\_representation\_info\_sei( payloadSize ) |  |
| **...** |  |
| else |  |
| reserved\_sei\_message( payloadSize ) |  |
| else /\* nal\_unit\_type = = SUFFIX\_SEI\_NUT \*/ |  |
| if( payloadType = = 3 ) |  |
| filler\_payload( payloadSize ) |  |
| **...** |  |
| else |  |
| reserved\_sei\_message( payloadSize ) |  |
| if( more\_data\_in\_payload( ) ) { |  |
| if( payload\_extension\_present( ) ) |  |
| **reserved\_payload\_extension\_data** | u(v) |
| **payload\_bit\_equal\_to\_one** /\* equal to 1 \*/ | f(1) |
| while( !byte\_aligned( ) ) |  |
| **payload\_bit\_equal\_to\_zero** /\* equal to 0 \*/ | f(1) |
| } |  |
| } |  |

*Modify subclause E.2.1 as follows:*

1. Annex E  
     
   Video usability information

(This annex forms an integral part of this Recommendation | International Standard)

* 1. VUI semantics
     1. VUI parameters semantics

The specifications in clause E.2.1 apply with the following modifications and additions.

**video\_signal\_type\_present\_flag** equal to 1 specifies that video\_format, video\_full\_range\_flag and colour\_description\_present\_flag are present. video\_signal\_type\_present\_flag equal to 0, specifies that video\_format, video\_full\_range\_flag and colour\_description\_present\_flag are not present. It is a requirement of bitstream conformance that, when nuh\_layer\_id is greater than 0, video\_signal\_type\_present\_flag shall be equal to 0.

When a current picture with nuh\_layer\_id layerIdCurr greater than 0 refers to an SPS containing the VUI parameter syntax structure, the values of video\_format, video\_full\_range\_flag, colour\_primaries, transfer\_characteristics, and matrix\_coeffs are inferred as follows:

– If the nuh\_layer\_id of the active SPS for the layer with nuh\_layer\_id equal to layerIdCurr is equal to 0, the values of video\_format, video\_full\_range\_flag, colour\_primaries, transfer\_characteristics, and matrix\_coeffs are inferred to be equal to video\_vps\_format, video\_full\_range\_vps\_flag, colour\_primaries\_vps, transfer\_characteristics\_vps, and matrix\_coeffs\_vps, respectively, of the vps\_video\_signal\_info\_idx[ j ]-th video\_signal\_info( ) syntax structure in the active VPS where j is equal to LayerIdxInVps[ layerIdCurr ] and the values of video\_format, video\_full\_range\_flag, colour\_primaries, transfer\_characteristics, and matrix\_coeffs of the active SPS for the layer with nuh\_layer\_id equal to layerIdCurr are ignored.

NOTE X – The values are inferred from the VPS when a non-base layer refers to an SPS that is also referred to by the base layer, in which case the SPS has nuh\_layer\_id equal to 0. For the base layer, the values of these parameters in the active SPS for the base layer apply.

– Otherwise (the nuh\_layer\_id of the active SPS for the layer with nuh\_layer\_id equal to layerIdCurr is greater than zero), values of video\_format, video\_full\_range\_flag, colour\_primaries, transfer\_characteristics, and matrix\_coeffs are inferred to be equal to video\_vps\_format, video\_full\_range\_vps\_flag, colour\_primaries\_vps, transfer\_characteristics\_vps, and matrix\_coeffs\_vps, respectively, of the vps\_video\_signal\_info\_idx[ j ]-th video\_signal\_info( ) syntax structure in the active VPS, where j is equal to LayerIdxInVps[ layerIdCurr ].

[Ed. (GT) Consider shortening duplicated inference specification above. What should happen when VPS VUI is not present? ]

1. Annex F  
     
   Common syntax, semantics and decoding processes for multi-layer video coding extensions

(This annex forms an integral part of this Recommendation | International Standard)

This annex specifies the common syntax, semantics and decoding processes for multi-layer video coding extensions.

* 1. Scope

Common syntax, semantics and decoding processes for multi-layer video coding extensions are specified in this annex with reference made to clauses 2-9 and Annexes A-E and G.

* 1. Normative references

The specifications in clause 2 apply.

* 1. Definitions

For the purpose of this annex, the following definitions apply in addition to the definitions in clause 3. These definitions are either not present in clause 3 or replace definitions in clause 3.

[Ed. (YK&MH&CY): Definitions should be checked and potentially refined, including: BLA AU, IDR AU, CRA AU, output order, picture order count, RADL AU, RASL AU, (reference picture), STSA AU, TSA AU.]

1. access unit: A set of *NAL units* that are associated with each other according to a specified classification rule, are consecutive in *decoding order,* and contain the *VCL NAL units* of all *coded pictures* associated with the same output time and their associated non-VCL NAL units.

NOTE 1 – Pictures in the same access unit are associated with the same picture order count.

1. aligned inter-layer reference picture: An *inter-layer reference picture* included in the *access unit* that also contains the current *picture*.
2. associated IRAP picture: The previous *IRAP picture* in *decoding order* within the same *layer* (if present).
3. auxiliary picture: A *picture* that has no normative effect on the *decoding process* of *primary pictures*.
4. base layer: A *layer* in which all *VCL NAL units* have nuh\_layer\_id equal to 0.
5. coded picture: A *coded representation* of a *picture* comprising *VCL NAL units* with a particular value of nuh\_layer\_id within an *access unit* and containing all *coding tree units* of the *picture*.  
   [Ed. (CY): consider defining picture by associating nuh\_layer\_id. In HEVC base, picture is defined as arrays of luma and chroma samples, however, it is often associated with other properties, e.g., coding tree units. So to be absolutely precise, it might be clearer and applicable to define picture as follows: *picture*: An array of *luma* samples in monochrome format or an array of *luma* samples and two corresponding arrays of *chroma* samples in 4:2:0, 4:2:2, and 4:4:4 colour format with the same value of nuh\_layer\_id.]
6. coded video sequence (CVS): A sequence of *access units* that consists, in decoding order, of an *initial IRAP access unit*, followed by zero or more *access units* that are not *initial IRAP access units*, including all subsequent *access units* up to but not including any subsequent *access unit* that is an *initial IRAP access unit*.
7. collocated sample: A sample TBD. [ Ed. (GT) Maybe it is easier to define a collocated position and require collocated samples to have it? ]
8. diagonal inter-layer reference picture: An *inter-layer reference picture* specified using a reference-layer short-term reference picture set.
9. direct reference layer: A *layer* that may be used for inter-layer prediction of another *layer*.
10. indirect reference layer: A *layer* that is not a *direct reference layer* of another *layer* but is a *direct reference layer* of a *layer* that is a *direct reference layer* or indirect reference *layer* of a *direct reference layer* of the *layer*.
11. initial intra random access point (IRAP) access unit: An *IRAP access unit* in which the *coded picture* with nuh\_layer\_id equal to 0 has NoRaslOutputFlag equal to 1.
12. inter-layer prediction: A *prediction* in a manner that is dependent on data elements (e.g. sample values or motion vectors) of *reference pictures* with a different value of nuh\_layer\_id than that of the current *picture.*
13. inter-layer reference picture: A *reference picture* that has a different nuh\_layer\_id value than that of the current *picture*.
14. inter-layer reference picture set: A set of *aligned inter-layer reference pictures* associated with the current *picture*.
15. intra random access point (IRAP) access unit: An *access unit* in which the *coded picture* with nuh\_layer\_id equal to 0 is an *IRAP picture*.
16. leading picture: A *picture* that is in the same *layer* as the *associated IRAP picture* and precedes the *associated IRAP picture* in *output order*.
17. non-base layer: A *layer* in which all *VCL NAL units* have the same nuh\_layer\_id value greater than 0.
18. picture order count: A variable that is associated with each *picture* and that uniquely identifies the associated *picture* among all *pictures* with the same value of nuh\_layer\_id in the *CVS*, and, when the associated *picture* is to be output from the *decoded picture buffer*, indicates the position of the associated *picture* in *output order* relative to the *output order* positions of the other *pictures* with the same value of nuh\_layer\_id in the same *CVS* that are to be output from the *decoded picture buffer*.
19. primary picture: a *picture* with a nuh\_layer\_id value such that AuxId[ nuh\_layer\_id ] is equal to 0*.*
20. reference layer picture: A *picture* in a *direct* *reference layer* which is used for inter-layer prediction of the current *picture*.
21. reference-layer short-term reference picture set: A *short-term reference picture set* that is specified for a *direct reference layer* to specify *diagonal inter-layer reference pictures*.
22. reference picture list: A list of reference pictures that is used for inter prediction or inter-layer prediction of a P or B slice.
23. target output layer: A *layer* that is to be output.
24. trailing picture: A *picture* that is in the same *layer* as the *associated IRAP picture* and follows the *associated IRAP picture* in *output order*.
25. output time: A time when a *decoded* *picture* is to be output as specified in Annex C, if the timing information is present in the *coded video sequence*. [Ed.: Consider adding this definition in clause 3 of the main specification containing both version 1 and Annex F specifications.]
26. view: A sequence of pictures associated with the same value of ViewOrderIdx.

NOTE 2 – A view typically represents a sequence of pictures captured by one camera.

* 1. Abbreviations

The specifications in clause 4 apply.

* 1. Conventions

The specifications in clause 5 apply.

* 1. Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships

The specifications in clause 6 apply.

* 1. Syntax and semantics

This clause specifies syntax and semantics for CVSs that conform to one or more of the profiles specified in this annex.

* + 1. Method of specifying syntax in tabular form

The specifications in subclause 7.1 apply.

* + 1. Specification of syntax functions, categories, and descriptors

The specifications in subclause 7.2 apply.

* + 1. Syntax in tabular form
       1. NAL unit syntax

The specifications in subclause 7.3.1 apply.

* + - * 1. General NAL unit syntax

The specifications in subclause 7.3.1.1 apply.

* + - * 1. NAL unit header syntax

The specifications in subclause 7.3.1.2 apply.

* + - 1. Raw byte sequence payloads and RBSP trailing bits syntax
         1. Video parameter set RBSP

|  |  |
| --- | --- |
| video\_parameter\_set\_rbsp( ) { | Descriptor |
| **vps\_video\_parameter\_set\_id** | u(4) |
| **vps\_reserved\_three\_2bits** | u(2) |
| **vps\_max\_layers\_minus1** | u(6) |
| **vps\_max\_sub\_layers\_minus1** | u(3) |
| **vps\_temporal\_id\_nesting\_flag** | u(1) |
| **vps\_extension\_offset** **//**vps\_reserved\_0xffff\_16bits | u(16) |
| profile\_tier\_level( 1, vps\_max\_sub\_layers\_minus1 ) |  |
| **vps\_sub\_layer\_ordering\_info\_present\_flag** | u(1) |
| for( i = ( vps\_sub\_layer\_ordering\_info\_present\_flag ? 0 : vps\_max\_sub\_layers\_minus1 );  i <= vps\_max\_sub\_layers\_minus1; i++ ) { |  |
| **vps\_max\_dec\_pic\_buffering\_minus1**[ i ] | ue(v) |
| **vps\_max\_num\_reorder\_pics**[ i ] | ue(v) |
| **vps\_max\_latency\_increase\_plus1**[ i ] | ue(v) |
| } |  |
| **vps\_max\_layer\_id** | u(6) |
| **vps\_num\_layer\_sets\_minus1** | ue(v) |
| for( i = 1; i <= vps\_num\_layer\_sets\_minus1; i++ ) |  |
| for( j = 0; j <= vps\_max\_layer\_id; j++ ) |  |
| **layer\_id\_included\_flag**[ i ][ j ] | u(1) |
| **vps\_timing\_info\_present\_flag** | u(1) |
| if( vps\_timing\_info\_present\_flag ) { |  |
| **vps\_num\_units\_in\_tick** | u(32) |
| **vps\_time\_scale** | u(32) |
| **vps\_poc\_proportional\_to\_timing\_flag** | u(1) |
| if( vps\_poc\_proportional\_to\_timing\_flag ) |  |
| **vps\_num\_ticks\_poc\_diff\_one\_minus1** | ue(v) |
| **vps\_num\_hrd\_parameters** | ue(v) |
| for( i = 0; i < vps\_num\_hrd\_parameters; i++ ) { |  |
| **hrd\_layer\_set\_idx**[ i ] | ue(v) |
| if( i > 0 ) |  |
| **cprms\_present\_flag**[ i ] | u(1) |
| hrd\_parameters( cprms\_present\_flag[ i ], vps\_max\_sub\_layers\_minus1 ) |  |
| } |  |
| } |  |
| **vps\_extension\_flag** | u(1) |
| if( vps\_extension\_flag ) { |  |
| while( !byte\_aligned( ) ) |  |
| **vps\_extension\_alignment\_bit\_equal\_to\_one** | u(1) |
| vps\_extension( ) |  |
| **vps\_extension2\_flag** | u(1) |
| if( vps\_extension2\_flag ) |  |
| while( more\_rbsp\_data( ) ) |  |
| **vps\_extension\_data\_flag** | u(1) |
| } |  |
| rbsp\_trailing\_bits( ) |  |
| } |  |

Video parameter set extension syntax

|  |  |
| --- | --- |
| vps\_extension( ) { | Descriptor |
| **avc\_base\_layer\_flag** | u(1) |
| **vps\_vui\_present\_flag** | u(1) |
| if( vps\_vui\_present\_flag ) |  |
| **vps\_vui\_offset** | u(16) |
| **splitting\_flag** | u(1) |
| for( i = 0, NumScalabilityTypes = 0; i < 16; i++ ) { |  |
| **scalability\_mask\_flag**[ i ] | u(1) |
| NumScalabilityTypes += scalability\_mask\_flag[ i ] |  |
| } |  |
| for( j = 0; j < ( NumScalabilityTypes − splitting\_flag ); j++ ) |  |
| **dimension\_id\_len\_minus1**[ j ] | u(3) |
| **vps\_nuh\_layer\_id\_present\_flag** | u(1) |
| for( i = 1; i <= MaxLayersMinus1; i++ ) { |  |
| if( vps\_nuh\_layer\_id\_present\_flag ) |  |
| **layer\_id\_in\_nuh**[ i ] | u(6) |
| if( !splitting\_flag ) |  |
| for( j = 0; j < NumScalabilityTypes; j++ ) |  |
| **dimension\_id**[ i ][ j ] | u(v) |
| } |  |
| **view\_id\_len** | u(4) |
| if( view\_id\_len > 0 ) |  |
| for( i = 0; i < NumViews; i++ ) |  |
| **view\_id\_val**[ i ] | u(v) |
| for( i = 1; i <= MaxLayersMinus1; i++ ) |  |
| for( j = 0; j < i; j++ ) |  |
| **direct\_dependency\_flag**[ i ][ j ] | u(1) |
| **vps\_sub\_layers\_max\_minus1\_present\_flag** | u(1) |
| if( vps\_sub\_layers\_ max\_minus1\_present\_flag ) |  |
| for( i = 0; i <= MaxLayersMinus1; i++ ) |  |
| sub\_layers\_vps\_max\_minus1[ i ] | u(3) |
| **max\_tid\_ref\_present\_flag** | u(1) |
| if( max\_tid\_ref\_present\_flag ) |  |
| for( i = 0; i < MaxLayersMinus1; i++ ) |  |
| for( j = i + 1; j <= MaxLayersMinus1; j++ ) |  |
| if( direct\_dependency\_flag[ j ][ i ] ) |  |
| **max\_tid\_il\_ref\_pics\_plus1**[ i ][ j ] | u(3) |
| **all\_ref\_layers\_active\_flag** | u(1) |
| **vps\_number\_layer\_sets\_minus1** | u(10) |
| **vps\_num\_profile\_tier\_level\_minus1** | u(6) |
| for( i = 1; i <= vps\_num\_profile\_tier\_level\_minus1; i ++ ) { |  |
| **vps\_profile\_present\_flag**[ i ] | u(1) |
| if( !vps\_profile\_present\_flag[ i ] ) |  |
| **profile\_ref\_minus1**[ i ] | u(6) |
| profile\_tier\_level( vps\_profile\_present\_flag[ i ], vps\_max\_sub\_layers\_minus1 ) |  |
| } |  |
| NumOutputLayerSets = vps\_number\_layer\_sets\_minus1 + 1 |  |
| **more\_output\_layer\_sets\_than\_default\_flag** | u(1) |
| if( more\_output\_layer\_sets\_than\_default\_flag ) { |  |
| **num\_add\_output\_layer\_sets\_minus1** | u(10) |
| numOutputLayerSets += num\_add\_output\_layer\_sets\_minus1 + 1 |  |
| } |  |
| if( numOutputLayerSets > 1 ) |  |
| **default\_one\_target\_output\_layer\_idc** | u(2) |
| for( i = 1; i < numOutputLayerSets; i++ ) { |  |
| if( i > vps\_number\_layer\_sets\_minus1 ) { |  |
| **output\_layer\_set\_idx\_minus1**[ i ] | u(v) |
| lsIdx = output\_layer\_set\_idx\_minus1[ i ] + 1 |  |
| for( j = 0 ; j < NumLayersInIdList[ lsIdx ] − 1; j++) |  |
| **output\_layer\_flag**[ i ][ j ] | u(1) |
| } |  |
| **profile\_level\_tier\_idx**[ i ] | u(v) |
| } |  |
| if( vps\_max\_layers\_minus1 > 0 ) |  |
| **alt\_output\_layer\_flag** | u(1) |
| **rep\_format\_idx\_present\_flag** | u(1) |
| if( rep\_format\_idx\_present\_flag ) |  |
| **vps\_num\_rep\_formats\_minus1** | u(8) |
| for( i = 0; i <= vps\_num\_rep\_formats\_minus1; i++ ) |  |
| rep\_format( ) |  |
| if( rep\_format\_idx\_present\_flag ) |  |
| for( i = 1; i <= MaxLayersMinus1; i++ ) |  |
| if( vps\_num\_rep\_formats\_minus1 > 0 ) |  |
| **vps\_rep\_format\_idx**[ i ] | u(8) |
| **max\_one\_active\_ref\_layer\_flag** | u(1) |
| for( i = 1; i <= MaxLayersMinus1; i++ ) |  |
| if( NumDirectRefLayers[ layer\_id\_in\_nuh[ i ] ] = = 0 ) |  |
| **poc\_lsb\_not\_present\_flag**[ i ] | u(1) |
| **cross\_layer\_phase\_alignment\_flag** | u(1) |
| dpb\_size( ) |  |
| **direct\_dep\_type\_len\_minus2** | ue(v) [Ed. (JB): Should this be ue(v)? ] |
| **default\_direct\_dependency\_flag** | u(1) |
| if( default\_direct\_dependency\_flag ) |  |
| **default\_direct\_dependency\_type** | u(v) |
| else { |  |
| for( i = 1; i <= MaxLayersMinus1; i++ ) |  |
| for( j = 0; j < i; j++ ) |  |
| if( direct\_dependency\_flag[ i ][ j ] ) |  |
| **direct\_dependency\_type**[ i ][ j ] | u(v) |
| } |  |
| if( vps\_vui\_present\_flag ) { |  |
| while( !byte\_aligned( ) ) |  |
| **vps\_vui\_alignment\_bit\_equal\_to\_one** | u(1) |
| vps\_vui( ) |  |
| } |  |
| } |  |

Representation format syntax

[Ed. (YK): The syntax and semantics for rep\_format( ), dpb\_size( ), and vps\_vui( ) should probably have one-level-higher section titles, similarly as profile\_tier\_level( ).]

|  |  |
| --- | --- |
| rep\_format( ) { | Descriptor |
| **pic\_width\_vps\_in\_luma\_samples** | u(16) |
| **pic\_height\_vps\_in\_luma\_samples** | u(16) |
| **chroma\_and\_bit\_depth\_vps\_present\_flag** | u(1) |
| if( chroma\_and\_bit\_depth\_vps\_present\_flag ) { |  |
| **chroma\_format\_vps\_idc** | u(2) |
| if( chroma\_format\_vps\_idc = = 3 ) |  |
| **separate\_colour\_plane\_vps\_flag** | u(1) |
| **bit\_depth\_vps\_luma\_minus8** | u(4) |
| **bit\_depth\_vps\_chroma\_minus8** | u(4) |
| } |  |
| } |  |

DPB size syntax

|  |  |
| --- | --- |
| dpb\_size( ) { |  |
| for( i = 1; i < NumOutputLayerSets; i++ ) { |  |
| **sub\_layer\_flag\_info\_present\_flag**[ i ] | u(1) |
| for( j = 0; j <= vps\_max\_sub\_layers\_minus1; j++ ) { |  |
| if( j > 0 && sub\_layer\_flag\_info\_present\_flag[ i ] ) |  |
| **sub\_layer\_dpb\_info\_present\_flag**[ i ][ j ] | u(1) |
| if( sub\_layer\_dpb\_info\_present\_flag[ i ][ j ] ) { |  |
| for( k = 0; k < NumSubDpbs[ i ]; k++ ) |  |
| **max\_vps\_dec\_pic\_buffering\_minus1**[ i ][ k ][ j ] | ue(v) |
| **max\_vps\_num\_reorder\_pics**[ i ][ j ] | ue(v) |
| **max\_vps\_latency\_increase\_plus1**[ i ][ j ] | ue(v) |
| } |  |
| } |  |
| } |  |
| } |  |

VPS VUI syntax

|  |  |
| --- | --- |
| vps\_vui( ){ | Descriptor |
| **cross\_layer\_pic\_type\_aligned\_flag** | u(1) |
| if( !cross\_layer\_pic\_type\_aligned\_flag ) |  |
| **cross\_layer\_irap\_aligned\_flag** | u(1) |
| **bit\_rate\_present\_vps\_flag** | u(1) |
| **pic\_rate\_present\_vps\_flag** | u(1) |
| if( bit\_rate\_present\_vps\_flag | | pic\_rate\_present\_vps\_flag ) |  |
| for( i = 0; i <= vps\_number\_layer\_sets\_minus1; i++ ) |  |
| for( j = 0; j <= vps\_max\_sub\_layers\_minus1; j++ ) { |  |
| if( bit\_rate\_present\_vps\_flag ) |  |
| **bit\_rate\_present\_flag**[ i ][ j ] | u(1) |
| if( pic\_rate\_present\_vps\_flag ) |  |
| **pic\_rate\_present\_flag**[ i ][ j ] | u(1) |
| if( bit\_rate\_present\_flag[ i ][ j ] ) { |  |
| **avg\_bit\_rate**[ i ][ j ] | u(16) |
| **max\_bit\_rate**[ i ][ j ] | u(16) |
| } |  |
| if( pic\_rate\_present\_flag[ i ][ j ] ) { |  |
| **constant\_pic\_rate\_idc**[ i ][ j ] | u(2) |
| **avg\_pic\_rate**[ i ][ j ] | u(16) |
| } |  |
| } |  |
| **tiles\_not\_in\_use\_flag** | u(1) |
| if( !tiles\_not\_in\_use\_flag ) { |  |
| for( i = 0; i <= MaxLayersMinus1; i++ ) { | u(1) |
| **tiles\_in\_use\_flag**[ i ] |  |
| if( tiles\_in\_use\_flag[ i ] ) |  |
| **loop\_filter\_not\_across\_tiles\_flag**[ i ] | u(1) |
| } |  |
| for( i = 1; i <= MaxLayersMinus1; i++ ) |  |
| for( j = 0; j < NumDirectRefLayers[ layer\_id\_in\_nuh[ i ] ]; j++ ) { |  |
| layerIdx = LayerIdxInVps[ RefLayerId[ layer\_id\_in\_nuh[ i ] ][ j ] ] |  |
| if( tiles\_in\_use\_flag[ i ] && tiles\_in\_use\_flag[ layerIdx ] ) |  |
| **tile\_boundaries\_aligned\_flag**[ i ][ j ] | u(1) |
| } |  |
| } |  |
| **wpp\_not\_in\_use\_flag** |  |
| if( !wpp\_not\_in\_use\_flag ) |  |
| for( i = 0; i <= MaxLayersMinus1; i++ ) |  |
| **wpp\_in\_use\_flag**[ i ] | u(1) |
| **single\_layer\_for\_non\_irap\_flag** | u(1) |
| **higher\_layer\_irap\_skip\_flag** | u(1) |
| **ilp\_restricted\_ref\_layers\_flag** | u(1) |
| if( ilp\_restricted\_ref\_layers\_flag ) |  |
| for( i = 1; i <= MaxLayersMinus1; i++ ) |  |
| for( j = 0; j < NumDirectRefLayers[ layer\_id\_in\_nuh[ i ] ]; j++ ) { |  |
| **min\_spatial\_segment\_offset\_plus1**[ i ][ j ] | ue(v) |
| if( min\_spatial\_segment\_offset\_plus1[ i ][ j ] > 0 ) { |  |
| **ctu\_based\_offset\_enabled\_flag**[ i ][ j ] | u(1) |
| if( ctu\_based\_offset\_enabled\_flag[ i ][ j ] ) |  |
| **min\_horizontal\_ctu\_offset\_plus1**[ i ][ j ] | ue(v) |
| } |  |
| } |  |
| **video\_signal\_info\_idx\_present\_flag** | u(1) |
| if( video\_signal\_info\_idx\_present\_flag ) |  |
| **vps\_num\_video\_signal\_info\_minus1** | u(4) |
| for( i = 0; i <= vps\_num\_video\_signal\_info\_minus1; i++ ) |  |
| video\_signal\_info( ) |  |
| if( video\_signal\_info\_idx\_present\_flag && vps\_num\_video\_signal\_info\_minus1 > 0 ) |  |
| for( i = 1; i <= MaxLayersMinus1; i++ ) |  |
| **vps\_video\_signal\_info\_idx**[ i ] | u(4) |
| **vps\_vui\_bsp\_hrd\_present\_flag** | u(1) |
| if( vps\_vui\_bsp\_hrd\_present\_flag ) |  |
| vps\_vui\_bsp\_hrd\_parameters( ) |  |
| } |  |

Video signal info syntax

|  |  |
| --- | --- |
| video\_signal\_info( ) { | Descriptor |
| **video\_vps\_format** | u(3) |
| **video\_full\_range\_vps\_flag** | u(1) |
| **colour\_primaries\_vps** | u(8) |
| **transfer\_characteristics\_vps** | u(8) |
| **matrix\_coeffs\_vps** | u(8) |
| } |  |

VPS VUI bitstream partition HRD parameters syntax

|  |  |
| --- | --- |
| vps\_vui\_bsp\_hrd\_parameters( ){ | **Descriptor** |
| **vps\_num\_bsp\_hrd\_parameters\_minus1** | ue(v) |
| for( i = 0; i <= vps\_num\_bsp\_hrd\_parameters\_minus1; i++ ) { |  |
| if( i > 0 ) |  |
| **bsp\_cprms\_present\_flag**[ i ] | u(1) |
| hrd\_parameters( bsp\_cprms\_present\_flag[ i ], vps\_max\_sub\_layers\_minus1 ) |  |
| } |  |
| for( h=1; h <= vps\_num\_layer\_sets\_minus1; h++ ) { |  |
| **num\_bitstream\_partitions**[ h ] | ue(v) |
| for( i = 0; i < num\_bitstream\_partitions[ h ]; i++ ) |  |
| for( j = 0; j <= vps\_max\_layers\_minus1; j++ ) |  |
| if( layer\_id\_included\_flag[ h ][ j ] ) |  |
| **layer\_in\_bsp\_flag**[ h ][ i ][ j ] | u(1) |
| if( num\_bitstream\_partitions[ h ] ) { |  |
| **num\_bsp\_sched\_combinations**[ h ] | ue(v) |
| for( i = 0; i < num\_bsp\_sched\_combinations[ h ]; i++ ) |  |
| for( j = 0; j < num\_bitstream\_partitions[ h ]; j++ ) { |  |
| **bsp\_comb\_hrd\_idx**[ h ][ i ][ j ] | ue(v) |
| **bsp\_comb\_sched\_idx**[ h ][ i ][ j ] | ue(v) |
| } |  |
| } |  |
| } |  |
| } |  |

* + - * 1. Sequence parameter set RBSP syntax

|  |  |
| --- | --- |
| seq\_parameter\_set\_rbsp( ) { | Descriptor |
| **sps\_video\_parameter\_set\_id** | u(4) |
| if( nuh\_layer\_id = = 0 ) { |  |
| **sps\_max\_sub\_layers\_minus1** | u(3) |
| **sps\_temporal\_id\_nesting\_flag** | u(1) |
| profile\_tier\_level( 1, sps\_max\_sub\_layers\_minus1 ) |  |
| } |  |
| **sps\_seq\_parameter\_set\_id** | ue(v) |
| if( nuh\_layer\_id > 0 ) { |  |
| **update\_rep\_format\_flag** | u(1) |
| if( update\_rep\_format\_flag ) |  |
| **sps\_rep\_format\_idx** | u(8) |
| } else { |  |
| **chroma\_format\_idc** | ue(v) |
| if( chroma\_format\_idc = = 3 ) |  |
| **separate\_colour\_plane\_flag** | u(1) |
| **pic\_width\_in\_luma\_samples** | ue(v) |
| **pic\_height\_in\_luma\_samples** | ue(v) |
| } |  |
| **conformance\_window\_flag** | u(1) |
| if( conformance\_window\_flag ) { |  |
| **conf\_win\_left\_offset** | ue(v) |
| **conf\_win\_right\_offset** | ue(v) |
| **conf\_win\_top\_offset** | ue(v) |
| **conf\_win\_bottom\_offset** | ue(v) |
| } |  |
| if( nuh\_layer\_id = = 0 ) { |  |
| **bit\_depth\_luma\_minus8** | ue(v) |
| **bit\_depth\_chroma\_minus8** | ue(v) |
| } |  |
| **log2\_max\_pic\_order\_cnt\_lsb\_minus4** | ue(v) |
| **sps\_sub\_layer\_ordering\_info\_present\_flag** | u(1) |
| for( i = ( sps\_sub\_layer\_ordering\_info\_present\_flag ? 0 : sps\_max\_sub\_layers\_minus1 );  i <= sps\_max\_sub\_layers\_minus1; i++ ) { |  |
| **sps\_max\_dec\_pic\_buffering\_minus1**[ i ] | ue(v) |
| **sps\_max\_num\_reorder\_pics**[ i ] | ue(v) |
| **sps\_max\_latency\_increase\_plus1**[ i ] | ue(v) |
| } |  |
| **log2\_min\_luma\_coding\_block\_size\_minus3** | ue(v) |
| **log2\_diff\_max\_min\_luma\_coding\_block\_size** | ue(v) |
| **log2\_min\_transform\_block\_size\_minus2** | ue(v) |
| **log2\_diff\_max\_min\_transform\_block\_size** | ue(v) |
| **max\_transform\_hierarchy\_depth\_inter** | ue(v) |
| **max\_transform\_hierarchy\_depth\_intra** | ue(v) |
| **scaling\_list\_enabled\_flag** | u(1) |
| if( scaling\_list\_enabled\_flag ) { |  |
| if( nuh\_layer\_id > 0 ) |  |
| **sps\_infer\_scaling\_list\_flag** | u(1) |
| if( sps\_infer\_scaling\_list\_flag ) |  |
| **sps\_scaling\_list\_ref\_layer\_id** | u(6) |
| else { |  |
| **sps\_scaling\_list\_data\_present\_flag** | u(1) |
| if( sps\_scaling\_list\_data\_present\_flag ) |  |
| scaling\_list\_data( ) |  |
| } |  |
| } |  |
| **amp\_enabled\_flag** | u(1) |
| **sample\_adaptive\_offset\_enabled\_flag** | u(1) |
| **pcm\_enabled\_flag** | u(1) |
| if( pcm\_enabled\_flag ) { |  |
| **pcm\_sample\_bit\_depth\_luma\_minus1** | u(4) |
| **pcm\_sample\_bit\_depth\_chroma\_minus1** | u(4) |
| **log2\_min\_pcm\_luma\_coding\_block\_size\_minus3** | ue(v) |
| **log2\_diff\_max\_min\_pcm\_luma\_coding\_block\_size** | ue(v) |
| **pcm\_loop\_filter\_disabled\_flag** | u(1) |
| } |  |
| **num\_short\_term\_ref\_pic\_sets** | ue(v) |
| for( i = 0; i < num\_short\_term\_ref\_pic\_sets; i++) |  |
| short\_term\_ref\_pic\_set( i ) |  |
| **long\_term\_ref\_pics\_present\_flag** | u(1) |
| if( long\_term\_ref\_pics\_present\_flag ) { |  |
| **num\_long\_term\_ref\_pics\_sps** | ue(v) |
| for( i = 0; i < num\_long\_term\_ref\_pics\_sps; i++ ) { |  |
| **lt\_ref\_pic\_poc\_lsb\_sps**[ i ] | u(v) |
| **used\_by\_curr\_pic\_lt\_sps\_flag**[ i ] | u(1) |
| } |  |
| } |  |
| **sps\_temporal\_mvp\_enabled\_flag** | u(1) |
| **strong\_intra\_smoothing\_enabled\_flag** | u(1) |
| **vui\_parameters\_present\_flag** | u(1) |
| if( vui\_parameters\_present\_flag ) |  |
| vui\_parameters( ) |  |
| **sps\_extension\_flag** [Ed. (GT): Syntax and semantics should be moved to base spec later.] | u(1) |
| if( sps\_extension\_flag ) { |  |
| for ( i = 0; i < 8; i++ ) |  |
| **sps\_extension\_type\_flag**[ i ] | u(1) |
| if( sps\_extension\_type\_flag[ 1 ] ) |  |
| sps\_multilayer\_extension( ) |  |
| if( sps\_extension\_type\_flag[ 7 ] ) |  |
| while( more\_rbsp\_data( ) ) |  |
| **sps\_extension\_data\_flag** | u(1) |
| } |  |
| rbsp\_trailing\_bits( ) |  |
| } |  |

Sequence parameter set multilayer extension syntax

|  |  |
| --- | --- |
| sps\_multilayer\_extension( ) { | **Descriptor** |
| **inter\_view\_mv\_vert\_constraint\_flag** | u(1) |
| **sps\_ref\_layer\_rps\_enabled\_flag** | u(1) |
| **num\_scaled\_ref\_layer\_offsets** | ue(v) |
| for( i = 0; i < num\_scaled\_ref\_layer\_offsets; i++) { |  |
| **scaled\_ref\_layer\_id**[ i ] | u(6) |
| **scaled\_ref\_layer\_left\_offset**[ scaled\_ref\_layer\_id[ i ] ] | se(v) |
| **scaled\_ref\_layer\_top\_offset**[ scaled\_ref\_layer\_id[ i ] ] | se(v) |
| **scaled\_ref\_layer\_right\_offset**[ scaled\_ref\_layer\_id[ i ] ] | se(v) |
| **scaled\_ref\_layer\_bottom\_offset**[ scaled\_ref\_layer\_id[ i ] ] | se(v) |
| **}** |  |
| } |  |

* + - * 1. Picture parameter set RBSP syntax

|  |  |
| --- | --- |
| pic\_parameter\_set\_rbsp( ) { | Descriptor |
| **pps\_pic\_parameter\_set\_id** | ue(v) |
| **pps\_seq\_parameter\_set\_id** | ue(v) |
| **dependent\_slice\_segments\_enabled\_flag** | u(1) |
| **output\_flag\_present\_flag** | u(1) |
| **num\_extra\_slice\_header\_bits** | u(3) |
| **sign\_data\_hiding\_enabled\_flag** | u(1) |
| **cabac\_init\_present\_flag** | u(1) |
| **num\_ref\_idx\_l0\_default\_active\_minus1** | ue(v) |
| **num\_ref\_idx\_l1\_default\_active\_minus1** | ue(v) |
| **init\_qp\_minus26** | se(v) |
| **constrained\_intra\_pred\_flag** | u(1) |
| **transform\_skip\_enabled\_flag** | u(1) |
| **cu\_qp\_delta\_enabled\_flag** | u(1) |
| if( cu\_qp\_delta\_enabled\_flag ) |  |
| **diff\_cu\_qp\_delta\_depth** | ue(v) |
| **pps\_cb\_qp\_offset** | se(v) |
| **pps\_cr\_qp\_offset** | se(v) |
| **pps\_slice\_chroma\_qp\_offsets\_present\_flag** | u(1) |
| **weighted\_pred\_flag** | u(1) |
| **weighted\_bipred\_flag** | u(1) |
| **transquant\_bypass\_enabled\_flag** | u(1) |
| **tiles\_enabled\_flag** | u(1) |
| **entropy\_coding\_sync\_enabled\_flag** | u(1) |
| if( tiles\_enabled\_flag ) { |  |
| **num\_tile\_columns\_minus1** | ue(v) |
| **num\_tile\_rows\_minus1** | ue(v) |
| **uniform\_spacing\_flag** | u(1) |
| if( !uniform\_spacing\_flag ) { |  |
| for( i = 0; i < num\_tile\_columns\_minus1; i++ ) |  |
| **column\_width\_minus1**[ i ] | ue(v) |
| for( i = 0; i < num\_tile\_rows\_minus1; i++ ) |  |
| **row\_height\_minus1**[ i ] | ue(v) |
| } |  |
| **loop\_filter\_across\_tiles\_enabled\_flag** | u(1) |
| } |  |
| **pps\_loop\_filter\_across\_slices\_enabled\_flag** | u(1) |
| **deblocking\_filter\_control\_present\_flag** | u(1) |
| if( deblocking\_filter\_control\_present\_flag ) { |  |
| **deblocking\_filter\_override\_enabled\_flag** | u(1) |
| **pps\_deblocking\_filter\_disabled\_flag** | u(1) |
| if( !pps\_deblocking\_filter\_disabled\_flag ) { |  |
| **pps\_beta\_offset\_div2** | se(v) |
| **pps\_tc\_offset\_div2** | se(v) |
| } |  |
| } |  |
| if( nuh\_layer\_id > 0 ) |  |
| **pps\_infer\_scaling\_list\_flag** | u(1) |
| if( pps\_infer\_scaling\_list\_flag ) |  |
| **pps\_scaling\_list\_ref\_layer\_id** | u(6) |
| else { |  |
| **pps\_scaling\_list\_data\_present\_flag** | u(1) |
| if( pps\_scaling\_list\_data\_present\_flag ) |  |
| scaling\_list\_data( ) |  |
| } |  |
| **lists\_modification\_present\_flag** | u(1) |
| **log2\_parallel\_merge\_level\_minus2** | ue(v) |
| **slice\_segment\_header\_extension\_present\_flag** | u(1) |
| **pps\_extension\_flag** | u(1) |
| if( pps\_extension\_flag ) |  |
| while( more\_rbsp\_data( ) ) |  |
| **pps\_extension\_data\_flag** | u(1) |
| rbsp\_trailing\_bits( ) |  |
| } |  |

* + - * 1. Supplemental enhancement information RBSP syntax

The specifications in subclause 7.3.2.4 apply.

* + - * 1. Access unit delimiter RBSP syntax

The specifications in subclause 7.3.2.5 apply.

* + - * 1. End of sequence RBSP syntax

The specifications in subclause 7.3.2.6 apply.

* + - * 1. End of bitstream RBSP syntax

The specifications in subclause 7.3.2.7 apply.

* + - * 1. Filler data RBSP syntax

The specifications in subclause 7.3.2.8 apply.

* + - * 1. Slice segment layer RBSP syntax

The specifications in subclause 7.3.2.9 apply.

* + - * 1. RBSP slice segment trailing bits syntax

The specifications in subclause 7.3.2.10 apply.

* + - * 1. RBSP trailing bits syntax

The specifications in subclause 7.3.2.11 apply.

* + - * 1. Byte alignment syntax

The specifications in subclause 7.3.2.12 apply.

* + - 1. Profile, tier and level syntax

|  |  |
| --- | --- |
| profile\_tier\_level(  profilePresentFlag, maxNumSubLayersMinus1 ) { | **Descriptor** |
| if( profilePresentFlag ) { |  |
| **general\_profile\_space** | u(2) |
| **general\_tier\_flag** | u(1) |
| **general\_profile\_idc** | u(5) |
| for( j = 0; j < 32; j++ ) |  |
| **general\_profile\_compatibility\_flag**[ j ] | u(1) |
| **general\_progressive\_source\_flag** | u(1) |
| **general\_interlaced\_source\_flag** | u(1) |
| **general\_non\_packed\_constraint\_flag** | u(1) |
| **general\_frame\_only\_constraint\_flag** | u(1) |
| **general\_reserved\_zero\_44bits** | u(44) |
| } |  |
| **general\_level\_idc** | u(8) |
| for( i = 0; i < maxNumSubLayersMinus1; i++ ) { |  |
| **sub\_layer\_profile\_present\_flag**[ i ] | u(1) |
| **sub\_layer\_level\_present\_flag**[ i ] | u(1) |
| } |  |
| if( maxNumSubLayersMinus1 > 0 ) |  |
| for( i = maxNumSubLayersMinus1; i < 8; i++ ) |  |
| **reserved\_zero\_2bits**[ i ] | u(2) |
| for( i = 0; i < maxNumSubLayersMinus1; i++ ) { |  |
| if( sub\_layer\_profile\_present\_flag[ i ] ) { |  |
| **sub\_layer\_profile\_space**[ i ] | u(2) |
| **sub\_layer\_tier\_flag**[ i ] | u(1) |
| **sub\_layer\_profile\_idc**[ i ] | u(5) |
| for( j = 0; j < 32; j++ ) |  |
| **sub\_layer\_profile\_compatibility\_flag**[ i ][ j ] | u(1) |
| **sub\_layer\_progressive\_source\_flag**[ i ] | u(1) |
| **sub\_layer\_interlaced\_source\_flag**[ i ] | u(1) |
| **sub\_layer\_non\_packed\_constraint\_flag**[ i ] | u(1) |
| **sub\_layer\_frame\_only\_constraint\_flag**[ i ] | u(1) |
| **sub\_layer\_reserved\_zero\_44bits**[ i ] | u(44) |
| } |  |
| if( sub\_layer\_level\_present\_flag[ i ] ) |  |
| **sub\_layer\_level\_idc**[ i ] | u(8) |
| } |  |
| } |  |

* + - 1. Scaling list data syntax

The specifications in subclause 7.3.4 apply.

* + - 1. Supplemental enhancement information message syntax

The specifications in subclause 7.3.5 apply.

* + - 1. Slice segment header syntax
         1. General slice segment header syntax

|  |  |
| --- | --- |
| slice\_segment\_header( ) { | Descriptor |
| **first\_slice\_segment\_in\_pic\_flag** | u(1) |
| if( nal\_unit\_type >= BLA\_W\_LP && nal\_unit\_type <= RSV\_IRAP\_VCL23 ) |  |
| **no\_output\_of\_prior\_pics\_flag** | u(1) |
| **slice\_pic\_parameter\_set\_id** | ue(v) |
| if( !first\_slice\_segment\_in\_pic\_flag ) { |  |
| if( dependent\_slice\_segments\_enabled\_flag ) |  |
| **dependent\_slice\_segment\_flag** | u(1) |
| **slice\_segment\_address** | u(v) |
| } |  |
| if( !dependent\_slice\_segment\_flag ) { |  |
| i = 0 |  |
| if( num\_extra\_slice\_header\_bits > i ) { |  |
| i++ |  |
| **discardable\_flag** | u(1) |
| } |  |
| if( num\_extra\_slice\_header\_bits > i ) { |  |
| i++ |  |
| **cross\_layer\_bla\_flag** | u(1) |
| } |  |
| if( num\_extra\_slice\_header\_bits > i ) { |  |
| i++ |  |
| **poc\_reset\_flag** | u(1) |
| } |  |
| for( ~~i = 1~~; i < num\_extra\_slice\_header\_bits; i++ ) |  |
| **slice\_reserved\_flag**[ i ] | u(1) |
| **slice\_type** | ue(v) |
| if( output\_flag\_present\_flag ) |  |
| **pic\_output\_flag** | u(1) |
| if( separate\_colour\_plane\_flag = = 1 ) |  |
| **colour\_plane\_id** | u(2) |
| if( ( nuh\_layer\_id > 0 && !poc\_lsb\_not\_present\_flag[ LayerIdxInVPS[ nuh\_layer\_id ] ] )  | | ( nal\_unit\_type != IDR\_W\_RADL && nal\_unit\_type != IDR\_N\_LP ) ) |  |
| **slice\_pic\_order\_cnt\_lsb** | u(v) |
| if( nal\_unit\_type != IDR\_W\_RADL && nal\_unit\_type != IDR\_N\_LP ) { |  |
| **short\_term\_ref\_pic\_set\_sps\_flag** | u(1) |
| if( !short\_term\_ref\_pic\_set\_sps\_flag ) |  |
| short\_term\_ref\_pic\_set( num\_short\_term\_ref\_pic\_sets ) |  |
| else if( num\_short\_term\_ref\_pic\_sets > 1 ) |  |
| **short\_term\_ref\_pic\_set\_idx** | u(v) |
| if( long\_term\_ref\_pics\_present\_flag ) { |  |
| if( num\_long\_term\_ref\_pics\_sps > 0 ) |  |
| **num\_long\_term\_sps** | ue(v) |
| **num\_long\_term\_pics** | ue(v) |
| for( i = 0; i < num\_long\_term\_sps + num\_long\_term\_pics; i++ ) { |  |
| if( i < num\_long\_term\_sps ) { |  |
| if( num\_long\_term\_ref\_pics\_sps > 1 ) |  |
| **lt\_idx\_sps**[ i ] | u(v) |
| } else { |  |
| **poc\_lsb\_lt**[ i ] | u(v) |
| **used\_by\_curr\_pic\_lt\_flag**[ i ] | u(1) |
| } |  |
| **delta\_poc\_msb\_present\_flag**[ i ] | u(1) |
| if( delta\_poc\_msb\_present\_flag[ i ] ) |  |
| **delta\_poc\_msb\_cycle\_lt**[ i ] | ue(v) |
| } |  |
| } |  |
| if( sps\_temporal\_mvp\_enabled\_flag ) |  |
| **slice\_temporal\_mvp\_enabled\_flag** | u(1) |
| } |  |
| if( nuh\_layer\_id > 0 && !all\_ref\_layers\_active\_flag &&  NumDirectRefLayers[ nuh\_layer\_id ] > 0 ) { |  |
| **inter\_layer\_pred\_enabled\_flag** | u(1) |
| if( inter\_layer\_pred\_enabled\_flag && NumDirectRefLayers[ nuh\_layer\_id ] > 1) { |  |
| if( !max\_one\_active\_ref\_layer\_flag ) |  |
| **num\_inter\_layer\_ref\_pics\_minus1** | u(v) |
| if( NumActiveRefLayerPics != NumDirectRefLayers[ nuh\_layer\_id ] ) |  |
| for( i = 0; i < NumActiveRefLayerPics; i++ ) |  |
| **inter\_layer\_pred\_layer\_idc[**i ] | u(v) |
| } |  |
| } |  |
| if ( sps\_ref\_layer\_rps\_enabled\_flag && NumDirectRefLayers[ nuh\_layer\_id ] ) |  |
| for( refLayerFound = 0; i = NumDirectRefLayers[ nuh\_layer\_id ] – 1;   i >= 0 && !refLayerFound; i-- ) {  // [Ed. (MH): For a design option allowing diagonal inter-layer prediction from more than one direct reference layer, refLayerFound would be removed.] |  |
| if( NumDirectRefLayers[ nuh\_layer\_id ] > 1 ) |  |
| **ref\_layer\_rps\_present\_flag**[ i ] | u(1) |
| refLayerFound = ref\_layer\_rps\_present\_flag[ i ] |  |
| if( ref\_layer\_rps\_present\_flag[ i ] ) |  |
| short\_term\_ref\_pic\_set( num\_short\_term\_ref\_pic\_sets ) |  |
| } |  |
| if( sample\_adaptive\_offset\_enabled\_flag ) { |  |
| **slice\_sao\_luma\_flag** | u(1) |
| **slice\_sao\_chroma\_flag** | u(1) |
| } |  |
| if( slice\_type = = P | | slice\_type = = B ) { |  |
| **num\_ref\_idx\_active\_override\_flag** | u(1) |
| if( num\_ref\_idx\_active\_override\_flag ) { |  |
| **num\_ref\_idx\_l0\_active\_minus1** | ue(v) |
| if( slice\_type = = B ) |  |
| **num\_ref\_idx\_l1\_active\_minus1** | ue(v) |
| } |  |
| if( lists\_modification\_present\_flag && NumPicTotalCurr > 1 ) |  |
| ref\_pic\_lists\_modification( ) |  |
| if( slice\_type = = B ) |  |
| **mvd\_l1\_zero\_flag** | u(1) |
| if( cabac\_init\_present\_flag ) |  |
| **cabac\_init\_flag** | u(1) |
| if( slice\_temporal\_mvp\_enabled\_flag ) { |  |
| if( slice\_type = = B ) |  |
| **collocated\_from\_l0\_flag** | u(1) |
| if( ( collocated\_from\_l0\_flag && num\_ref\_idx\_l0\_active\_minus1 > 0 ) | |  ( !collocated\_from\_l0\_flag && num\_ref\_idx\_l1\_active\_minus1 > 0 ) ) |  |
| **collocated\_ref\_idx** | ue(v) |
| } |  |
| if( ( weighted\_pred\_flag && slice\_type = = P ) | |  ( weighted\_bipred\_flag && slice\_type = = B ) ) |  |
| pred\_weight\_table( ) |  |
| **five\_minus\_max\_num\_merge\_cand** | ue(v) |
| } |  |
| **slice\_qp\_delta** | se(v) |
| if( pps\_slice\_chroma\_qp\_offsets\_present\_flag ) { |  |
| **slice\_cb\_qp\_offset** | se(v) |
| **slice\_cr\_qp\_offset** | se(v) |
| } |  |
| if( deblocking\_filter\_override\_enabled\_flag ) |  |
| **deblocking\_filter\_override\_flag** | u(1) |
| if( deblocking\_filter\_override\_flag ) { |  |
| **slice\_deblocking\_filter\_disabled\_flag** | u(1) |
| if( !slice\_deblocking\_filter\_disabled\_flag ) { |  |
| **slice\_beta\_offset\_div2** | se(v) |
| **slice\_tc\_offset\_div2** | se(v) |
| } |  |
| } |  |
| if( pps\_loop\_filter\_across\_slices\_enabled\_flag &&  ( slice\_sao\_luma\_flag | | slice\_sao\_chroma\_flag | |  !slice\_deblocking\_filter\_disabled\_flag ) ) |  |
| **slice\_loop\_filter\_across\_slices\_enabled\_flag** | u(1) |
| } |  |
| if( tiles\_enabled\_flag | | entropy\_coding\_sync\_enabled\_flag ) { |  |
| **num\_entry\_point\_offsets** | ue(v) |
| if( num\_entry\_point\_offsets > 0 ) { |  |
| **offset\_len\_minus1** | ue(v) |
| for( i = 0; i < num\_entry\_point\_offsets; i++ ) |  |
| **entry\_point\_offset\_minus1**[ i ] | u(v) |
| } |  |
| } |  |
| if( slice\_segment\_header\_extension\_present\_flag ) { |  |
| **slice\_segment\_header\_extension\_length** | ue(v) |
| for( i = 0; i < slice\_segment\_header\_extension\_length; i++) |  |
| **slice\_segment\_header\_extension\_data\_byte**[ i ] | u(8) |
| } |  |
| byte\_alignment( ) |  |
| } |  |

* + - * 1. Reference picture list modification syntax

The specifications in subclause 7.3.6.2 apply.

* + - * 1. Weighted prediction parameters syntax

The specifications in subclause 7.3.6.3 apply.

* + - 1. Short-term reference picture set syntax

The specifications in subclause 7.3.7 apply.

* + - 1. Slice segment data syntax
         1. General slice segment data syntax

The specifications in subclause 7.3.8.1 apply.

* + - * 1. Coding tree unit syntax

The specifications in subclause 7.3.8.2 apply.

* + - * 1. Sample adaptive offset syntax

The specifications in subclause 7.3.8.3 apply.

* + - * 1. Coding quadtree syntax

The specifications in subclause 7.3.8.4 apply.

* + - * 1. Coding unit syntax

The specifications in subclause 7.3.8.5 apply.

* + - * 1. Prediction unit syntax

The specifications in subclause 7.3.8.6 apply.

* + - * 1. PCM sample syntax

The specifications in subclause 7.3.8.7 apply.

* + - * 1. Transform tree syntax

The specifications in subclause 7.3.8.8 apply.

* + - * 1. Motion vector difference syntax

The specifications in subclause 7.3.8.9 apply.

* + - * 1. Transform unit syntax

The specifications in subclause 7.3.8.10 apply.

* + - * 1. Residual coding syntax

The specifications in subclause 7.3.8.11 apply.

* + 1. Semantics
       1. General
       2. NAL unit semantics
          1. General NAL unit semantics

The specifications in subclause 7.4.2.1 apply.

* + - * 1. NAL unit header semantics

The specifications in subclause 7.4.2.2 apply with following modifications and additions.

**nal\_unit\_type** specifies the type of RBSP data structure contained in the NAL unit as specified in Table 7 1.

When one picture picA of a layer layerA has nal\_unit\_type equal to TSA\_N or TSA\_R, each picture in the same access unit as picA in a direct or indirect reference layer of layerA shall have nal\_unit\_type equal to TSA\_N or TSA\_R.

When one picture picA of a layer layerA has nal\_unit\_type equal to STSA\_N or STSA\_R, each picture in the same access unit as picA in a direct or indirect reference layer of layerA shall have nal\_unit\_type equal to STSA\_N or STSA\_R.

**nuh\_layer\_id** specifies the identifier of the layer. The value of nuh\_layer\_id shall be in the range of 0 to 62, inclusive. The value of 63 may be specified in the future by ITU-T | ISO/IEC. Decoders shall ignore all data that follow the value 63 for nuh\_layer\_id in a NAL unit.

NOTE  – It is anticipated that in a future super multiview coding extension of this specification, the value of 63 for nuh\_layer\_id will be used to indicate an extended layer identifier.

When nal\_unit\_type is equal to AUD\_NUT, the value of nuh\_layer\_id shall be equal to the minimum of the nuh\_layer\_id values of all VCL NAL units in the access unit.

When nal\_unit\_type is equal to VPS\_NUT, the value of nuh\_layer\_id shall be equal to 0. Decoder shall ignore NAL units with nal\_unit\_type equal to VPS\_NUT and nuh\_layer\_id greater than 0.

When nal\_unit\_type is equal to PPS\_NUT and the NAL unit contains the active PPS for a layer layerA with nuh\_layer\_id equal to nuhLayerIdA, the value of nuh\_layer\_id shall be equal to 0, nuhLayerIdA, or the nuh\_layer\_id of a direct or indirect reference layer of layerA.

When nal\_unit\_type is equal to SPS\_NUT and the NAL unit contains the active SPS for a layer layerA with nuh\_layer\_id equal to nuhLayerIdA, the value of nuh\_layer\_id shall be equal to 0, nuhLayerIdA, or the nuh\_layer\_id of a direct or indirect reference layer of layerA.

* + - * 1. Encapsulation of an SODB within an RBSP (informative)

The specifications in subclause 7.4.2.3 apply.

* + - * 1. Order of NAL units and association to coded pictures, access units, and coded video sequences

General

The specifications in subclause 7.4.2.4.1 apply with the following additions.

A coded picture with nuh\_layer\_id equal to nuhLayerIdA shall precede, in decoding order, all coded pictures with nuh\_layer\_id greater than nuhLayerIdA in the same access unit.

Order of VPS, SPS and PPS RBSPs and their activation

The specifications in subclause 7.4.2.4.2 apply with the following additions.

The contents of the hrd\_parameters( ) syntax structure shall remain unchanged within a sequence of activated SPS RBSPs, in their activation order, from any activated SPS RBSP until the end of the bitstream or up to but excluding an SPS RBSP that is activated within the next access unit in which at least one of the following conditions is true:

* The access unit includes a picture for each nuh\_layer\_id value in TargetDecLayerIdList and each picture in the access unit is an IDR picture.
* The access unit includes an IRAP picture with nuh\_layer\_id equal to 0 for which NoClrasOutputFlag is equal to 1.

An activated VPS RBSP shall remain active until the end of the bitstream or until it is deactivated by another VPS RBSP in an access unit in which at least one of the following conditions is true:

* The access unit includes a picture for each nuh\_layer\_id value in TargetDecLayerIdList and each picture in the access unit is an IDR picture.
* The access unit includes an IRAP picture with nuh\_layer\_id equal to 0 for which NoClrasOutputFlag is equal to 1.

An activated SPS RBSP for a particular layer with nuh\_layer\_id greater than 0 shall remain active for a sequence of pictures in decoding order with that nuh\_layer\_id value starting from a picture, inclusive, that is an IRAP picture with NoRaslOutputFlag equal to 1 or for which FirstPicInLayerDecodedFlag[ nuh\_layer\_id ] is equal to 0, until the next picture, exclusive, that is an IRAP picture with NoRaslOutputFlag equal to 1 or for which FirstPicInLayerDecodedFlag[ nuh\_layer\_id ] is equal to 0.

The variable NumShortTermRefPicSets[ layerId ] is set equal to num\_short\_term\_ref\_pic\_sets of the active SPS for the layer with nuh\_layer\_id equal to layerId.

Any SPS NAL unit containing the value of sps\_seq\_parameter\_set\_id for the active SPS RBSP for a particular non-base layer shall have the same content as that of the active SPS RBSP for the particular non-base layer unless it follows the last coded picture for which the active SPS RBSP for the particular non-base layer is required to be active for the particular non-base layer and precedes the first NAL unit that activates an SPS RBSP with the same value of seq\_parameter\_set\_id.

During operation of the decoding process for NAL units of a non-base layer, the values of parameters of the active VPS RBSP, the active SPS RBSP for the non-base layer, and the active PPS RBSP for the non-base layer are considered in effect. For interpretation of SEI messages applicable to a coded picture of a non-base layer, the values of the active VPS RBSP, the active SPS RBSP for the non-base layer, and the active PPS RBSP for the non-base layer for the operation of the decoding process for the VCL NAL units of the coded picture are considered in effect unless otherwise specified in the SEI message semantics.

Order of access units and their association to CVS

The specifications in subclause 7.4.2.4.3 apply.

Order of NAL units and coded pictures and association to access units

The specifications in subclause 7.4.2.4.4 apply.

Order of VCL NAL units and association to coded pictures

The specifications in subclause 7.4.2.4.5 apply.

* + - 1. Raw byte sequence payloads, trailing bits, and byte alignment semantics
         1. Video parameter set RBSP semantics

The specifications in subclause 7.4.3.1 apply with following modifications and additions:

* layerSetLayerIdList *is replaced by* LayerSetLayerIdList.
* numLayersInIdList *is replaced by* NumLayersInIdList.
* *Replace* "Each operation point is identified by the associated layer identifier list, denoted as OpLayerIdList, which consists of the list of nuh\_layer\_id values of all NAL units included in the operation point, in increasing order of nuh\_layer\_id values, and a variable OpTid, which is equal to the highest TemporalId of all NAL units included in the operation point." *with* "Each operation point is identified by the a list of nuh\_layer\_id values of all the pictures that are to be output, in increasing order of nuh\_layer\_id values, denoted as OptLayerIdList, and a variable OpTid, which is equal to the highest TemporalId of all NAL units included in the operation point. The layer identifier list associated with the list OptLayerIdList, denoted as OpLayerIdList, consists of the list of nuh\_layer\_id values of all NAL units included in the operation point, in increasing order of nuh\_layer\_id values.".

**vps\_max\_layers\_minus1** plus 1 specifies the maximum allowed number of layers in the CVS. vps\_max\_layers\_minus1 shall be less than 63 in bitstreams conforming to this version of this Specification. The value of 63 for vps\_max\_layers\_minus1 is reserved for future use by ITU-T | ISO/IEC. Although the value of vps\_max\_layers\_minus1 is required to be less than 63 in this version of this Specification, decoders shall allow a value of vps\_max\_layers\_minus1 equal to 63 to appear in the syntax.

NOTE  – It is anticipated that in a future super multiview coding extension of this specification, the value of 63 for vps\_max\_layers\_minus1 will be used to indicate an extended number of layers.

The variable MaxLayersMinus1 is set equal to Min( 62, vps\_max\_layers\_minus1 ).

**vps\_max\_layer\_id** specifies the maximum allowed value of nuh\_layer\_id of all NAL units in the CVS. vps\_max\_layer\_id shall be less than 63 in bitstreams conforming to this version of this Specification. The value of 63 for vps\_max\_layer\_id is reserved for future use by ITU-T | ISO/IEC. Although the value of vps\_max\_layer\_id is required to be less than 63 in this version of this Specification, decoders shall allow a value of vps\_max\_layer\_id equal to 63 to appear in the syntax.

**vps\_extension\_offset** specifies the byte offset, starting from the beginning of the VPS NAL unit, of the next set of fixed-length coded information starting from avc\_base\_layer\_flag, when present, in the VPS NAL unit. When present, emulation prevention bytes that appear in the VPS NAL unit are counted for purposes of byte offset identification.

NOTE – VPS information for non-base layer or view starts from a byte-aligned position of the VPS NAL unit, with fixed-length coded information that is essential for session negotiation and/or capability exchange. The byte offset specified by vps\_extension\_offset would then help to locate and access those essential information in the VPS NAL unit without the need of entropy decoding, which may not be equipped with some network entities that may desire to access only the information in the VPS that is essential for session negotiation and/or capability exchange.

**vps\_extension\_flag** equal to 0 specifies that no vps\_extension( ) syntax structure is present in the VPS RBSP syntax structure. vps\_extension\_flag equal to 1 specifies that the vps\_extension( ) syntax structure is present in the VPS RBSP syntax structure. When MaxLayersMinus1 is greater than 0, vps\_extension\_flag shall be equal to 1.

**vps\_extension\_alignment\_bit\_equal\_to\_one** shall be equal to 1.

**vps\_extension2\_flag** equal to 0 specifies that no vps\_extension\_data\_flag syntax elements are present in the VPS RBSP syntax structure. vps\_extension2\_flag shall be equal to 0 in bitstreams conforming to this version of this Specification. The value of 1 for vps\_extension2\_flag is reserved for future use by ITU‑T | ISO/IEC. Decoders shall ignore all data that follow the value 1 for vps\_extension2\_flag in a VPS NAL unit.

Video parameter set extension semantics

**avc\_base\_layer\_flag** equal to 1 specifies that the base layer conforms to Rec. ITU-T H.264 | ISO/IEC 14496-10. avc\_base\_layer\_flag equal to 0 specifies that the base layer conforms to this Specification.

[Ed. (YK): For possible support of base layer of other codecs, e.g. MPEG-2, a flag is not sufficient.]

When avc\_base\_layer\_flag is equal to 1, in the Rec. ITU-T H.264 | ISO/IEC 14496-10 conforming base layer, after applying the Rec. ITU-T H.264 | ISO/IEC 14496-10 decoding process for reference picture lists construction the output reference picture lists refPicList0 and refPicList1 (when applicable) does not contain any pictures for which the TemporalId is greater than TemporalId of the coded picture. All sub-bitstreams of the Rec. ITU-T H.264 | ISO/IEC 14496-10 conforming base layer, that can be derived using the sub-bitstream extraction process as specified in Rec. ITU­T H.264 | ISO/IEC 14496-10 subclause G.8.8.1 with any value for temporal\_id as the input shall result in a set of CVSs, with each CVS conforming to one or more of the profiles specified in Rec. ITU­T H.264 | ISO/IEC 14496-10 Annexes A, G and H.

**vps\_vui\_present\_flag** equal to 1 specifies that the vps\_vui( ) syntax structure is present in the VPS. vps\_vui\_present\_flag equal to 0 specifies that the vps\_vui( ) syntax structure is not present in the VPS.

**vps\_vui\_offset** specifies the byte offset, starting from the beginning of the VPS NAL unit, of the set of fixed-length coded information starting from bit\_rate\_present\_vps\_flag, when present, in the VPS NAL unit. When present, emulation prevention bytes that appear in the VPS NAL unit are counted for purposes of byte offset identification.

**splitting\_flag** equal to 1 indicates that the dimension\_id[ i ][ j ] syntax elements are not present and that the binary representation of the nuh\_layer\_id value in the NAL unit header are split into NumScalabilityTypes segments with lengths, in bits, according to the values of dimension\_id\_len\_minus1[ j ] and that the values of dimension\_id[ LayerIdxInVps[ nuh\_layer\_id ] ][ j ] are inferred from the NumScalabilityTypes segments. splitting\_flag equal to 0 indicates that the syntax elements dimension\_id[ i ][ j ] are present.

NOTE 1 – When splitting\_flag is equal to 1, scalable identifiers can be derived from the nuh\_layer\_id syntax element in the NAL unit header by a bit masked copy. The respective bit mask for the i-th scalable dimension is defined by the value of the dimension\_id\_len\_minus1[ i ] syntax element and dimBitOffset[ i ] as specified in the semantics of dimension\_id\_len\_minus1[ j ].

**scalability\_mask\_flag**[ i ] equal to 1 indicates that dimension\_id syntax elements corresponding to the i-th scalability dimension in Table F‑1 are present. scalability\_mask\_flag[ i ] equal to 0 indicates that dimension\_id syntax elements corresponding to the i-th scalability dimension are not present.

Table F‑1 – Mapping of ScalabiltyId to scalability dimensions

|  |  |  |
| --- | --- | --- |
| **scalability mask**  **index** | **Scalability dimension** | **ScalabilityId mapping** |
| 0 | Reserved |  |
| 1 | Multiview | View Order Index |
| 2 | spatial/SNR scalability | DependencyId |
| 3 | Auxiliary | AuxId |
| 4-15 | Reserved |  |

NOTE 2 – It is anticipated that in future 3D extensions of this Specification, scalability mask index 0 will be used to indicate depth maps. It is anticipated that in future scalability extensions of this Specification, scalability mask index 2 will be used to indicate spatial/SNR scalability.

**dimension\_id\_len\_minus1**[ j ] plus 1 specifies the length, in bits, of the dimension\_id[ i ][ j ] syntax element.

When splitting\_flag is equal to 1, the following applies:

– The variable dimBitOffset[ 0 ] is set equal to 0 and for j in the range of 1 to  NumScalabilityTypes − 1 , inclusive, dimBitOffset[ j ] is derived as follows:

 (F‑1)

– The value of dimension\_id\_len\_minus1[ NumScalabilityTypes − 1 ] is inferred to be equal to 5 − dimBitOffset[ NumScalabilityTypes − 1 ].

– The value of dimBitOffset[ NumScalabilityTypes ] is set equal to 6.

It is a requirement of bitstream conformance that when NumScalabilityTypes is greater than 0, dimBitOffset[ NumScalabilityTypes − 1 ] shall be less than 6.

**vps\_nuh\_layer\_id\_present\_flag** equal to 1specifies that layer\_id\_in\_nuh[ i ] for i from 0 to MaxLayersMinus1, inclusive, are present. vps\_nuh\_layer\_id\_present\_flag equal to 0 specifies that layer\_id\_in\_nuh[ i ] for i from 0 to MaxLayersMinus1, inclusive, are not present.

**layer\_id\_in\_nuh**[ i ] specifies the value of the nuh\_layer\_id syntax element in VCL NAL units of the i-th layer. For i in the range of 0 to MaxLayersMinus1, inclusive, when layer\_id\_in\_nuh[ i ] is not present, the value is inferred to be equal to i.

When i is greater than 0, layer\_id\_in\_nuh[ i ] shall be greater than layer\_id\_in\_nuh[ i – 1 ].

For i from 0 to MaxLayersMinus1, inclusive, the variable LayerIdxInVps[ layer\_id\_in\_nuh[ i ] ] is set equal to i.

**dimension\_id**[ i ][ j ] specifies the identifier of the j-th present scalability dimension type of the i-th layer. The number of bits used for the representation of dimension\_id[ i ][ j ] is dimension\_id\_len\_minus1[ j ] + 1 bits.

Depending on splitting\_flag, the following applies:

– If splitting\_flag is equal to 1, for i from 0 to MaxLayersMinus1, inclusive, and j from 0 to NumScalabilityTypes − 1, inclusive, dimension\_id[ i ][ j ] is inferred to be equal to ( ( layer\_id\_in\_nuh[ i ] & ( (1  <<  dimBitOffset[ j + 1 ] ) − 1) )  >>  dimBitOffset[ j ] ).

– Otherwise ( splitting\_flag is equal to 0 ), for j from 0 to NumScalabilityTypes − 1, inclusive, dimension\_id[ 0 ][ j ] is inferred to be equal to 0.

The variable ScalabilityId[ i ][ smIdx ] specifying the identifier of the smIdx-th scalability dimension type of the i-th layer, the variable ViewOrderIdx[ layer\_id\_in\_nuh[ i ] ] specifying the view order index of the i-th layer, DependencyId[ layer\_id\_in\_nuh[ i ] ] specifying the spatial/SNR scalability identifier of the i-th layer, and the variable ViewScalExtLayerFlag[ layer\_id\_in\_nuh[ i ] ] specifying whether the i-th layer is a view scalability extension layer are derived as follows:

NumViews = 1

for( i = 0; i <= MaxLayersMinus1; i++ ) {

lId = layer\_id\_in\_nuh[ i ]

for( smIdx= 0, j = 0; smIdx < 16; smIdx++ )

if( scalability\_mask\_flag[ smIdx ] )

ScalabilityId[ i ][ smIdx ] = dimension\_id[ i ][ j++ ]

ViewOrderIdx[ lId ] = ScalabilityId[ i ][ 1 ]

DependencyId [ lId  ] = ScalabilityId[ i ][ 2 ]

if( i > 0 && ( ViewOrderIdx[ lId ] != ScalabilityId[ i – 1][ 1 ] ) )

NumViews++

ViewScalExtLayerFlag[ lId ] = ( ViewOrderIdx[ lId ] > 0 )

AuxId[ lId ] = ScalabilityId[ i ][ 3 ]

}

AuxId[ lId ] equal to 0 specifies the layer with nuh\_layer\_id equal to lId does not contain auxiliary pictures. AuxId[ lId ] greater than 0 specifies the type of auxiliary pictures in layer with nuh\_layer\_id equal to lId as specified in Table F‑2.

Table F‑2 – Mapping of AuxId to the type of auxiliary pictures

|  |  |  |
| --- | --- | --- |
| **AuxId** | **Name of AuxId** | **Type of auxiliary pictures** |
| 1 | AUX\_ALPHA | Alpha plane |
| 2 | AUX\_DEPTH | Depth picture |
| 4-127 |  | Reserved |
| 128-143 |  | Unspecified |
| 144-255 |  | Reserved |

NOTE 3 – The interpretation of auxiliary pictures associated with AuxId in the range of 128 to 143, inclusive, is specified through means other than the AuxId value.

AuxId[ lId ] shall be in the range of 0 to 2, inclusive, or 128 to 143, inclusive, for bitstreams conforming to this version of this Specification. Although the value of AuxId[ lId ] shall be in the range of 0 to 2, inclusive, or 128 to 143, inclusive, in this version of this Specification, decoders shall allow values of AuxId[ lId ] in the range of 0 to 255, inclusive.

For an auxiliary picture with nuh\_layer\_id equal to nuhLayerIdA, an associated primary picture, if any, is the picture in the same access unit having AuxId[ nuhLayerIdB ] equal to 0 such that ScalabilityId[ LayerIdxInVps[ nuhLayerIdA ] ][ j ] is equal to ScalabilityId[ LayerIdxInVps[ nuhLayerIdB ] ][ j ] for all values of j in the range of 0 to 2, inclusive, and 4 to 15, inclusive.

It is a requirement of bitstream conformance that there shall be an associated primary picture for each auxiliary picture with AuxId[ nuh\_layer\_id ] equal to AUX\_ALPHA.

NOTE 4 – It is not required that each auxiliary picture of each auxiliary picture type has an associated primary picture. For example, a layer with AuxId[ nuh\_layer\_id ] equal to AUX\_DEPTH may represent a viewpoint of a range sensing camera, while the layers containing primary pictures may represent conventional cameras.

**view\_id\_len** specifies the length, in bits, of the view\_id\_val[ i ] syntax element. The value of view\_id\_len shall be greater than or equal to Ceil( Log2 ( NumViews ) ). [Ed. (GT): Regarding that currently two different views are not required to have different view\_id\_val values the last constraint is not necessary. ]

**view\_id\_val**[ i ] specifies the view identifier of the i-th view specified by the VPS. The length of the view\_id\_val[ i ] syntax element is view\_id\_len bits. When not present, the value of view\_id\_val[ i ] is inferred to be equal to 0.

For each layer with nuh\_layer\_id equal to nuhLayerId, the value ViewId[ nuhLayerId ] is set equal to view\_id\_val[ ViewOrderIdx[ nuhLayerId ] ].

**direct\_dependency\_flag**[ i ][ j ] equal to 0 specifies that the layer with index j is not a direct reference layer for the layer with index i. direct\_dependency\_flag[ i ][ j ] equal to 1 specifies that the layer with index j may be a direct reference layer for the layer with index i. When direct\_dependency\_flag[ i ][ j ] is not present for i and j in the range of 0 to MaxLayersMinus1, it is inferred to be equal to 0.

The variables NumDirectRefLayers[ i ] and RefLayerId[ i ][ j ] are derived as follows:

for( i = 0; i <= MaxLayersMinus1; i++ ) {  
 iNuhLId = layer\_id\_in\_nuh[ i ]  
 NumDirectRefLayers[ iNuhLId ] = 0  
 for( j = 0; j < i; j++ )  
 if( direct\_dependency\_flag[ i ][ j ] )  
 RefLayerId[ iNuhLId ][ NumDirectRefLayers[ iNuhLId ]++ ] = layer\_id\_in\_nuh[ j ]}

The variable NumRefLayers[ i ] is derived as follows:

– NumRefLayers[ i ] is first initialized to 0 for all values of i in the range of 0 and 63, inclusive.

– For each layer with nuh\_layer\_id equal to currLayerId, and for all values of j in the range of 0 to 63, inclusive, the variable recursiveRefLayerFlag[ currLayerId ][ j ] is first initialized to 0. The variable recursiveRefLayerFlag[ currLayerId ][ j ] is then modified using the function setRefLayerFlags( currLayerId ), specified as follows:

for( j = 0; j < NumDirectRefLayers[ currLayerId ]; j++ ) {   
 refLayerId = RefLayerId[ currLayerId ][ j ]  
 recursiveRefLayerFlag[ currLayerId ][ refLayerId ] = 1  
 for( k = 0; k <= 63; k++ )  
 recursiveRefLayerFlag[ currLayerId ][ k ] =   
 recursiveRefLayerFlag[ currLayerId ][ k ] | recursiveRefLayerFlag[ refLayerId ][ k ]  
}

– NumRefLayers [ i ] is modified as follows:

for( i = 0; i <= vps\_max\_layers\_minus1; i++ ) {  
 iNuhLId = layer\_id\_in\_nuh[ i ]  
 setRefLayerFlags( iNuhLId )  
 for( j = 0; j <= 63; j++ )  
 NumRefLayers[ iNuhLId ] += recursiveRefLayerFlag[ iNuhLId ][ j ]  
}

It is a requirement of bitstream conformance that AuxId[ RefLayerId[ nuhLayerIdA ][ j ] ] for any values of nuhLayerIdA and j shall be equal to AuxId[ nuhLayerIdA ], when AuxId[ nuhLayerIdA ] is in the range of 0 to 2, inclusive.

NOTE 5 – In other words, no prediction takes place between layers with a different value of AuxId, when AuxId is in the range of 0 to 2, inclusive.

**vps\_sub\_layers\_max\_minus1\_present\_flag** equal to 1 specifies that the syntax elements sub\_layers\_vps\_max\_minus1[ i ] are present. vps\_sub\_layers\_max\_minus1\_present\_flag equal to 0 specifies that the syntax elements sub\_layers\_vps\_max\_minus1[ i ] are not present.

**sub\_layers\_vps\_max\_minus1**[ i ] plus 1 specifies the maximum number of temporal sub-layers that may be present in the CVS for the layer with nuh\_layer\_id equal to layer\_id\_in\_nuh[ i ]. The value of sub\_layers\_vps\_max\_minus1[ i ] shall be in the range of 0 to vps\_max\_sub\_layers\_minus1, inclusive. When not present, sub\_layers\_vps\_max\_minus1[ i ] is inferred to be equal to vps\_max\_sub\_layers\_minus1.

**max\_tid\_ref\_present\_flag** equal to 1 specifies that the syntax element max\_tid\_il\_ref\_pics\_plus1[ i ][ j ] is present. max\_tid\_ref\_present\_flag equal to 0 specifies that the syntax element max\_tid\_il\_ref\_pics\_plus1[ i ][ j ] is not present.

**max\_tid\_il\_ref\_pics\_plus1**[ i ][ j ] equal to 0 specifies that within the CVS non-IRAP pictures with nuh\_layer\_id equal to layer\_id\_in\_nuh[ i ] are not used as reference for inter-layer prediction for pictures with nuh\_layer\_id equal to layer\_id\_in\_nuh[ j ]. max\_tid\_il\_ref\_pics\_plus1[ i ][ j ] greater than 0 specifies that within the CVS pictures with nuh\_layer\_id equal to layer\_id\_in\_nuh[ i ] and TemporalId greater than max\_tid\_il\_ref\_pics\_plus1[ i ][ j ] – 1 are not used as reference for inter-layer prediction for pictures with nuh\_layer\_id equal to layer\_id\_in\_nuh[ j ]. When not present, max\_tid\_il\_ref\_pics\_plus1[ i ][ j ] is inferred to be equal to 7.

**all\_ref\_layers\_active\_flag** equal to 1 specifies that for each picture referring to the VPS, the reference layer pictures that belong to all direct reference layers of the layer containing the picture and that might be used for inter-layer prediction as specified by the values of sub\_layers\_vps\_max\_minus1[ i ] and max\_tid\_il\_ref\_pics\_plus1[ i ][ j ] are present in the same access unit as the picture and are included in the inter-layer reference picture set of the picture. all\_ref\_layers\_active\_flag equal to 0 specifies that the above restriction may or may not apply. [ Ed. (GT): Consider renaming the syntax element, since not all reference layers are active anymore. ]

**vps\_number\_layer\_sets\_minus1** plus 1 specifies the number of layer sets that are specified by the VPS. The value of vps\_number\_layer\_sets\_minus1 shall be in the range of 0 to 1023, inclusive, and shall be equal to vps\_num\_layer\_sets\_minus1.

**vps\_num\_profile\_tier\_level\_minus1** plus 1 specifies the number of profile\_tier\_level( ) syntax structures in the VPS.

**vps\_profile\_present\_flag**[ i ] equal to 1 specifies that the profile and tier information for layer set i is present in the i-th profile\_tier\_level( ) syntax structure.vps\_profile\_present\_flag[ i ] equal to 0 specifies that profile and tier information is not present in the i-th profile\_tier\_level( ) syntax structure and is inferred.

**profile\_ref\_minus1**[ i ] specifies that the profile and tier information for the i-th profile\_tier\_level( ) syntax structure is inferred to be equal to the profile and tier information for the ( profile \_ref\_minus1[ i ] + 1 )-th layer set. The value of profile\_ref\_minus1[ i ] + 1 shall be less than or equal to i.

**more\_output\_layer\_sets\_than\_default\_flag** equal to 1 specifies that the number of output layer sets specified by the VPS is greater than vps\_number\_layer\_sets\_minus1 + 1. more\_output\_layer\_sets\_than\_default\_flag equal to 0 specifies that the number of output layer sets specified by the VPS is equal to vps\_number\_layer\_sets\_minus1 + 1.

[Ed. (MH): The value of more\_output\_layer\_sets\_than\_default\_flag may be restricted to be equal to 0 by an SHVC profile, such that the number of output layer sets is equal to the number of layer sets.]

**num\_add\_output\_layer\_sets\_minus1** plus 1 specifies the number of output layer sets in addition to the default output layer sets specified by the VPS. The default output layer sets refer to the first vps\_number\_layer\_sets\_minus1 + 1 output layer sets specified by the VPS. For the default output layer sets, either only the highest layer is a target output layer or all layers are target output layers.

**default\_one\_target\_output\_layer\_idc** equal to 1 specifies that only the layer with the highest value of nuh\_layer\_id such that nuh\_layer\_id equal to nuhLayerIdA and AuxId[ nuhLayerIdA ] equal to 0 in each of the default output layer sets is a target output layer. default\_one\_target\_output\_layer\_idc equal to 0 specifies that all layers in each of the default output layer sets are target output layers. default\_one\_target\_output\_layer\_idc shall be equal to 0 or 1 in bitstreams conforming to this version of this Specification. Other values for default\_one\_target\_output\_layer\_idc are reserved for future use by ITU-T | ISO/IEC. [ Ed. GT: Should there be a default behaviour when reserved values appear? ]

**output\_layer\_set\_idx\_minus1**[ i ] plus 1specifies the index of the layer set for the i-th output layer set. The value of output\_layer\_set\_idx\_minus1[ i ] shall be in the range of 0 to vps\_num\_layer\_sets\_minus1 − 1, inclusive. The length of the output\_layer\_set\_idx\_minus1[ i ] syntax element is Ceil( Log2( vps\_num\_layer\_sets\_minus1 ) ) bits.

The layer set for the i-th output layer set with i in the range of 0 to vps\_num\_layer\_sets\_minus1, inclusive, is inferred to be the i-th layer set.

The variable NumSubDpbs[ i ], specifying the number of sub-DPBs for the i-th output layer set, is set equal to NumLayersInIdList[ i ].

**output\_layer\_flag**[ i ][ j ] equal to 1 specifies that the j-th layer in the i-th output layer set is a target output layer. output\_layer\_flag[ i ][ j ] equal to 0 specifies that the j-th layer in the i-th output layer set is not a target output layer.

**profile\_level\_tier\_idx**[ i ] specifies the index, into the list of profile\_tier\_level( ) syntax structures in the VPS, of the profile\_tier\_level( ) syntax structure that applies to i-th output layer set. The length of the profile\_level\_tier\_idx[ i ] syntax element is Ceil( Log2( vps\_num\_profile\_tier\_level\_minus1 + 1 ) ) bits. The value of profile\_level\_tier\_idx[ 0 ] is inferred to be equal to 0. The value of profile\_level\_tier\_idx[ i ] shall be in the range of 0 to vps\_num\_profile\_tier\_level\_minus1, inclusive.

**alt\_output\_layer\_flag** affects picture output as specified in subclause F.13. [Ed. (GT) semantics should be more specific.]

NOTE 1 – When alt\_output\_layer\_flag is equal to 0, pictures that are not at the target output layers are not output. When alt\_output\_layer\_flag equal to 1 and a picture at the a target output layer is not present in an access unit, a picture with highest nuh\_layer\_id among those pictures of the access unit for which PicOutputFlag is equal to 1 and which are not among the target output layers is output.

**rep\_format\_idx\_present\_flag** equal to 1 specifies that the syntax elements vps\_num\_rep\_formats\_minus1 and vps\_rep\_format\_idx[ i ] are present. rep\_format\_idx\_present\_flag equal to 0 specifies that the syntax elements vps\_num\_rep\_formats\_minus1 and vps\_rep\_format\_idx[ i ] are not present.

**vps\_num\_rep\_formats\_minus1** plus 1 specifies the number of the following rep\_format( ) syntax structures in the VPS. When not present, the value of vps\_num\_rep\_formats\_minus1 is inferred to be equal to MaxLayersMinus1.

**vps\_rep\_format\_idx**[ i ] specifies the index, into the list of rep\_format( ) syntax structures in the VPS, of the rep\_format( ) syntax structure that applies to the layer with nuh\_layer\_id equal to layer\_id\_in\_nuh[ i ]. When not present, the value of vps\_rep\_format\_idx[ i ] is inferred to be equal to ( rep\_format\_idx\_present\_flag ? 0 : i ). The value of vps\_rep\_format\_idx[ i ] shall be in the range of 0 to vps\_num\_rep\_formats\_minus1, inclusive.

**max\_one\_active\_ref\_layer\_flag** equal to 1 specifies that at most one picture is used for inter-layer prediction for each picture in the CVS. max\_one\_active\_ref\_layer\_flag equal to 0 specifies that more than one picture may be used for inter-layer prediction for each picture in the CVS.

**poc\_lsb\_not\_present\_flag**[ i ] equal to 1 specifies that the slice\_pic\_order\_cnt\_lsb syntax element is not present in the slice headers of IDR pictures with nuh\_layer\_id equal to layer\_id\_in\_nuh[ i ] in the CVS. poc\_lsb\_not\_present\_flag[ i ] equal to 0 specifies that slice\_pic\_order\_cnt\_lsb syntax element may or may not be present in the slice headers of IDR pictures with nuh\_layer\_id equal to layer\_id\_in\_nuh[ i ] in the CVS. When not present, poc\_lsb\_not\_present\_flag[ i ] is inferred to be equal to 0.

**cross\_layer\_phase\_alignment\_flag** equal to 1 specifies that the locations of the luma sample grids of all layers are aligned at the center sample position of the pictures. cross\_layer\_phase\_alignment\_flag equal to 0 specifies that the locations of the luma sample grids of all layers are aligned at the top-left sample position of the pictures. [Ed. (MH): The semantics should be clarified. What are "the center sample position of the pictures" and "the top-left sample position of the pictures"?]

**direct\_dep\_type\_len\_minus2** plus 2 specifies the number of bits of the direct\_dependency\_type[ i ][ j ] and the default\_direct\_dependency\_type syntax elements. In bitstreams conforming to this version of this Specification the value of direct\_dep\_type\_len\_minus2 shall be equal 0. Although the value of direct\_dep\_type\_len\_minus2 shall be equal to 0 in this version of this Specification, decoders shall allow other values of direct\_dep\_type\_len\_minus2 in the range of 0 to 30, inclusive, to appear in the syntax.

**default\_direct\_dependency\_flag** equal to 1 specifies that the syntax element direct\_dependency\_type[ i ][ j ] is not present and inferred from default\_direct\_dependency\_type. default\_direct\_dependency\_flag equal to 0 indicates that the syntax element direct\_dependency\_type[ i ][ j ] is present.

**default\_direct\_dependency\_type**, when present, specifies the inferred value of direct\_dependency\_type[ i ][ j ]. The length of the default\_direct\_dependency\_type syntax element is direct\_dep\_type\_len\_minus2 + 2 bits. Although the value of default\_direct\_dependency\_type is required to be in the range of 0 to 2, inclusive, in this version of this Specification, decoders shall allow values of default\_direct\_dependency\_type in the range of 3 to 232 − 2, inclusive, to appear in the syntax.

**direct\_dependency\_type**[ i ][ j ] indicates the type of dependency between the layer with nuh\_layer\_id equal layer\_id\_in\_nuh[ i ] and the layer with nuh\_layer\_id equal to layer\_id\_in\_nuh[ j ]. direct\_dependency\_type[ i ][ j ] equal to 0 indicates that the layer with nuh\_layer\_id equal to layer\_id\_in\_nuh[ j ] is used for both inter-layer sample prediction and inter-layer motion prediction of the layer with nuh\_layer\_id equal layer\_id\_in\_nuh[ i ]. direct\_dependency\_type[ i ][ j ] equal to 1 indicates that the layer with nuh\_layer\_id equal to layer\_id\_in\_nuh[ j ] is used for inter-layer sample prediction but not for inter-layer motion prediction of the layer with nuh\_layer\_id equal layer\_id\_in\_nuh[ i ]. direct\_dependency\_type[ i ][ j ] equal to 2 indicates that the layer with nuh\_layer\_id equal to layer\_id\_in\_nuh[ j ] is used for inter-layer motion prediction but not for inter-layer sample prediction of the layer with nuh\_layer\_id equal layer\_id\_in\_nuh[ i ]. The length of the direct\_dependency\_type[ i ][ j ] syntax element is direct\_dep\_type\_len\_minus2 + 2 bits. Although the value of direct\_dependency\_type[ i ][ j ] shall be in the range of 0 to 2, inclusive, in this version of this Specification, decoders shall allow values of direct\_dependency\_type[ i ][ j ] in the range of 3 to 232 − 2, inclusive, to appear in the syntax. When not present, the value of direct\_dependency\_type[ i ][ j ] is inferred to be equal to default\_direct\_dependency\_type.

The variables VpsInterLayerSamplePredictionEnabled[ i ][ j ] and VpsInterLayerMotionPredictionEnabled[ i ][ j ] are derived as follows:

VpsInterLayerSamplePredictionEnabled[ i ][ j ] = direct\_dependency\_type[ i ][ j ] & 0x1 (F‑2)

VpsInterLayerMotionPredictionEnabled[ i ][ j ] = direct\_dependency\_type[ i ][ j ] & 0x2 (F‑3)

[Ed. (JB): May need to define semantic constraints associated with values of VpsInterLayerSamplePredictionEnabled[ i ][ j ] and VpsInterLayerMotionPredictionEnabled[ i ][ j ].]

**vps\_vui\_alignment\_bit\_equal\_to\_one** shall be equal to 1.

Representation format semantics

**chroma\_and\_bit\_depth\_vps\_present\_flag** equal to 1 specifies that the syntax elements, chroma\_format\_vps\_idc, bit\_depth\_vps\_luma\_minus8, and bit\_depth\_vps\_chroma\_minus8 are present and that the syntax element separate\_colour\_plane\_vps\_flag might be present. chroma\_and\_bit\_depth\_vps\_present\_flag equal to 0 specifies that the syntax elements, chroma\_format\_vps\_idc, separate\_colour\_plane\_vps\_flag, bit\_depth\_vps\_luma\_minus8, and bit\_depth\_vps\_chroma\_minus8 are not present and inferred from the previous rep\_format( ) syntax structure in the VPS. The value of chroma\_and\_bit\_depth\_vps\_present\_flag of the first rep\_format( ) syntax structure in the VPS shall be equal to 1.

**pic\_width\_vps\_in\_luma\_samples**, **pic\_height\_vps\_in\_luma\_samples**, **chroma\_format\_vps\_idc**, **separate\_colour\_plane\_vps\_flag**, **bit\_depth\_vps\_luma\_minus8**, and **bit\_depth\_vps\_chroma\_minus8** are used for inference of the values of the SPS syntax elements pic\_width\_in\_luma\_samples, pic\_height\_in\_luma\_samples, chroma\_format\_idc, separate\_colour\_plane\_flag, bit\_depth\_luma\_minus8, and bit\_depth\_chroma\_minus8, respectively, for each SPS that refers to the VPS. When not present in the i-th rep\_format( ) syntax structure in the VPS, the value of each of these syntax elements is inferred to be equal to the value of the corresponding syntax element in the (i − 1)-th rep\_format( ) syntax structure in the VPS. For each of these syntax elements, all constraints, if any, that apply to the value of the corresponding SPS syntax element also apply. [Ed. (GT) Consider explicit constraints here.].

DPB size semantics

**sub\_layer\_flag\_info\_present\_flag**[ i ] equal to 1 specifies that sub\_layer\_dpb\_info\_present\_flag[ i ][ j ] is present for i in the range of 1 to vps\_max\_sub\_layers\_minus1, inclusive. sub\_layer\_flag\_info\_present\_flag[ i ] equal to 0 specifies that, for each value of j greater than 0, sub\_layer\_dpb\_info\_present\_flag[ i ][ j ] is not present and the value is inferred to be equal to 0.

**sub\_layer\_dpb\_info\_present\_flag**[ i ][ j ] equal to 1 specifies that max\_vps\_dec\_pic\_buffering\_minus1[ i ][ k ][ j ] is present for k in the range of 0 to NumSubDpbs[ i ] – 1, inclusive, for the j-th sub-layer, and max\_vps\_num\_reorder\_pics[ i ][ j ] and max\_vps\_latency\_increase\_plus1[ i ][ j ] are present for the j-th sub-layer. sub\_layer\_dpb\_info\_present\_flag[ i ][ j ] equal to 0 specifies that the values of max\_vps\_dec\_pic\_buffering\_minus1[ i ][ k ][ j ] are equal to max\_vps\_dec\_pic\_buffering\_minus1[ i ][ k ][ j − 1 ] for k in the range of 0 to NumSubDpbs[ i ] – 1, inclusive, and that the values max\_vps\_num\_reorder\_pics[ i ][ j ] and max\_vps\_latency\_increase\_plus1[ i ][ j ] are set equal to max\_vps\_num\_reorder\_pics[ i ][ j ‑ 1 ] and max\_vps\_latency\_increase\_plus1[ i ][ j ‑ 1 ], respectively. The value of sub\_layer\_dpb\_info\_present\_flag[ i ][ 0 ] for any possible value of i is inferred to be equal to 1.

**max\_vps\_dec\_pic\_buffering\_minus1**[ i ][ k ][ j ] plus 1 specifies the maximum required size of the k-th sub-DPB for the CVS in the i-th output layer set in units of picture storage buffers when HighestTid is equal to j. When j is greater than 0, max\_vps\_dec\_pic\_buffering\_minus1[ i ][ k ][ j ] shall be greater than or equal to max\_vps\_dec\_pic\_buffering\_minus1[ i ][ k ][ j ‑ 1 ]. When max\_vps\_dec\_pic\_buffering\_minus1[ i ][ k ][ j ] is not present for j in the range of 1 to vps\_max\_sub\_layers\_minus1 − 1, inclusive, it is inferred to be equal to max\_vps\_dec\_pic\_buffering\_minus1[ i ][ k ][ j ‑ 1].

**max\_vps\_num\_reorder\_pics**[ i ][ j ] specifies, when HighestTid is equal to j, the maximum allowed number of access units containing a picture with PicOutputFlag equal to 1 that can precede any access unit auA that contains a picture with PicOutputFlag equal to 1 in the i-th output layer set in the CVS in decoding order and follow the access unit auA that contains a picture with PicOutputFlag equal to 1 in output order. When max\_vps\_num\_reorder\_pics[ i ][ j ] is not present for j in the range of 1 to vps\_max\_sub\_layers\_minus1 – 1, inclusive, due to sub\_layer\_dpb\_info\_present\_flag[ i ][ j ] being equal to 0, it is inferred to be equal to max\_vps\_num\_reorder\_pics[ i ][ j ‑ 1].

**max\_vps\_latency\_increase\_plus1**[ i ][ j ] not equal to 0 is used to compute the value of VpsMaxLatencyPictures[ i ][ j ], which, when HighestTid is equal to j, specifies the maximum number of access units containing a picture with PicOutputFlag equal to 1 in the i-th output layer set that can precede any access unit auA that contains a picture with PicOutputFlag equal to 1 in the CVS in output order and follow the access unit auA that contains a picture with PicOutputFlag equal to 1 in decoding order. When max\_vps\_latency\_increase\_plus1[ i ][ j ] is not present for j in the range of 1 to vps\_max\_sub\_layers\_minus1 – 1, inclusive, due to sub\_layer\_dpb\_info\_present\_flag[ i ][ j ] being equal to 0, it is inferred to be equal to max\_vps\_latency\_increase\_plus1[ i ][ j ‑ 1 ].

When max\_vps\_latency\_increase\_plus1[ i ][ j ] is not equal to 0, the value of VpsMaxLatencyPictures[ i ][ j ] is specified as follows:

VpsMaxLatencyPictures[ i ][ j ] = max\_vps\_num\_reorder\_pics[ i ][ j ] +  
 max\_vps\_latency\_increase\_plus1[ i ][ j ] ‑ 1 (F‑4)

When max\_vps\_latency\_increase\_plus1[ i ][ j ] is equal to 0, no corresponding limit is expressed. The value of max\_vps\_latency\_increase\_plus1[ i ][ j ] shall be in the range of 0 to 232 – 2, inclusive.

VPS VUI semantics

**cross\_layer\_pic\_type\_aligned\_flag** equal to 1 specifies that within a CVS that refers to the VPS, all VCL NAL units that belong to an access unit have the same value of nal\_unit\_type. cross\_layer\_pic\_type\_aligned\_flag equal to 0 specifies that within a CVS that refers to the VPS, all VCL NAL units in each access unit may or may not have the same value of nal\_unit\_type.

**cross\_layer\_irap\_aligned\_flag** equal to 1 specifies that IRAP pictures in the CVS are cross-layer aligned, i.e. when a picture pictureA of a layer layerA in an access unit is an IRAP picture, each picture pictureB in the same access unit that belongs to a direct reference layer of layerA or that belongs to a layer for which layerA is a direct reference layer of that layer is an IRAP picture and the VCL NAL units of pictureB have the same value of nal\_unit\_type as that of pictureA. cross\_layer\_irap\_aligned\_flag equal to 0 specifies that the above restriction may or may not apply. When not present, the value of cross\_layer\_irap\_aligned\_flag is inferred to be equal to 1. [Ed. (JB): Need to change the inference to be based on value of cross\_layer\_pic\_type\_aligned\_flag, so that alignment is not inferred when the VPS VUI is not present.]

**bit\_rate\_present\_vps\_flag** equal to 1 specifies that the syntax element bit\_rate\_present\_flag[ i ][ j ] is present. bit\_rate\_present\_vps\_flag equal to 0 specifies that the syntax element bit\_rate\_present\_flag[ i ][ j ] is not present.

**pic\_rate\_present\_vps\_flag** equal to 1 specifies that the syntax element pic\_rate\_present\_flag[ i ][ j ] is present. pic\_rate\_present\_vps\_flag equal to 0 specifies that the syntax element pic\_rate\_present\_flag[ i ][ j ] is not present.

**bit\_rate\_present\_flag**[ i ][ j ] equal to 1 specifies that the bit rate information for the j-th subset of the i-th layer set is present. bit\_rate\_present\_flag[ i ] equal to 0 specifies that the bit rate information for the j-th subset of the i-th layer set is not present. The j-th subset of a layer set is the output of the sub-bitstream extraction process when it is invoked with the layer set, j, and the layer identifier list associated with the layer set as inputs. When not present, the value of bit\_rate\_present\_flag[ i ][ j ] is inferred to be equal to 0.

**pic\_rate\_present\_flag**[ i ][ j ] equal to 1 specifies that picture rate information for the j-th subset of the i-th layer set is present. pic\_rate\_present\_flag[ i ][ j ] equal to 0 specifies that picture rate information for the j-th subset of the i-th layer set is not present. When not present, the value of pic\_rate\_present\_flag[ i ][ j ] is inferred to be equal to 0.

**avg\_bit\_rate**[ i ][ j ] indicates the average bit rate of the j-th subset of the i-th layer set, in bits per second. The value is given by BitRateBPS( avg\_bit\_rate[ i ][ j ] ) with the function BitRateBPS( ) being specified as follows:

BitRateBPS( x ) = ( x & ( 214 − 1 ) ) \* 10( 2 + ( x >> 14 ) ) (F‑5)

The average bit rate is derived according to the access unit removal time specified in clause F.13. In the following, bTotal is the number of bits in all NAL units of the j-th subset of the i-th layer set, t1 is the removal time (in seconds) of the first access unit to which the VPS applies, and t2 is the removal time (in seconds) of the last access unit (in decoding order) to which the VPS applies. With x specifying the value of avg\_bit\_rate[ i ][ j ], the following applies:

– If t1 is not equal to t2, the following condition shall be true:

( x & ( 214 − 1 ) ) = = Round( bTotal ÷ ( ( t2 − t1 ) \* 10( 2 + ( x >> 14 ) ) ) ) (F‑6)

– Otherwise (t1 is equal to t2), the following condition shall be true:

( x & ( 214 − 1 ) ) = = 0 (F‑7)

**max\_bit\_rate\_layer**[ i ][ j ] indicates an upper bound for the bit rate of the j-th subset of the i-th layer set in any one-second time window of access unit removal time as specified in clause F.13. The upper bound for the bit rate in bits per second is given by BitRateBPS( max\_bit\_rate\_layer[ i ][ j ] ). The bit rate values are derived according to the access unit removal time specified in clause F.13. In the following, t1 is any point in time (in seconds), t2 is set equal to t1 + 1 ÷ 100, and bTotal is the number of bits in all NAL units of access units with a removal time greater than or equal to t1 and less than t2. With x specifying the value of max\_bit\_rate\_layer[ i ][ j ], the following condition shall be obeyed for all values of t1:

( x & ( 214 − 1 ) ) >= bTotal ÷ ( ( t2 − t1 ) \* 10( 2 + ( x >> 14 ) ) ) (F‑8)

**constant\_pic\_rate\_idc**[ i ][ j ] indicates whether the picture rate of the j-th subset of the i-th layer set is constant. In the following, a temporal segment tSeg is any set of two or more consecutive access units, in decoding order, of the j-th subset of the i-th layer set, auTotal( tSeg ) is the number of access units in the temporal segment tSeg, t1( tSeg ) is the removal time (in seconds) of the first access unit (in decoding order) of the temporal segment tSeg, t2( tSeg ) is the removal time (in seconds) of the last access unit (in decoding order) of the temporal segment tSeg, and avgPicRate( tSeg ) is the average picture rate in the temporal segment tSeg, and is specified as follows:

avgPicRate( tSeg ) = = Round( auTotal( tSeg ) \* 256 ÷ ( t2( tSeg ) − t1( tSeg ) ) ) (F‑9)

If the j-th subset of the i-th layer set only contains one or two access units or the value of avgPicRate( tSeg ) is constant over all the temporal segments, the picture rate is constant; otherwise, the picture rate is not constant.

constant\_pic\_rate\_idc[ i ][ j ] equal to 0 indicates that the picture rate of the j-th subset of the i-th layer set is not constant. constant\_pic\_rate\_idc[ i ][ j ] equal to 1 indicates that the picture rate of the j-th subset of the i-th layer set is constant. constant\_pic\_rate\_idc[ i ][ j ] equal to 2 indicates that the picture rate of the j-th subset of the i-th layer set may or may not be constant. The value of constant\_pic\_rate\_idc[ i ][ j ] shall be in the range of 0 to 2, inclusive.

**avg\_pic\_rate**[ i ] indicates the average picture rate, in units of picture per 256 seconds, of the j-th subset of the layer set. With auTotal being the number of access units in the j-th subset of the i-th layer set, t1 being the removal time (in seconds) of the first access unit to which the VPS applies, and t2 being the removal time (in seconds) of the last access unit (in decoding order) to which the VPS applies, the following applies:

– If t1 is not equal to t2, the following condition shall be true:

avg\_pic\_rate[ i ] = = Round( auTotal \* 256 ÷ ( t2 − t1 ) ) (F‑10)

– Otherwise (t1 is equal to t2), the following condition shall be true:

avg\_pic\_rate[ i ] = = 0 (F‑11)

**tiles\_not\_in\_use\_flag** equal to 1 indicates that the value of tiles\_enabled\_flag is equal to 0 for each PPS that is referred to by at least one picture referring to the VPS. tiles\_not\_in\_use\_flag equal to 0 indicates that such a restriction may or may not apply.

**tiles\_in\_use\_flag**[ i ] equal to 1 indicates that the value of tiles\_enabled\_flag is equal to 1 for each PPS that is referred to by at least one picture of the i-th layer specified by the VPS. tiles\_in\_use\_flag[ i ] equal to 0 indicates that such a restriction may or may not apply.

**loop\_filter\_not\_across\_tiles\_flag**[ i ] equal to 1 indicates that the value of loop\_filter\_across\_tiles\_enabled\_flag is equal to 0 for each PPS that is referred to by at least one picture of the i-th layer specified by the VPS. loop\_filter\_not\_across\_tiles\_flag[ i ] equal to 0 indicates that such a restriction may or may not apply.

**tile\_boundaries\_aligned\_flag**[ i ][ j ] equal to 1 indicates that, when any two samples of one picture of the i-th layer specified by the VPS belong to one tile, the two collocated samples, when both present in the picture of the j-th direct reference layer of the i-th layer, belong to one tile, and when any two samples of one picture of the i-th layer belong to different tiles, the two collocated samples, when both present in the picture of the j-th direct reference layer of the i-th layer belong to different tiles. tile\_boundaries\_aligned\_flag equal to 0 indicates that such a restriction may or may not apply. When not present, the value of tile\_boundaries\_aligned\_flag[ i ][ j ] is inferred to be equal to 0.

**wpp\_not\_in\_use\_flag** equal to 1 indicates that the value of entropy\_coding\_sync\_enabled\_flag is equal to 0 for each PPS that is referred to by at least one picture referring to the VPS. wpp\_not\_in\_use\_flag equal to 0 indicates that such a restriction may or may not apply.

**wpp\_in\_use\_flag**[ i ] equal to 1 indicates that the value of entropy\_coding\_sync\_enabled\_flag is equal to 1 for each PPS that is referred to by at least one picture of the i-th layer specified by the VPS. wpp\_in\_use\_flag[ i ] equal to 0 indicates that such a restriction may or may not apply.

[Ed. (YK): Define "collocated sample".]

**single\_layer\_for\_non\_irap\_flag** equal to 1 indicates either that all the VCL NAL units of an access unit have the same nuh\_layer\_id value or that two nuh\_layer\_id values are used by the VCL NAL units of an access unit and the picture with the greater nuh\_layer\_id value is an IRAP picture. single\_layer\_for\_non\_irap\_flag equal to 0 indicates that nuh\_layer\_id values may or may not be constrained beyond constraints specified in other parts of this Recommendation | International Standard. When single\_layer\_for\_non\_irap\_flag is not present, it is inferred to be equal to 0.

**higher\_layer\_irap\_skip\_flag** equal to 1 indicates that for every IRAP picture that refer to the VPS, for which there is another picture in the same access unit with a lower value of nuh\_layer\_id, the following constraints apply:

– For all slices of the IRAP picture:

* slice\_type shall be equal to P.
* slice\_sao\_luma\_flag and slice\_sao\_chroma\_flag shall both be equal to 0.
* five\_minus\_max\_num\_merge\_cand shall be equal to 4.
* weighted\_pred\_flag shall be equal to 0 in the PPS that is refered to by the slices.

– For all coding units of the IRAP picture:

* cu\_skip\_flag[ i ][ j ] shall be equal to 1.

higher\_layer\_irap\_skip\_flag equal to 0 indicates that the above constraints may or may not apply.

When single\_layer\_for\_non\_irap\_flag is equal to 0, higher\_layer\_irap\_skip\_flag shall be equal to 0. When higher\_layer\_irap\_skip\_flag is not present it is inferred to be equal to 0.

NOTE 2 – An encoder may set both single\_layer\_for\_non\_irap\_flag and higher\_layer\_irap\_skip\_flag equal to 1 as an indication to a decoder that whenever there are two pictures in the same access unit, the one with the higher nuh\_layer\_id is an IRAP picture for which the decoded samples can be derived by applying the resampling process for inter layer reference pictures specified in subclause H.8.1.4 with the other picture as input.

**ilp\_restricted\_ref\_layers\_flag** equal to 1 indicates that additional restrictions on inter-layer prediction as specified below apply for each direct reference layer of each layer specified by the VPS. ilp\_restricted\_ref\_layers\_flag equal to 0 indicates that additional restrictions on inter-layer prediction may or may not apply.

[Ed. (YK): Consider using better syntax element names for min\_spatial\_segment\_offset\_plus1[ i ][ j ], ctu\_based\_offset\_enabled\_flag[ i ][ j ], and min\_horizontal\_ctu\_offset\_plus1[ i ][ j ].]

The variables refCtbLog2SizeY[ i ][ j ], refPicWidthInCtbsY[ i ][ j ], and refPicHeightInCtbsY[ i ][ j ] are set equal to CtbLog2SizeY, PicWidthInCtbsY, and PicHeightInCtbsY, respectively, of the j-th direct reference layer of the i-th layer.

**min\_spatial\_segment\_offset\_plus1**[ i ][ j ] indicates the spatial region, in each picture of the j-th direct reference layer of the i-th layer, that is not used for inter-layer prediction for decoding of any picture of the i-th layer, by itself or together with min\_horizontal\_ctu\_offset\_plus1[ i ][ j ], as specified below. The value of min\_spatial\_segment\_offset\_plus1[ i ][ j ] shall be in the range of 0 to refPicWidthInCtbsY[ i ][ j ] \* refPicHeightInCtbsY[ i ][ j ], inclusive. When not present, the value of min\_spatial\_segment\_offset\_plus1[ i ][ j ] is inferred to be equal to 0.

**ctu\_based\_offset\_enabled\_flag**[ i ][ j ] equal to 1 specifies that the spatial region, in units of CTUs, in each picture of the j-th direct reference layer of the i-th layer, that is not used for inter-layer prediction for decoding of any picture of the i-th layer is indicated by min\_spatial\_segment\_offset\_plus1[ i ][ j ] and min\_horizontal\_ctu\_offset\_plus1[ i ][ j ] together. ctu\_based\_offset\_enabled\_flag[ i ][ j ] equal to 0 specifies that the spatial region, in units of slice segments, tiles, or CTU rows, in each picture of the j-th direct reference layer of the i-th layer, that is not used for inter-layer prediction for decoding of any picture of the i-th layer is indicated by min\_spatial\_segment\_offset\_plus1[ i ] only. When not present, the value of ctu\_based\_offset\_enabled\_flag[ i ] is inferred to be equal to 0.

**min\_horizontal\_ctu\_offset\_plus1**[ i ][ j ], when ctu\_based\_offset\_enabled\_flag[ i ][ j ] is equal to 1, indicates the spatial region, in each picture of the j-th direct reference layer of the i-th layer, that is not used for inter-layer prediction for decoding of any picture of the i-th layer, together with min\_spatial\_segment\_offset\_plus1[ i ][ j ], as specified below. The value of min\_horizontal\_ctu\_offset\_plus1[ i ][ j ] shall be in the range of 0 to refPicWidthInCtbsY[ i ][ j ], inclusive.

When ctu\_based\_offset\_enabled\_flag[ i ][ j ] is equal to 1, the variable minHorizontalCtbOffset[ i ][ j ] is derived as follows:

minHorizontalCtbOffset[ i ][ j ] = ( min\_horizontal\_ctu\_offset\_plus1[ i ][ j ] > 0 ) ? (F‑12)  
 ( min\_horizontal\_ctu\_offset\_plus1[ i ][ j ] – 1 ) : ( refPicWidthInCtbsY[ i ][ j ] ‑ 1 )

The variables curPicWidthInSamplesL[ i ], curPicHeightInSamplesL[ i ], curCtbLog2SizeY[ i ], curPicWidthInCtbsY[ i ], and curPicHeightInCtbsY[ i ] are set equal to PicWidthInSamplesL, PicHeightInSamplesL, CtbLog2SizeY, PicWidthInCtbsY, and PicHeightInCtbsY, respectively, of the i-th layer.

The variables refPicWidthInSamplesL[ i ][ j ], refPicHeightInSamplesL[ i ][ j ], refCtbLog2SizeY[ i ][ j ], refPicWidthInCtbsY[ i ][ j ], and refPicHeightInCtbsY[ i ][ j ] are set equal to PicWidthInSamplesL, PicHeightInSamplesL, CtbLog2SizeY, PicWidthInCtbsY, and PicHeightInCtbsY, respectively, of the j-th direct reference layer of the i-th layer.

The variables curScaledRefLayerLeftOffset[ i ][ j ], curScaledRefLayerTopOffset[ i ][ j ], curScaledRefLayerRightOffset[ i ][ j ] and curScaledRefLayerBottomOffset[ i ][ j ] are set equal to scaled\_ref\_layer\_left\_offset[ j ]<<1, scaled\_ref\_layer\_top\_offset[ j ]<<1, scaled\_ref\_layer\_right\_offset[ j ]<<1, scaled\_ref\_layer\_bottom\_offset [ j ]<<1, respectively, of the j-th direct reference layer of the i-th layer.

The variable colCtbAddr[ i ][ j ] that denotes the raster scan address of the collocated CTU, in a picture in the j-th direct reference layer of the i-th layer, of the CTU with raster scan address equal to ctbAddr in a picture of the i-th layer is derived as follows [Ed. (YK): Define "collocated CTU".]:

– The variables ( xP, yP ) specifying the location of the top-left luma sample of the CTU with raster scan address equal to ctbAddr relative to top-left luma luma sample in a picture of the i-th layer are derived as follows:

xP = ( ctbAddr % curPicWidthInCtbsY )  <<  curCtbLog2SizeY (F‑13)

yP = ( ctbAddr / curPicWidthInCtbsY )  <<  curCtbLog2SizeY (F‑14)

– The variables scaleFactorX[ i ][ j ] and scaleFactorY[ i ][ j ] are derived as follows:

curScaledRefLayerPicWidthInSamplesL[ i ][ j ] = curPicWidthInSamplesL[ i ]  –   
 curScaledRefLayerLeftOffset[ i ][ j ] – curScaledRefLayerRightOffset[ i ][ j ] (F‑15)

curScaledRefLayerPicHeightInSamplesL[ i ][ j ] = curPicHeightInSamplesL[ i ]  –   
 curScaledRefLayerTopOffset[ i ][ j ] – curScaledRefLayerBottomOffset[ i ][ j ] (F‑16)

scaleFactorX[ i ][ j ] = ( ( refPicWidthInSamplesL [ i ][ j ]  << 16 ) +   
( curScaledRefLayerPicWidthInSamplesL [ i ][ j ]>> 1 ) )/curScaledRefLayerPicWidthInSamplesL [ i ][ j ] (F‑17)

scaleFactorY[ i ][ j ] = ( ( refPicHeightInSamplesL [ i ][ j ] << 16 ) +   
( curScaledRefLayerPicHeightInSamplesL >> 1 ) ) / curScaledRefLayerPicHeightInSamplesL [ i ][ j ] (F‑18)

[Ed. (JC): the global variables related to scaling factor were already defined in Annex H. It’s desired to move the definition to Annex F, e.g. at the semantics part of related syntax, so that the depulicated derivation process here can be deleted]

– The variables ( xCol[ I ][ j ], yCol xCol[ I ][ j ]) specifying the collocated luma sample location in a picture in the j-th direct reference layer of the luma sample location ( xP, yP ) in the i-th layer are derived as follows:

xCol[ i ][ j ] = Clip3( 0, ( refPicWidthInSamplesL[ i ][ j ]– 1 ), ( ( xP ‑ curScaledRefLayerLeftOffset[ i ][ j ]) \* scaleFactorX[ i ][ j ] + ( 1 << 15 ) ) >> 16)) (F‑19)

yCol[ i ][ j ] = Clip3( 0 , ( refPicHeightInSamplesL[ i ][ j ]– 1 ), ( ( yP ‑ curScaledRefLayerTopOffset[ i ][ j ]) \* scaleFactorY[ i ][ j ] + ( 1 << 15 ) ) >> 16)) (F‑20)

– The variable colCtbAddr[ i ][ j ] is derived as follows:

xColCtb[ i ][ j ] = xCol[ i ][ j ]  >>  refCtbLog2SizeY[ i ][ j ] (F‑21)

yColCtb[ i ][ j ] = yCol[ i ][ j ]  >>  refCtbLog2SizeY[ i ][ j ] (F‑22)

colCtbAddr[ i ][ j ] = xColCtb[ i ][ j ] + ( yColCtb[ i ][ j ] \* refPicWidthInCtbsY[ i ][ j ] ) (F‑23)

When min\_spatial\_segment\_offset\_plus1[ i ][ j ] is greater than 0, it is a requirement of bitstream conformance that the following shall apply:

* If ctu\_based\_offset\_enabled\_flag[ i ][ j ] is equal to 0, exactly one of the following applies:
* In each PPS referred to by a picture in the j-th direct reference layer of the i-th layer, tiles\_enabled\_flag is equal to 0 and entropy\_coding\_sync\_enabled\_flag is equal to 0, and the following applies:
* Let slice segment A be any slice segment of a picture of the i-th layer and ctbAddr be the raster scan address of the last CTU in slice segment A. Let slice segment B be the slice segment that belongs to the same access unit as slice segment A, belongs to the j-th direct reference layer of the i-th layer, and contains the CTU with raster scan address colCtbAddr[ i ][ j ]. Let slice segment C be the slice segment that is in the same picture as slice segment B and follows slice segment B in decoding order, and between slice segment B and that slice segment there are min\_spatial\_segment\_offset\_plus1[ i ] ‑ 1 slice segments in decoding order. When slice segment C is present, the syntax elements of slice segment A are constrained such that no sample or syntax elements values in slice segment C or any slice segment of the same picture following C in decoding order are used for inter-layer prediction in the decoding process of any samples within slice segment A.
* In each PPS referred to by a picture in the j-th direct reference layer of the i-th layer, tiles\_enabled\_flag is equal to 1 and entropy\_coding\_sync\_enabled\_flag is equal to 0, and the following applies:
* Let tile A be any tile in any picture picA of the i-th layer and ctbAddr be the raster scan address of the last CTU in tile A. Let tile B be the tile that is in the picture picB belonging to the same access unit as picA and belonging to the j-th direct reference layer of the i-th layer and that contains the CTU with raster scan address colCtbAddr[ i ][ j ]. Let tile C be the tile that is also in picB and follows tile B in decoding order, and between tile B and that tile there are min\_spatial\_segment\_offset\_plus1[ i ] ‑ 1 tiles in decoding order. When slice segment C is present, the syntax elements of tile A are constrained such that no sample or syntax elements values in tile C or any tile of the same picture following C in decoding order are used for inter-layer prediction in the decoding process of any samples within tile A.
* In each PPS referred to by a picture in the j-th direct reference layer of the i-th layer, tiles\_enabled\_flag is equal to 0 and entropy\_coding\_sync\_enabled\_flag is equal to 1, and the following applies:
* Let CTU row A be any CTU row in any picture picA of the i-th layer and ctbAddr be the raster scan address of the last CTU in CTU row A. Let CTU row B be the CTU row that is in the picture picB belonging to the same access unit as picA and belonging to the j-th direct reference layer of the i-th layer and that contains the CTU with raster scan address colCtbAddr[ i ][ j ]. Let CTU row C be the CTU row that is also in picB and follows CTU row B in decoding order, and between CTU row B and that CTU row there are min\_spatial\_segment\_offset\_plus1[ i ] ‑ 1 CTU rows in decoding order. When CTU row C is present, the syntax elements of CTU row A are constrained such that no sample or syntax elements values in CTU row C or row of the same picture following C are used for inter-layer prediction in the decoding process of any samples within CTU row A.
* Otherwise (ctu\_based\_offset\_enabled\_flag[ i ][ j ] is equal to 1), the following applies:
* The variable refCtbAddr[ i ][ j ] is derived as follows:

xOffset[ i ][ j ] = ( ( xColCtb[ i ][ j ] + minHorizontalCtbOffset[ i ][ j ] ) > ( refPicWidthInCtbsY[ i ][ j ] ‑ 1 ) ) ?   
( refPicWidthInCtbsY[ i ][ j ] ‑ 1 –xColCtb[ i ][ j ] ) : ( minHorizontalCtbOffset[ i ][ j ] ) (F‑24)

yOffset[ i ][ j ] = ( min\_spatial\_segment\_offset\_plus1[ i ][ j ] – 1 ) \* refPicWidthInCtbsY[ i ][ j ] (F‑25)

refCtbAddr[ i ][ j ] = colCtbAddr[ i ][ j ] + xOffset[ i ][ j ] + yOffset[ i ][ j ] (F‑26)

* Let CTU A be any CTU in any picture picA of the i-th layer, and ctbAddr be the raster scan address ctbAddr of CTU A. Let CTU B be a CTU that is in the picture belonging to the same access unit as picA and belonging to the j-th direct reference layer of the i-th layer and that has raster scan address greater than refCtbAddr[ i ][ j ]. When CTU B is present, the syntax elements of CTU A are constrained such that no sample or syntax elements values in CTU B are used for inter-layer prediction in the decoding process of any samples within CTU A.

**video\_signal\_info\_idx\_present\_flag** equal to 1 specifies that the syntax elements vps\_num\_video\_signal\_info\_minus1, and vps\_video\_signal\_info\_idx[ i ] are present. video\_signal\_info\_idx\_present\_flag equal to 0 specifies that the syntax elements vps\_num\_video\_signal\_info\_minus1, and vps\_video\_signal\_info\_idx[ i ] are not present.

**vps\_num\_video\_signal\_info\_minus1** plus 1 specifies the number of the following video\_signal\_info( ) syntax structures in the VPS. When not present, the value of vps\_num\_video\_signal\_info\_minus1 is inferred to be equal to MaxLayersMinus1.

**vps\_video\_signal\_info\_idx**[ i ] specifies the index, into the list of video\_signal\_info( ) syntax structures in the VPS, of the video\_signal\_info( ) syntax structure that applies to the layer with nuh\_layer\_id equal to layer\_id\_in\_nuh[ i ]. When vps\_video\_signal\_info\_idx[ i ] is not present, vps\_video\_signal\_info\_idx[ i ] is inferred to be equal to ( video\_signal\_info\_idx\_present\_flag ? 0 : i ). The value of vps\_video\_signal\_info\_idx[ i ] shall be in the range of 0 to vps\_num\_video\_signal\_info\_minus1, inclusive.

**vps\_vui\_bsp\_hrd\_present\_flag** equal to 0 specifies that no bitstream partition HRD parameters are present in the VPS VUI. vps\_vui\_bsp\_hrd\_present\_flag equal to 1 specifies that bitstream partition HRD parameters are present in the VPS VUI.

Video signal info semantics

**video\_vps\_format**, **video\_full\_range\_vps\_flag**, **colour\_primaries\_vps**, **transfer\_characteristics\_vps**, **matrix\_coeffs\_vps** are used for inference of the values of the SPS VUI syntax elements video\_format, video\_full\_range\_flag, colour\_primaries, transfer\_characteristics, matrix\_coeffs respectively, for each SPS that refers to the VPS.

For each of these syntax elements, all constraints, if any, that apply to the value of the corresponding SPS VUI syntax element also apply.

VPS VUI bitstream partition HRD parameters semantics

**vps\_num\_bsp\_hrd\_parameters\_minus1** plus 1 specifies the number of hrd\_parameters( ) syntax structures present within the vps\_vui\_bsp\_hrd\_parameters( ) syntax structure.

**bsp\_cprms\_present\_flag**[ i ] equal to 1 specifies that the HRD parameters that are common for all sub-layers are present in the i-th hrd\_parameters( ) syntax structure in the vps\_vui\_bsp\_hrd\_parameters( ) syntax structure. bsp\_cprms\_present\_flag[ i ] equal to 0 specifies that the HRD parameters that are common for all sub-layers are not present in the i-th hrd\_parameters( ) syntax structure in the vps\_vui\_bsp\_hrd\_parameters( ) syntax structure and are derived to be the same as the ( i − 1 )-th hrd\_parameters( ) syntax structure in the in the vps\_vui\_bsp\_hrd\_parameters( ) syntax structure. bsp\_cprms\_present\_flag[ 0 ] is inferred to be equal to 1.

**num\_bitstream\_partitions**[ h ] specifies the number of bitstream partitions for which HRD parameters are specified for the layer set with index h.

**layer\_in\_bsp\_flag**[ h ][ i ][ j ] specifies that the layer with index j is a part of bitstream partition with index i within the layer set with index h.

It is a requirement of bitstream conformance that bitstream partition with index j shall not include direct or indirect reference layers of any layers in bitstream partition i for any values of i and j in the range of 0 to num\_bitstream\_partitions[ h ] – 1, inclusive, such that i is less than j.

**num\_bsp\_sched\_combinations**[ h ] specifies the number of combinations of delivery schedules and hrd\_parameters( ) specified for bitstream partitions for the layer set with index h.

**bsp\_comb\_hrd\_idx**[ h ][ i ][ j ] specifies the index of hrd\_parameters( ) within the vps\_vui\_bsp\_hrd\_parameters( ) syntax structure used in the i-th combination of a delivery schedule and hrd\_parameters( ) specified for the bitstream partition with index j and for the layer set with index h.

**bsp\_comb\_sched\_idx**[ h ][ i ][ j ] specifies the index of a delivery schedule within the hrd\_parameters( ) syntax structure with the index bsp\_comb\_hrd\_idx[ h ][ i ][ j ] that is used in the i-th combination of a delivery schedule and hrd\_parameters( ) specified for the bitstream partition with index j and for the layer set with index h.

* + - * 1. Sequence parameter set RBSP semantics

The specifications in subclause 7.4.3.2 apply, with following additions and modifications.

NOTE 1 – All SPSs, regardless of the values of their nuh\_layer\_id, share the same value space for sps\_seq\_parameter\_set\_id. In other words, an SPS with nuh\_layer\_id equal to X and sps\_seq\_parameter\_set\_id equal to A would update the previously received SPS with nuh\_layer\_id not equal to X and sps\_seq\_parameter\_set\_id equal to A.

**sps\_max\_sub\_layers\_minus1** plus 1 specifies the maximum number of temporal sub-layers that may be present in each CVS referring to the SPS. The value of sps\_max\_sub\_layers\_minus1 shall be in the range of 0 to 6, inclusive. When not present sps\_max\_sub\_layers\_minus1 is inferred to be equal to vps\_max\_sub\_layers\_minus1.

**update\_rep\_format\_flag** equal to 1 specifies that sps\_rep\_format\_idx is present and that the sps\_rep\_format\_idx-th rep\_format( ) syntax structures in the active VPS applies to the layers that refer to this SPS. update\_rep\_format\_flag equal to 0 specifies that sps\_rep\_format\_idx is not present. When not present, the value of update\_rep\_format\_flag is inferred to be equal to 0.

**sps\_rep\_format\_idx** specifies the index, into the list of rep\_format( ) syntax structures in the VPS, of the rep\_format( ) syntax structure that applies to the layers that refer to this SPS. When not present, the value of sps\_rep\_format\_idx is inferred to be equal to 0. The value of sps\_rep\_format\_idx shall be in the range of 0 to vps\_num\_rep\_formats\_minus1, inclusive. [Ed. (GT): Inferences to 0 seems not to be necessary. We might consider to infer it to vps\_rep\_format\_idx[ LayerIdxInVps[ layerIdCurr ] ], when not present. ].

When a current picture with nuh\_layer\_id layerIdCurr greater than 0 refers to an SPS, the values of chroma\_format\_idc, separate\_colour\_plane\_flag, pic\_width\_in\_luma\_samples, pic\_height\_in\_luma\_samples, bit\_depth\_luma\_minus8, and bit\_depth\_chroma\_minus8 are inferred or constrained as follows:

– The variable repFormatIdx is derived as follows:

– If update\_rep\_format\_flag is equal to 0, the variable repFormatIdx is set equal to vps\_rep\_format\_idx[ LayerIdxInVps[ layerIdCurr ] ].

– Otherwise, (update\_rep\_format\_flag is equal to 1), the variable repFormatIdx is set equal to sps\_rep\_format\_idx.

– If the nuh\_layer\_id of the active SPS for the layer with nuh\_layer\_id equal to layerIdCurr is equal to 0, the values of chroma\_format\_idc, separate\_colour\_plane\_flag, pic\_width\_in\_luma\_samples, pic\_height\_in\_luma\_samples, bit\_depth\_luma\_minus8, and bit\_depth\_chroma\_minus8 are inferred to be equal to chroma\_format\_vps\_idc, separate\_colour\_plane\_vps\_flag, pic\_width\_vps\_in\_luma\_samples, pic\_height\_vps\_in\_luma\_samples, bit\_depth\_vps\_luma\_minus8, and bit\_depth\_vps\_chroma\_minus8, respectively, of the repFormatIdx-th rep\_format( ) syntax structure in the active VPS and the values of chroma\_format\_idc, separate\_colour\_plane\_flag, pic\_width\_in\_luma\_samples, pic\_height\_in\_luma\_samples, bit\_depth\_luma\_minus8, and bit\_depth\_chroma\_minus8 of the active SPS for the layer with nuh\_layer\_id equal to layerIdCurr are ignored.

NOTE 2 – The values are inferred from the VPS when a non-base layer refers to an SPS that is also referred to by the base layer, in which case the SPS has nuh\_layer\_id equal to 0. For the base layer, the values of these parameters in the active SPS for the base layer apply.

– Otherwise (the nuh\_layer\_id of the active SPS for the layer with nuh\_layer\_id equal to layerIdCurr is greater than zero), the following applies:

– The values of chroma\_format\_idc, separate\_colour\_plane\_flag, pic\_width\_in\_luma\_samples, pic\_height\_in\_luma\_samples, bit\_depth\_luma\_minus8, and bit\_depth\_chroma\_minus8 are inferred to be equal to chroma\_format\_vps\_idc, separate\_colour\_plane\_vps\_flag, pic\_width\_vps\_in\_luma\_samples, pic\_height\_vps\_in\_luma\_samples, bit\_depth\_vps\_luma\_minus8, and bit\_depth\_vps\_chroma\_minus8, respectively, of the repFormatIdx-th rep\_format( ) syntax structure in the active VPS.

– When update\_rep\_format\_flag is equal to 1, it is a requirement of bitstream conformance that the value of chroma\_format\_idc, separate\_colour\_plane\_flag, pic\_width\_in\_luma\_samples, pic\_height\_in\_luma\_samples, bit\_depth\_luma\_minus8, or bit\_depth\_chroma\_minus8 shall be less than or equal to chroma\_format\_vps\_idc, separate\_colour\_plane\_vps\_flag, pic\_width\_vps\_in\_luma\_samples, pic\_height\_vps\_in\_luma\_samples, bit\_depth\_vps\_luma\_minus8, or bit\_depth\_vps\_chroma\_minus8, respectively, of the vps\_rep\_format\_idx[ j ]-th rep\_format( ) syntax structure in the active VPS, where j is equal to LayerIdxInVps[ layerIdCurr ].

**chroma\_format\_idc** specifies the chroma sampling relative to the luma sampling as specified in subclause 6.2. The value of chroma\_format\_idc shall be in the range of 0 to 3, inclusive. The value of chroma\_format\_idc shall be less than or equal to chroma\_format\_vps\_idc. [ Ed. (GT): These requirements seem to be redundant now. We should consider to remove them. ]

It is a requirement of bitstream conformance that when AuxId[ lId ] is equal to AUX\_ALPHA or AUX\_DEPTH, chroma\_format\_idc shall be equal to 0 in the active SPS for the layer with nuh\_layer\_id equal to lId.

**separate\_colour\_plane\_flag** equal to 1 specifies that the three colour components of the 4:4:4 chroma format are coded separately. separate\_colour\_plane\_flag equal to 0 specifies that the colour components are not coded separately. When separate\_colour\_plane\_flag is not present, it is inferred to be equal to 0. When separate\_colour\_plane\_flag is equal to 1, the coded picture consists of three separate components, each of which consists of coded samples of one colour plane (Y, Cb, or Cr) and uses the monochrome coding syntax. In this case, each colour plane is associated with a specific colour\_plane\_id value. The value of separate\_colour\_plane\_flag shall be less than or equal to separate\_colour\_plane\_vps\_flag

NOTE 3 – There is no dependency in decoding processes between the colour planes having different colour\_plane\_id values. For example, the decoding process of a monochrome picture with one value of colour\_plane\_id does not use any data from monochrome pictures having different values of colour\_plane\_id for inter prediction.

Depending on the value of separate\_colour\_plane\_flag, the value of the variable ChromaArrayType is assigned as follows:

– If separate\_colour\_plane\_flag is equal to 0, ChromaArrayType is set equal to chroma\_format\_idc.

– Otherwise (separate\_colour\_plane\_flag is equal to 1), ChromaArrayType is set equal to 0.

**pic\_width\_in\_luma\_samples** specifies the width of each decoded picture in units of luma samples. pic\_width\_in\_luma\_samples shall not be equal to 0 and shall be an integer multiple of MinCbSizeY. The value of pic\_width\_in\_luma\_samples shall be less than or equal to pic\_width\_vps\_in\_luma\_samples.

**pic\_height\_in\_luma\_samples** specifies the height of each decoded picture in units of luma samples. pic\_height\_in\_luma\_samples shall not be equal to 0 and shall be an integer multiple of MinCbSizeY. The value of pic\_height\_in\_luma\_samples shall be less than or equal to pic\_height\_vps\_in\_luma\_samples.

**bit\_depth\_luma\_minus8** specifies the bit depth of the samples of the luma array BitDepthY and the value of the luma quantization parameter range offset QpBdOffsetY as follows:

BitDepthY = 8 + bit\_depth\_luma\_minus8 (F‑27)

QpBdOffsetY = 6 \* bit\_depth\_luma\_minus8 (F‑28)

bit\_depth\_luma\_minus8 shall be in the range of 0 to 6, inclusive. bit\_depth\_luma\_minus8 shall be less than or equal to bit\_depth\_vps\_luma\_minus8.

**bit\_depth\_chroma\_minus8** specifies the bit depth of the samples of the chroma arrays BitDepthC and the value of the chroma quantization parameter range offset QpBdOffsetC as follows:

BitDepthC = 8 + bit\_depth\_chroma\_minus8 (F‑29)

QpBdOffsetC = 6 \* bit\_depth\_chroma\_minus8 (F‑30)

bit\_depth\_chroma\_minus8 shall be in the range of 0 to 6, inclusive. bit\_depth\_chroma\_minus8 shall be less than or equal to bit\_depth\_vps\_chroma\_minus8.

**sps\_infer\_scaling\_list\_flag** equal to 1 specifies that the syntax elements of the scaling list data syntax structure of the SPS are inferred to be equal to those of the SPS that is active for the layer with nuh\_layer\_id equal to sps\_scaling\_list\_ref\_layer\_id. sps\_infer\_scaling\_list\_flag equal to 0 specifies that the syntax elements of the scaling list data syntax structure are not inferred. When not present, the value of sps\_infer\_scaling\_list\_flag is inferred to be 0.

**sps\_scaling\_list\_ref\_layer\_id** specifies the value of the nuh\_layer\_id of the layer for which the active SPS is associated with the same scaling list data as the current SPS.

The value of sps\_scaling\_list\_ref\_layer\_id shall be in the range of 0 to 62, inclusive.

When avc\_base\_layer\_flag is equal to 1, it is a requirement of bitstream conformance that the value of sps\_scaling\_list\_ref\_layer\_id shall be greater than 0.

It is a requirement of bitstream conformance that, when an SPS with nuh\_layer\_id equal to nuhLayerIdA is active for a layer with nuh\_layer\_id equal to nuhLayerIdB and sps\_infer\_scaling\_list\_flag in the SPS is equal to 1, sps\_infer\_scaling\_list\_flag shall be equal to 0 for the SPS that is active for the layer with nuh\_layer\_id equal to sps\_scaling\_list\_ref\_layer\_id. [Ed. (YK): This constraint is not necessarily needed. It would be nice to allow for all SPSs recursively infer the scaling list data from the lowest HEVC layer, when desirable, as that does not impose any additional decoder complexity anyway.]

It is a requirement of bitstream conformance that, when an SPS with nuh\_layer\_id equal to nuhLayerIdA is active for a layer with nuh\_layer\_id equal to nuhLayerIdB, the layer with nuh\_layer\_id equal to sps\_scaling\_list\_ref\_layer\_id shall be a direct or indirect reference layer of the layer with nuh\_layer\_id equal to nuhLayerIdB.

**sps\_scaling\_list\_data\_present\_flag** equal to 1 specifies that the scaling list data syntax structure is present in the SPS. sps\_scaling\_list\_data\_present\_flag equal to 0 specifies that the scaling list data syntax structure is not present in the SPS. When not present, the value of sps\_scaling\_list\_data\_present\_flag is inferred to be equal to 0.

**sps\_temporal\_id\_nesting\_flag**, when sps\_max\_sub\_layers\_minus1 is greater than 0, specifies whether inter prediction is additionally restricted for CVSs referring to the SPS. When vps\_temporal\_id\_nesting\_flag is equal to 1, sps\_temporal\_id\_nesting\_flag shall be equal to 1. When sps\_max\_sub\_layers\_minus1 is equal to 0, sps\_temporal\_id\_nesting\_flag shall be equal to 1. When not present sps\_temporal\_id\_nesting\_flag is inferred to be equal to vps\_temporal\_id\_nesting\_flag.

NOTE 4 – The syntax element sps\_temporal\_id\_nesting\_flag is used to indicate that temporal up-switching, i.e. switching from decoding up to any TemporalId tIdN to decoding up to any TemporalId tIdM that is greater than tIdN, is always possible in the CVS.

**sps\_extension\_flag** equal to 1 specifies that sps\_extension\_type\_flag[ i ] for i in the range of 0 to 7, inclusive are present in the SPS RBSP syntax structure. sps\_extension\_flag equal to 0 specifies that sps\_extension\_flag[ i ] for i in the range of 0 to 7, inclusive are not present in the SPS RBSP syntax structure.

**sps\_extension\_type\_flag**[ i ] shall be equal to 0, for i equal to 0 and in the range of 2 to 6, inclusive, in bitstreams conforming to this version of this Specification. The value of 1 for sps\_extension\_type\_flag[ i ], for i equal to 0 and in the range of 2 to 6, inclusive, is reserved for future use by ITU-T | ISO/IEC. sps\_extension\_type\_flag[ 1 ] equal to 1 specifies that the sps\_multilayer\_extension syntax structure is present. sps\_extension\_type\_flag[ 1 ] equal to 0 specifies that the sps\_multilayer\_extension syntax structure is not present. sps\_extension\_type\_flag[ 7 ] equal to 0 specifies that no sps\_extension\_data\_flag syntax elements are present in the SPS RBSP syntax structure. sps\_extension\_type\_flag[ 7 ] shall be equal to 0 in bitstreams conforming to this version of this Specification. The value of 1 for sps\_extension\_type\_flag[ 7 ] is reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all sps\_extension\_data\_flag syntax elements that follow the value 1 for sps\_extension\_type\_flag[ 7 ] in an SPS NAL unit.

[Ed. (GT) constraints on sps\_extension\_type\_flag for i equal to 0 and in the range of 2 to 6 should be removed when semantics are moved to the base spec ]

Sequence parameter set multilayer extension semantics

**inter\_view\_mv\_vert\_constraint\_flag** equal to 1 specifies that vertical component of motion vectors used for inter-layer prediction are constrained in the CVS. When inter\_view\_mv\_vert\_constraint\_flag is equal to 1, the vertical component of the motion vectors used for inter-layer prediction shall be equal to or less than 56 in units of luma samples. When inter\_view\_mv\_vert\_constraint\_flag is equal to 0, no constraint for of the vertical component of the motion vectors used for inter-layer prediction is signalled by this flag. When not present, the inter\_view\_mv\_vert\_constraint\_flag is inferred to be equal to 0.

**sps\_ref\_layer\_rps\_enabled\_flag** equal to 1 specifies that one or more ref\_layer\_rps\_present\_flag[ i ] syntax elements are present in slice headers of pictures with nuh\_layer\_id equal to nuhLayerId such that NumDirectRefLayers[ nuhLayerId ] is greater than 0. sps\_ref\_layer\_rps\_enabled\_flag equal to 0 specifies that reference-layer short-term reference picture sets are not included in slice headers for which this SPS is the active SPS. When not present, the value of sps\_ref\_layer\_rps\_enabled\_flag is inferred to be equal to 0.

**num\_scaled\_ref\_layer\_offsets** specifies the number of sets of scaled reference layer offset parameters that are present in the SPS. The value of num\_scaled\_ref\_layer\_offsets shall be in the range of 0 to 62, inclusive. [Ed. (JB): Should consider if this constraint should be further restricted. Is there a limit on the number of direct reference layers? (MH): If that is desirable, we should specify the range like this: "in the range of 0 to highestActiveLayerId, inclusive, where the variable highestActiveLayerId is equal to the greatest value of nuh\_layer\_id of any picture for which this SPS is the active SPS".]

The i-th scaled reference layer offset parameters specify the spatial correspondence of a picture referring to this SPS relative to an associated inter-layer picture with nuh\_layer\_id equal to scaled\_ref\_layer\_id[ i ]. If the layer with nuh\_layer\_id equal to scaled\_ref\_layer\_id[ i ] is a direct reference layer of the current picture, the associated inter-layer picture is the picture that is or could be included in the reference picture lists of the current picture. Otherwise, the associated inter-layer picture is any picture with nuh\_layer\_id equal to scaled\_ref\_layer\_id[ i ]. [Ed. (MH): If the term associated inter-layer picture becomes needed in other parts of the specification too, move the definition to F.3.]

NOTE 1 – When spatial scalability is in use, the associated inter-layer picture is a resampled picture of a direct reference layer.

NOTE 2 – scaled\_ref\_layer\_id[ i ] need not be among the direct reference layers for example when the spatial correspondence of an auxiliary picture to its associated primary picture is specified.

**scaled\_ref\_layer\_id**[ i ] specifies the nuh\_layer\_id value of the associated inter-layer picture for which scaled\_ref\_layer\_left\_offset[ i ], scaled\_ref\_layer\_top\_offset[ i ], scaled\_ref\_layer\_right\_offset[ i ] and scaled\_ref\_layer\_bottom\_offset[ i ] are specified. The value of scaled\_ref\_layer\_id[ i ] shall be less than the nuh\_layer\_id of any layer for which this SPS is the active SPS. [Ed. (MH): A constraint that scaled reference offsets shall not be used for Stereo Main profile was added in the profile specification.]

**scaled\_ref\_layer\_left\_offset**[ scaled\_ref\_layer\_id[ i ] ] specifies the horizontal offset between the top-left luma sample of the associated inter-layer picture with nuh\_layer\_id equal to scaled\_ref\_layer\_id[ i ] and the top-left luma sample of the current picture in units of two luma samples. When not present, the value of scaled\_ref\_layer\_left\_offset[ scaled\_ref\_layer\_id[ i ] ] is inferred to be equal to 0.

**scaled\_ref\_layer\_top\_offset**[ scaled\_ref\_layer\_id[ i ] ] specifies the vertical offset between the top-left luma sample of the associated inter-layer picture with nuh\_layer\_id equal to scaled\_ref\_layer\_id[ i ] and the top-left luma sample of the current picture in units of two luma samples. When not present, the value of scaled\_ref\_layer\_top\_offset[ scaled\_ref\_layer\_id[ i ] ] is inferred to be equal to 0.

**scaled\_ref\_layer\_right\_offset**[ scaled\_ref\_layer\_id[ i ] ] specifies the horizontal offset between the bottom-right luma sample of the associated inter-layer picture with nuh\_layer\_id equal to scaled\_ref\_layer\_id[ i ] and the bottom-right luma sample of the current picture in units of two luma samples. When not present, the value of scaled\_ref\_layer\_right\_offset[ scaled\_ref\_layer\_id[ i ] ] is inferred to be equal to 0.

**scaled\_ref\_layer\_bottom\_offset**[ scaled\_ref\_layer\_id[ i ] ] specifies the vertical offset between the bottom-right luma sample of the associated inter-layer picture with nuh\_layer\_id equal to scaled\_ref\_layer\_id[ i ] and the bottom-right luma sample of the current picture in units of two luma samples. When not present, the value of scaled\_ref\_layer\_bottom\_offset[ scaled\_ref\_layer\_id[ i ] ] is inferred to be equal to 0.

* + - * 1. Picture parameter set RBSP semantics

The specifications in subclause 7.4.3.3 apply, with the following modifications:

NOTE – All PPSs, regardless of the values of their nuh\_layer\_id, share the same value space for pps\_pic\_parameter\_set\_id. In other words, a PPS with nuh\_layer\_id equal to X and pps\_pic\_parameter\_set\_id equal to A would update the previously received PPS with nuh\_layer\_id not equal to X and pps\_pic\_parameter\_set\_id equal to A.

**num\_extra\_slice\_header\_bits** specifies the number of extra slice header bits that are present in the slice header RBSP for coded pictures referring to the PPS. num\_extra\_slice\_header\_bits shall be in the range of 0 to 2, inclusive, in bitstreams conforming to this version of this Specification. Other values for num\_extra\_slice\_header\_bits are reserved for future use by ITU-T | ISO/IEC. However, decoders shall allow num\_extra\_slice\_header\_bits to have any value.

**pps\_infer\_scaling\_list\_flag** equal to 1 specifies that the syntax elements of the scaling list data syntax structure of the PPS are inferred to be equal to those of the PPS that is active for the layer with nuh\_layer\_id equal to pps\_scaling\_list\_ref\_layer\_id. pps\_infer\_scaling\_list\_flag equal to 0 specifies that the syntax elements of the scaling list data syntax structure of the PPS are not inferred. When not present, the value of pps\_infer\_scaling\_list\_flag is inferred to be 0.

**pps\_scaling\_list\_ref\_layer\_id** specifies the value of the nuh\_layer\_id of the layer for which the active PPS has the same scaling list data as the current PPS.

The value of pps\_scaling\_list\_ref\_layer\_id shall be in the range of 0 to 62, inclusive.

When avc\_base\_layer\_flag is equal to 1, it is a requirement of bitstream conformance that pps\_scaling\_list\_ref\_layer\_id shall be greater than 0.

It is a requirement of bitstream conformance that, when a PPS with nuh\_layer\_id equal to nuhLayerIdA is active for a layer with nuh\_layer\_id equal to nuhLayerIdB and pps\_infer\_scaling\_list\_flag in the PPS is equal to 1, pps\_infer\_scaling\_list\_flag shall be equal to 0 for the PPS that is active for the layer with nuh\_layer\_id equal to pps\_scaling\_list\_ref\_layer\_id.

It is a requirement of bitstream conformance that, when a PPS with nuh\_layer\_id equal to nuhLayerIdA is active for a layer with nuh\_layer\_id equal to nuhLayerIdB, the layer with nuh\_layer\_id equal to pps\_scaling\_list\_ref\_layer\_id shall be a direct or indirect reference layer of the layer with nuh\_layer\_id equal to nuhLayerIdB.

**pps\_scaling\_list\_data\_present\_flag** equal to 1 specifies that parameters are present in the PPS to modify the scaling lists specified by the active SPS. pps\_scaling\_list\_data\_present\_flag equal to 0 specifies that the scaling list data used for the pictures referring to the PPS are inferred to be equal to those specified by the active SPS. When scaling\_list\_enabled\_flag is equal to 0, the value of pps\_scaling\_list\_data\_present\_flag shall be equal to 0. When scaling\_list\_enabled\_flag is equal to 1, sps\_scaling\_list\_data\_present\_flag is equal to 0, and pps\_scaling\_list\_data\_present\_flag is equal to 0, the default scaling list data are used to derive the array ScalingFactor as described in the scaling list data semantics 7.4.5.

* + - * 1. Supplemental enhancement information RBSP semantics

The specifications in subclause 7.4.3.4 apply.

* + - * 1. Access unit delimiter RBSP semantics

The specifications in subclause 7.4.3.5 apply.

* + - * 1. End of sequence RBSP semantics

The specifications in subclause 7.4.3.6 apply.

* + - * 1. End of bitstream RBSP semantics

The specifications in subclause 7.4.3.7 apply.

* + - * 1. Filler data RBSP semantics

The specifications in subclause 7.4.3.8 apply.

* + - * 1. Slice segment layer RBSP semantics

The specifications in subclause 7.4.3.9 apply.

* + - * 1. RBSP slice segment trailing bits semantics

The specifications in subclause 7.4.3.10 apply.

* + - * 1. RBSP trailing bits semantics

The specifications in subclause 7.4.3.11 apply.

* + - * 1. Byte alignment semantics

The specifications in subclause 7.4.3.12 apply.

* + - 1. Profile, tier and level semantics

The profile\_tier\_level( ) syntax structure provides profile, tier and level information used for a layer set. When the profile\_tier\_level( ) syntax structure is included in a vps\_extension( ) syntax structure, the applicable layer set to which the profile\_tier\_level( ) syntax structure applies is specified by the corresponding lsIdx variable in the vps\_extension( ) syntax structure. When the profile\_tier\_level( ) syntax structure is included in a VPS, but not in a vps\_extension( ) syntax structure, the applicable layer set to which the profile\_tier\_level( ) syntax structure applies is the layer set specified by the index 0. When the profile\_tier\_level( ) syntax structure is included in an SPS, the layer set to which the profile\_tier\_level( ) syntax structure applies is the layer set specified by the index 0.

For interpretation of the following semantics, CVS refers to the CVS subset associated with the layer set to which the profile\_tier\_level( ) syntax structure applies.

When the syntax elements general\_profile\_space, general\_tier\_flag, general\_profile\_idc, general\_profile\_compatibility\_flag[ j ], general\_progressive\_source\_flag, general\_interlaced\_source\_flag, general\_non\_packed\_constraint\_flag, general\_frame\_only\_constraint\_flag, general\_reserved\_zero\_44bits are not present for the applicable layer set, they are inferred to be equal to the corresponding values of the layer set specified by the index (profile\_layer\_set\_ref\_minus1[ lsIdx ] +1 ).

When the syntax elements sub\_layer\_profile\_space[ i ], sub\_layer\_tier\_flag[ i ], sub\_layer\_profile\_idc[ i ], sub\_layer\_profile\_compatibility\_flag[ i ][ j ], sub\_layer\_progressive\_source\_flag[ i ], sub\_layer\_interlaced\_source\_flag[ i ], sub\_layer\_non\_packed\_constraint\_flag[ i ], sub\_layer\_frame\_only\_constraint\_flag[ i ], sub\_layer\_reserved\_zero\_44bits[ i ] are not present for the applicable layer set, and they are present in or inferred for the layer set specified by the index (profile\_layer\_set\_ref\_minus1[ lsIdx ] +1 ) they are inferred to be equal to the corresponding values of the layer set specified by the index (profile\_layer\_set\_ref\_minus1[ lsIdx ] +1 ).

The specifications in subclause 7.4.4 apply, with following modifications.

**general\_tier\_flag** specifies the tier context for the interpretation of general\_level\_idc as specified in Annex A or subclause G.11 or subclause H.11.

**general\_profile\_idc**, when general\_profile\_space is equal to 0, indicates a profile to which the CVS conforms as specified in Annex A or in subclause G.11 or in subclause H.11. Bitstreams shall not contain values of general\_profile\_idc other than those specified in Annex A or subclause G.11 or subclause H.11. Other values of general\_profile\_idc are reserved for future use by ITU-T | ISO/IEC.

**general\_profile\_compatibility\_flag**[ j ] equal to 1, when general\_profile\_space is equal to 0, indicates that the CVS conforms to the profile indicated by general\_profile\_idc equal to i as specified in Annex A or in subclause G.11 or in subclause H.11. When general\_profile\_space is equal to 0, general\_profile\_compatibility\_flag[ general\_profile\_idc ] shall be equal to 1. The value of general\_profile\_compatibility\_flag[ j ] shall be equal to 0 for any value of j that is not specified as an allowed value of general\_profile\_idc in Annex A or in subclause G.11 or in subclause H.11.

**general\_level\_idc** indicates a level to which the CVS conforms as specified in Annex A or subclause G.11 or subclause H.11. Bitstreams shall not contain values of general\_level\_idc other than those specified in Annex A or subclause G.11 or subclause H.11. Other values of general\_level\_idc are reserved for future use by ITU-T | ISO/IEC.

**sub\_layer\_profile\_present\_flag**[ i ] equal to 1, specifies that profile information is present in the profile\_tier\_level( ) syntax structure for the representation of the sub-layer with TemporalId equal to i. sub\_layer\_profile\_present\_flag[ i ] equal to 0 specifies that profile information is not present in the profile\_tier\_level( ) syntax structure for the representations of the sub-layer with TemporalId equal to i. When profilePresentFlag is equal to 0, sub\_layer\_profile\_present\_flag[ i ] shall be equal to 0.

* + - 1. Scaling list data semantics

The specifications in subclause 7.4.5 apply.

* + - 1. Supplemental enhancement information message semantics

The specifications in subclause 7.4.6 apply.

* + - 1. Slice segment header semantics
         1. General slice segment header semantics

The specifications in subclause 7.4.7.1 apply with the following modifications and additions.

When present, the value of the slice segment header syntax elements slice\_pic\_parameter\_set\_id, pic\_output\_flag, no\_output\_of\_prior\_pics\_flag, slice\_pic\_order\_cnt\_lsb, short\_term\_ref\_pic\_set\_sps\_flag, short\_term\_ref\_pic\_set\_idx, num\_long\_term\_sps, num\_long\_term\_pics, slice\_temporal\_mvp\_enabled\_flag, discardable\_flag, cross\_layer\_bla\_flag, poc\_reset\_flag, inter\_layer\_pred\_enabled\_flag, and num\_inter\_layer\_ref\_pics\_minus1 shall be the same in all slice segment headers of a coded picture. When present, the value of the slice segment header syntax elements lt\_idx\_sps[ i ], poc\_lsb\_lt[ i ], used\_by\_curr\_pic\_lt\_flag[ i ], delta\_poc\_msb\_present\_flag[ i ], delta\_poc\_msb\_cycle\_lt[ i ], and inter\_layer\_pred\_layer\_idc[ i ] shall be the same in all slice segment headers of a coded picture for each possible value of  i.

– "When nal\_unit\_type has a value in the range of 16 to 23, inclusive (IRAP picture), slice\_type shall be equal to 2." is replaced by "When nal\_unit\_type has a value in the range of 16 to 23 and nuh\_layer\_id is equal to 0, inclusive (IRAP picture), slice\_type shall be equal to 2."

**discardable\_flag** equal to 1 specifies that the coded picture is not used as a reference picture for inter prediction and is not used as an inter-layer reference picture in the decoding process of subsequent pictures in decoding order. discardable\_flag equal to 0 specifies that the coded picture may be used as a reference picture for inter prediction and may be used as an inter-layer reference picture in the decoding process of subsequent pictures in decoding order. When not present, the value of discardable\_flag is inferred to be equal to 0.

**cross\_layer\_bla\_flag** equal to 1 affects the derivation of NoClrasOutputFlag as specified in clause 8.1. cross\_layer\_bla\_flag shall be equal to 0 for pictures with nal\_unit\_type not equal to IDR\_W\_RADL or IDR\_N\_LP or with nuh\_layer\_id not equal to 0.

**poc\_reset\_flag** equal to 1 specifies that the derived picture order count for the current picture is equal to 0. poc\_reset\_flag equal to 0 specifies that the derived picture order count for the current picture may or may not be equal to 0. When not present, the value of poc\_reset\_flag is inferred to be equal to 0.

NOTE – When poc\_reset\_flag is equal to 1 in a base-layer picture, PicOrderCntVal is derived differently depending on whether the decoding process of subclause 8.3.1 or subclause F.8.3.1 is applied. Furthermore, when a base-layer picture with poc\_reset\_flag equal to 1 is prevTid0Pic according to subclause 8.3.1 or F.8.3.1, the variable prevPicOrderCntLsb is derived differently in subclauses 8.3.1 and F.8.3.1. In order to avoid PicOrderCntMsb to be updated incorrectly in one of the subclauses 8.3.1 or F.8.3.1, when prevTid0Pic is a base-layer picture with poc\_reset\_flag equal to 1 and either of the following conditions is true for prevPicOrderCntLsb derived with one of the subclauses 8.3.1 or F.8.3.1, the value of pic\_order\_cnt\_lsb of prevTid0Pic shall be such that the same condition is true also for prevPicOrderCntLsb derived with the other one of the subclauses 8.3.1 or F.8.3.1:

– ( slice\_pic\_order\_cnt\_lsb < prevPicOrderCntLsb )  &&   
 ( ( prevPicOrderCntLsb − slice\_pic\_order\_cnt\_lsb )  >=  ( MaxPicOrderCntLsb / 2 ) )

– ( slice\_pic\_order\_cnt\_lsb > prevPicOrderCntLsb )  &&  
 ( ( slice\_pic\_order\_cnt\_lsb – prevPicOrderCntLsb ) > ( MaxPicOrderCntLsb / 2 ) )

**inter\_layer\_pred\_enabled\_flag** equal to 1 specifies that inter-layer prediction may be used in decoding of the current picture. inter\_layer\_pred\_enabled\_flag equal to 0 specifies that inter-layer prediction is not used in decoding of the current picture.

**num\_inter\_layer\_ref\_pics\_minus1** plus 1 specifies the number of pictures of the current access unit that may be used in decoding of the current picture for inter-layer prediction. The length of the num\_inter\_layer\_ref\_pics\_minus1 syntax element is Ceil( Log2( NumDirectRefLayers[ nuh\_layer\_id ] ) ) bits. The value of num\_inter\_layer\_ref\_pics\_minus1 shall be in the range of 0 to NumDirectRefLayers[ nuh\_layer\_id ] − 1, inclusive.

The variables numRefLayerPics and refLayerPicFlag[ i ] and refLayerPicIdc[ j ] are derived as follows:

for( i = 0, j = 0; i < NumDirectRefLayers[ nuh\_layer\_id ]; i++ ) {  
 refLayerIdx = LayerIdxInVps[ RefLayerId[ nuh\_layer\_id ][ i ] ]  
 refLayerPicFlag[ i ] = ( sub\_layers\_vps\_max\_minus1[ refLayerIdx ] >= TemporalId ) &&   
 ( max\_tid\_il\_ref\_pics\_plus1[ refLayerIdx ][ LayerIdxInVps[ nuh\_layer\_id ] ] > TemporalId )  
 if( refLayerPicFlag[ i ] )  
 refLayerPicIdc[ j++ ] = i  
}

numRefLayerPics = j

The variable NumActiveRefLayerPics is derived as follows:

if( nuh\_layer\_id = = 0 | | NumDirectRefLayers[ nuh\_layer\_id ] = = 0 )  
 NumActiveRefLayerPics = 0  
else if( all\_ref\_layers\_active\_flag )  
 NumActiveRefLayerPics = numRefLayerPics  
else if( !inter\_layer\_pred\_enabled\_flag )  
 NumActiveRefLayerPics = 0  
else if( max\_one\_active\_ref\_layer\_flag | | NumDirectRefLayers[ nuh\_layer\_id ] = = 1 )  
 NumActiveRefLayerPics = refLayerPicFlag[ 0 ] ? 1 : 0  
else  
 NumActiveRefLayerPics = num\_inter\_layer\_ref\_pics\_minus1 + 1

All slices of a coded picture shall have the same value of NumActiveRefLayerPics.

**inter\_layer\_pred\_layer\_idc[**i ] specifies the variable, RefPicLayerId[ i ], representing the nuh\_layer\_id of the i-th picture that may be used by the current picture for inter-layer prediction. The length of the syntax element inter\_layer\_pred\_layer\_idc[ i ] is Ceil( Log2( NumDirectRefLayers[ nuh\_layer\_id ] ) ) bits. The value of inter\_layer\_pred\_layer\_idc[ i ] shall be in the range of 0 to NumDirectRefLayers[ nuh\_layer\_id ] − 1, inclusive. When not present, the value of inter\_layer\_pred\_layer\_idc[ i ] is inferred to be equal to refLayerPicIdc[ i ].

When i is greater than 0, inter\_layer\_pred\_layer\_idc[ i ] shall be greater than inter\_layer\_pred\_layer\_idc[ i − 1 ].

The variables RefPicLayerId[ i ] for all values of i in the range of 0 to NumActiveRefLayerPics − 1, inclusive, are derived as follows:

for( i = 0, j = 0; i < NumActiveRefLayerPics; i++)  
 RefPicLayerId[ i ] = RefLayerId[ nuh\_layer\_id ][ inter\_layer\_pred\_layer\_idc[ i ] ]

It is a requirement of bitstream conformance that for each value of i in the range of 0 to NumActiveRefLayerPics − 1, inclusive, either of the following two conditions shall be true:

– The value of max\_tid\_il\_ref\_pics\_plus1[ LayerIdxInVps[ RefPicLayerId[ i ] ] ][ LayerIdxInVps[ nuh\_layer\_id ] ] is greater than TemporalId.

– The values of max\_tid\_il\_ref\_pics\_plus1[ LayerIdxInVps[ RefPicLayerId[ i ] ] ][ LayerIdxInVps[ nuh\_layer\_id ] ] and TemporalId are both equal to 0 and the picture in the current access unit with nuh\_layer\_id equal to RefPicLayerId[ i ] is an IRAP picture.

**ref\_layer\_rps\_present\_flag**[ i ] equal to 0 specifies that no short\_term\_ref\_pic\_set( ) syntax structure is provided for the direct reference layer with nuh\_layer\_id equal to RefLayerId[ nuh\_layer\_id ][ i ]. ref\_layer\_rps\_present\_flag[ i ] equal to 1 specifies that a short\_term\_ref\_pic\_set( ) syntax structure is provided for the direct reference layer with nuh\_layer\_id equal to RefLayerId[ nuh\_layer\_id ][ i ] and referred to as a reference-layer short-term reference picture set. When ref\_layer\_rps\_present\_flag[ i ] is not present, it is inferred to be equal to ( sps\_ref\_layer\_rps\_enabled\_flag && NumDirectRefLayers[ nuh\_layer\_id ]  = =  1 ).

All slices of a coded picture shall have the same values of ref\_layer\_rps\_present\_flag[ i ] and the same content in the short\_term\_ref\_pic\_set( ) syntax structure for the direct reference layer with nuh\_layer\_id equal to RefLayerId[ nuh\_layer\_id ][ i ].

For the short\_term\_ref\_pic\_set( ) syntax structure, the decoding process for reference picture set in subclause F.8.3.2 is invoked with the modifications of assigning currPicLayerId equal to RefLayerId[ nuh\_layer\_id ][ i ] and not changing marking of any pictures to "unused for reference" or "used for long-term reference". It is a requirement of bitstream conformance that the resulting lists PocStFoll, PocLtCurr, and PocLtFoll are empty. The resulting lists PocStCurrBefore and PocStCurrAfter are assigned to variables RefLayerPocStCurrBefore[ i ] and RefLayerPocStCurrAfter[ i ]. For the purpose of decoding the current picture, the pictures identified by the lists RefLayerPocStCurrBefore[ i ] and RefLayerPocStCurrAfter[ i ] are temporarily marked as "used for long-term reference", while their previous marking is restored after the decoding of the current picture. The resulting variables RefPicSetStCurrBefore, RefPicSetStCurrAfter, NumPocStCurrBefore and NumPocStCurrAfter are assigned to variables RefLayerRefPicSetCurrBefore, RefLayerRefPicSetCurrAfter, RefLayerNumPocStCurrBefore[ i ] and RefLayerNumPocStCurrAfter[ i ]. When vps\_base\_layer\_internal\_flag is equal to 0, ref\_layer\_rps\_present\_flag[ i ] is equal to 1 and RefLayerId[ nuh\_layer\_id ][ i ] is equal to 0, a decoded picture associated with each item in the lists RefLayerRefPicSetCurrBefore and RefLayerRefPicSetCurrAfter is provided by external means.

When all values of ref\_layer\_rps\_present\_flag[ i ] are equal to 0, the variable NumActiveDiagRefLayerPics is set equal to 0. When ref\_layer\_rps\_present\_flag[ i ] is equal to 1, the variable NumActiveDiagRefLayerPics is set equal to RefLayerNumPocStCurrBefore[ RefLayerId[ nuh\_layer\_id ][ i ] ] + RefLayerNumPocStCurrAfter[ RefLayerId[ nuh\_layer\_id ][ i ] ].

[Ed. (MH): For a design option allowing diagonal inter-layer prediction from more than one direct reference layer, NumActiveDiagRefLayerPics is derived as follows:

The variable NumActiveDiagRefLayerPics is derived as follows

NumActiveDiagRefLayerPics = 0  
for( i = 0; i < NumDirectRefLayers[ nuh\_layer\_id ]; i ++ ) {  
 if( ref\_layer\_rps\_present\_flag[ i ] )  
 NumActiveDiagRefLayerPics += RefLayerNumPocStCurrBefore[ i ] + RefLayerNumPocStCurrAfter[ i ]  
}

* + - * 1. ]Reference picture list modification semantics

The specifications in subclause 7.4.7.2 apply with following modifications.

– Equation (7‑43) specifying the derivation of NumPicTotalCurr is replaced by:

NumPicTotalCurr = 0  
for( i = 0; i < NumNegativePics[ CurrRpsIdx ]; i++)  
 if(UsedByCurrPicS0[ CurrRpsIdx ][ i ] )  
 NumPicTotalCurr++  
for( i = 0; i < NumPositivePics[ CurrRpsIdx ]; i++) (F‑31)  
 if(UsedByCurrPicS1[ CurrRpsIdx ][ i ] )  
 NumPicTotalCurr++  
for( i = 0; i < num\_long\_term\_sps + num\_long\_term\_pics; i++ )  
 if( UsedByCurrPicLt[ i ] )  
 NumPicTotalCurr++  
NumPicTotalCurr += NumActiveRefLayerPics  
NumPicTotalCurr += NumActiveDiagRefLayerPics

* + - * 1. Weighted prediction parameters semantics

The specifications in subclause 7.4.7.3 apply.

* + - 1. Short-term reference picture set semantics
      2. The specifications in subclause 7.4.8 apply.Slice segment data semantics
         1. General slice segment data semantics

The specifications in subclause 7.4.9.1 apply.

* + - * 1. Coding tree unit semantics

The specifications in subclause 7.4.9.2 apply.

* + - * 1. Sample adaptive offset semantics

The specifications in subclause 7.4.9.3 apply.

* + - * 1. Coding quadtree semantics

The specifications in subclause 7.4.9.4 apply.

* + - * 1. Coding unit semantics

The specifications in subclause 7.4.9.5 apply.

* + - * 1. Prediction unit semantics

The specifications in subclause 7.4.9.6 apply.

* + - * 1. PCM sample semantics

The specifications in subclause 7.4.9.7 apply.

* + - * 1. Transform tree semantics

The specifications in subclause 7.4.9.8 apply.

* + - * 1. Motion vector difference semantics

The specifications in subclause 7.4.9.9 apply.

* + - * 1. Transform unit semantics

The specifications in subclause 7.4.9.10 apply.

* + - * 1. Residual coding semantics

The specifications in subclause 7.4.9.11 apply.

* 1. Decoding process
     1. General decoding process

The specifications in subclause 8.1 apply with following changes:

– Replace the references to clause 7, clause 10, and subclause 8.1.1 with subclauses F.7, F.10, and F.8.1.1, respectively.

– Add at the end of the subclause, add the following sentence:

When the current picture has nuh\_layer\_id greater than 0, the decoding process for a coded picture with nuh\_layer\_id greater than 0 as specified in subclause F.8.1.2 is invoked.

* + - 1. Decoding process for a coded picture with nuh\_layer\_id equal to 0

The specifications in subclause 8.1.1 apply with the following changes:

– Replace the references to subclauses 8.2, 8.3, 8.3.1, 8.3.2, 8.3.3, 8.3.4, 8.4, 8.5, 8.6, and 8.7 with subclauses F.8.2, F.8.3, F.8.3.1, F.8.3.2, F.8.3.3, F.8.3.4, F.8.4, F.8.5, F.8.6, and F.8.7, respectively.

* + - 1. Decoding process for a coded picture with nuh\_layer\_id greater than 0

The decoding process operates as follows for the current picture CurrPic.

– For the decoding of the slice segment header of the first slice, in decoding order, of the current picture, the decoding process for starting the decoding of a coded picture with nuh\_layer\_id greater than 0 specified in subclause F.8.1.3 is invoked.

– If ViewScalExtLayerFlag[ nuh\_layer\_id ] is equal to 1, the decoding process for a coded picture with nuh\_layer\_id greater than 0 specified in subclause G.8.1.1 is invoked. [Ed. (YK): It looks a bit odd to refer to Annex G here. Is this avoidable?] [Ed. (JC): The current design of the spec is to use annex F as the entry for decoding process of all multi-layer extensions. This design may not feasible for combined scalability. When a layer is both scalable layer and multi-view layer, the invoking process may have problem]

– Otherwise, when DependencyId[ nuh\_layer\_id] is greater than 0, the decoding process for a coded picture with nuh\_layer\_id greater than 0 specified in subclause H.8.1.1 is invoked.

– After all slices of the current picture have been decoded, the decoding process for ending the decoding of a coded picture with nuh\_layer\_id greater than 0 specified in subclause F.8.1.4 is invoked.

* + - 1. Decoding process for starting the decoding of a coded picture with nuh\_layer\_id greater than 0

Each picture referred to in this subclause is a complete coded picture.

The decoding process operates as follows for the current picture CurrPic:

1. The decoding of NAL units is specified in subclause F.8.2.
2. The processes in subclause F.8.3 specify the following decoding processes using syntax elements in the slice segment layer and above:

– Variables and functions relating to picture order count are derived in subclause F.8.3.1. This needs to be invoked only for the first slice segment of a picture. It is a requirement of bitstream conformance that PicOrderCntVal shall remain unchanged within an access unit.

– The decoding process for RPS in subclause F.8.3.2 is invoked, wherein only reference pictures with a nuh\_layer\_id equal to that of CurrPic may be marked as "unused for reference" or "used for long-term reference" and any picture with a different value of nuh\_layer\_id is not marked. This needs to be invoked only for the first slice segment of a picture.

– When FirstPicInLayerDecodedFlag[ nuh\_layer\_id ] is equal to 0, the decoding process for generating unavailable reference pictures specified in subclause F.8.1.5 is invoked, which needs to be invoked only for the first slice segment of a picture.

– When FirstPicInLayerDecodedFlag[ nuh\_layer\_id ] is not equal to 0 and the current picture is an IRAP picture with NoRaslOutputFlag equal to 1, the decoding process for generating unavailable reference pictures specified in subclause F.8.3.3 is invoked, which needs to be invoked only for the first slice segment of a picture.

* + - 1. Decoding process for ending the decoding of a coded picture with nuh\_layer\_id greater than 0

PicOutputFlag is set as follows:

– If LayerInitializedFlag[ nuh\_layer\_id ] is equal to 0, PicOutputFlag is set equal to 0.

– Otherwise, if the current picture is a RASL picture and NoRaslOutputFlag of the associated IRAP picture is equal to 1, PicOutputFlag is set equal to 0.

– Otherwise, PicOutputFlag is set equal to pic\_output\_flag.

The following applies:

– If discardable\_flag is equal to 1, the decoded picture is marked as "unused for reference".

– Otherwise, the decoded picture is marked as "used for short-term reference".

When TemporalId is equal to HighestTid, the marking process for sub-layer non-reference pictures not needed for inter-layer prediction specified in subclause F.8.1.4.1 is invoked with latestDecLayerId equal to nuh\_layer\_id as input.

When FirstPicInLayerDecodedFlag[ nuh\_layer\_id ] is equal to 0, FirstPicInLayerDecodedFlag[ nuh\_layer\_id ] is set equal to 1.

* + - * 1. Marking process for sub-layer non-reference pictures not needed for inter-layer prediction

Input to this process is:

– a nuh\_layer\_id value latestDecLayerId

Output of this process is:

– potentially updated marking as "unused for reference" for some decoded pictures

NOTE – This process marks pictures that are not needed for inter or inter-layer prediction as "unused for reference". When TemporalId is less than HighestTid, the current picture may be used for reference in inter prediction and this process is not invoked.

The variables numTargetDecLayers, and latestDecIdx are derived as follows:

– numTargetDecLayers is set equal to the number of entries in TargetDecLayerIdList.

– latestDecIdx is set equal to the value of i for which TargetDecLayerIdList[ i ] is equal to latestDecLayerId.

For i in the range of 0 to latestDecIdx, inclusive, the following applies for marking of pictures as "unused for reference":

– Let currPic be the picture in the current access unit with nuh\_layer\_id equal to TargetDecLayerIdList[ i ].

– When currPic is marked as "used for reference" and is a sub-layer non-reference picture, the following applies:

– The variable currTid is set equal to the value of TemporalId of currPic.

– The variable remainingInterLayerReferencesFlag is derived as specified in the following:

remainingInterLayerReferencesFlag = 0  
 iLidx = LayerIdxInVps[ TargetDecLayerIdList[ i ] ]  
 for( j = latestDecIdx + 1; j < numTargetDecLayers; j++ ) {  
 jLidx = LayerIdxInVps[ TargetDecLayerIdList[ j ] ]  
 if( currTid <= ( max\_tid\_il\_ref\_pics\_plus1[ iLidx ][ jLidx ] –1 ) )  
 for( k = 0; k < NumDirectRefLayers[ TargetDecLayerIdList[ j ] ]; k++ )  
 if( TargetDecLayerIdList[ i ] = = RefLayerId[ TargetDecLayerIdList[ j ] ][ k ] )  
 remainingInterLayerReferencesFlag = 1  
 }

– When remainingInterLayerReferenceFlag is equal to 0, currPic is marked as "unused for reference".

* + - 1. Generation of unavailable reference pictures for pictures first in decoding order within a layer

This process is invoked for a picture with nuh\_layer\_id equal to layerId, when FirstPicInLayerDecodedFlag[ layerId ] is equal to 0.

NOTE – A cross-layer random access skipped (CL-RAS) picture is a picture with nuh\_layer\_id equal to layerId such that LayerInitializedFlag[ layerId ] is equal to 0 when the decoding process for starting the decoding of a coded picture with nuh\_layer\_id greater than 0 is invoked. The entire specification of the decoding process for CL-RAS pictures is included only for purposes of specifying constraints on the allowed syntax content of such CL-RAS pictures. During the decoding process, any CL-RAS pictures may be ignored, as these pictures are not specified for output and have no effect on the decoding process of any other pictures that are specified for output. However, in HRD operations as specified in Annex C, CL-RAS pictures may need to be taken into consideration in derivation of CPB arrival and removal times.

When this process is invoked, the following applies:

– For each RefPicSetStCurrBefore[ i ], with i in the range of 0 to NumPocStCurrBefore – 1, inclusive, that is equal to “no-reference picture”, a picture is generated as specified in subclause 8.3.3.2, and the following applies:

– The value of PicOrderCntVal for the generated picture is set equal to PocStCurrBefore[ i ].

– The value of PicOutputFlag for the generated picture is set equal to 0.

– The generated picture is marked as "used for short-term reference".

– RefPicSetStCurrBefore[ i ] is set to be the generated reference picture.

– The value of nuh\_layer\_id for the generated picture is set equal to nuh\_layer\_id.

– For each RefPicSetStCurrAfter[ i ], with i in the range of 0 to NumPocStCurrAfter – 1, inclusive, that is equal to “no-reference picture”, a picture is generated as specified in subclause 8.3.3.2, and the following applies:

– The value of PicOrderCntVal for the generated picture is set equal to PocStCurrAfter[ i ].

– The value of PicOutputFlag for the generated picture is set equal to 0.

– The generated picture is marked as "used for short-term reference".

– RefPicSetStCurrAfter[ i ] is set to be the generated reference picture.

– The value of nuh\_layer\_id for the generated picture is set equal to nuh\_layer\_id.

– For each RefPicSetStFoll[ i ], with i in the range of 0 to NumPocStFoll − 1, inclusive, that is equal to "no reference picture", a picture is generated as specified in subclause 8.3.3.2, and the following applies:

– The value of PicOrderCntVal for the generated picture is set equal to PocStFoll[ i ].

– The value of PicOutputFlag for the generated picture is set equal to 0.

– The generated picture is marked as "used for short-term reference".

– RefPicSetStFoll[ i ] is set to be the generated reference picture.

– The value of nuh\_layer\_id for the generated picture is set equal to nuh\_layer\_id.

– For each RefPicSetLtCurr[ i ], with i in the range of 0 to NumPocLtCurr – 1, inclusive, that is equal to “no-reference picture”, a picture is generated as specified in subclause 8.3.3.2, and the following applies:

– The value of PicOrderCntVal for the generated picture is set equal to PocLtCurr[ i ].

– The value of slice\_pic\_order\_cnt\_lsb for the generated picture is inferred to be equal to ( PocLtCurr[ i ] & ( MaxPicOrderCntLsb – 1 ) ).

– The value of PicOutputFlag for the generated picture is set equal to 0.

– The generated picture is marked as "used for long-term reference".

– RefPicSetLtCurr[ i ] is set to be the generated reference picture.

– The value of nuh\_layer\_id for the generated picture is set equal to nuh\_layer\_id.

– For each RefPicSetLtFoll[ i ], with i in the range of 0 to NumPocLtFoll − 1, inclusive, that is equal to "no reference picture", a picture is generated as specified in subclause 8.3.3.2, and the following applies:

– The value of PicOrderCntVal for the generated picture is set equal to PocLtFoll[ i ].

– The value of slice\_pic\_order\_cnt\_lsb for the generated picture is inferred to be equal to ( PocLtFoll[ i ] & ( MaxPicOrderCntLsb – 1 ) ).

– The value of PicOutputFlag for the generated picture is set equal to 0.

– The generated picture is marked as "used for long-term reference".

– RefPicSetLtFoll[ i ] is set to be the generated reference picture.

– The value of nuh\_layer\_id for the generated picture is set equal to nuh\_layer\_id.

* + 1. NAL unit decoding process

The specifications in subclause 8.2 apply.

* + 1. Slice decoding processes
       1. Decoding process for picture order count

Output of this process is PicOrderCntVal, the picture order count of the current picture.

Picture order counts are used to identify pictures, for deriving motion parameters in merge mode and motion vector prediction, and for decoder conformance checking (see subclause C.5).

Each coded picture is associated with a picture order count variable, denoted as PicOrderCntVal.

If FirstPicInLayerDecodedFlag[ nuh\_layer\_id ] is equal to 0 or the current picture is an IRAP picture with NoRaslOutputFlag equal to 1, the variable PicOrderCntMsb is set equal to 0. [Ed. (MH): When the first picture in an enhancement layer is in an access unit which follows in decoding order and precedes in output order an initial IRAP access unit with NoClrasOutputFlag equal to 1, PicOrderCntVal of the first picture in the enhancement layer differs from the PicOrderCntVal of the base-layer picture in the same access unit.] [Ed. (MH): This derivation of PicOrderCntMsb equal to 0 imposes a constraint that the layer-wise start-up up to the highest layer must take place within a POC range of 0 to MaxPicOrderLsb – 1, inclusive.] Otherwise, PicOrderCntMsb is derived as follows:

* The variable prevPicOrderCntLsb is set equal to PrevPicOrderCnt[ nuh\_layer\_id ] & ( MaxPicOrderCntLsb − 1 ).
* The variable prevPicOrderCntMsb is set equal to PrevPicOrderCnt[ nuh\_layer\_id ] − prevPicOrderCntLsb.
* PicOrderCntMsb is derived as follows:

if( ( slice\_pic\_order\_cnt\_lsb < prevPicOrderCntLsb ) &&  
 ( ( prevPicOrderCntLsb − slice\_pic\_order\_cnt\_lsb ) >= ( MaxPicOrderCntLsb / 2 ) ) )  
 PicOrderCntMsb = prevPicOrderCntMsb + MaxPicOrderCntLsb (F‑32)  
else if( (slice\_pic\_order\_cnt\_lsb > prevPicOrderCntLsb ) &&  
 ( (slice\_pic\_order\_cnt\_lsb − prevPicOrderCntLsb ) > ( MaxPicOrderCntLsb / 2 ) ) )  
 PicOrderCntMsb = prevPicOrderCntMsb − MaxPicOrderCntLsb  
else  
 PicOrderCntMsb = prevPicOrderCntMsb

PicOrderCntVal is derived as follows:

PicOrderCntVal = PicOrderCntMsb + slice\_pic\_order\_cnt\_lsb (F‑33)

When poc\_reset\_flag is equal to 1, the following steps apply in the order listed:

* The PicOrderCntVal of each picture that is in the DPB and belongs to the same layer as the current picture is decremented by PicOrderCntVal.
* PrevPicOrderCnt[ nuh\_layer\_id ] is decremented by PicOrderCntVal.
* PicOrderCntVal is set equal to 0.

When the current picture is not a RASL picture, a RADL picture or a sub-layer non-reference picture, and the current picture has TemporalId equal to 0, PrevPicOrderCnt[ nuh\_layer\_id ] is set equal to PicOrderCntVal.

The value of PicOrderCntVal shall be in the range of −231 to 231 − 1, inclusive. In one CVS, the PicOrderCntVal values for any two coded pictures in the same layer shall not be the same.

The function PicOrderCnt( picX ) is specified as follows:

PicOrderCnt( picX ) = PicOrderCntVal of the picture picX (F‑34)

The function DiffPicOrderCnt( picA, picB ) is specified as follows:

DiffPicOrderCnt( picA, picB ) = PicOrderCnt( picA ) − PicOrderCnt( picB ) (F‑35)

The bitstream shall not contain data that result in values of DiffPicOrderCnt( picA, picB ) used in the decoding process that are not in the range of −215 to 215 − 1, inclusive.

NOTE – Let X be the current picture and Y and Z be two other pictures in the same sequence, Y and Z are considered to be in the same output order direction from X when both DiffPicOrderCnt( X, Y ) and DiffPicOrderCnt( X, Z ) are positive or both are negative.

* + - 1. Decoding process for reference picture set

The specifications in subclause 8.3.2 apply with the following changes:

* Replace the references to subclauses 7.4.7.2, 8.3.1, 8.3.3, and 8.3.4 to subclauses F.7.4.7.2, F.8.3.1, F.8.3.3, and F.8.3.4, respectively.
  + - 1. Decoding process for generating unavailable reference pictures

The specifications in subclause 8.3.3 apply.

* + - 1. Decoding process for reference picture lists construction

The specifications in subclause 8.3.4 apply.

* + - 1. Decoding process for collocated picture and no backward prediction flag

The specifications in subclause 8.3.5 apply.

* + 1. Decoding process for coding units coded in intra prediction mode

The specifications in subclause 8.4 apply.

* + 1. Decoding process for coding units coded in inter prediction mode

The specifications in subclause 8.5 apply with the following additions.

The variable currLayerId is set equal to nuh\_layer\_id of the current decoded picture

It is a requirement of bitstream conformance that when the reference picture represented by the variable refIdxLX and derived by invoking the subclause 8.5.3.2, for X being replaced by either 0 or 1, is an inter-layer reference picture, VpsInterLayerSamplePredictionEnabled[ LayerIdxInVps[ currLayerId ] ][ LayerIdxInVps[ rLId ] ] shall be equal to 1, where rLId is set equal to nuh\_layer\_id of the inter-layer picture.

It is a requirement of bitstream conformance when the collocated picture colPic, used for temporal motion vector prediction and derived by invoking the subclause 8.5.3.2.7, is an inter-layer reference picture, VpsInterLayerMotionPredictionEnabled[ LayerIdxInVps[ currLayerId ] ][ LayerIdxInVps[ rLId ] ] shall be equal to 1, where rLId is set equal to nuh\_layer\_id of the inter-layer picture.

* + 1. Scaling, transformation and array construction process prior to deblocking filter process

The specifications in subclause 8.6 apply.

* + 1. In-loop filter process

The specifications in subclause 8.7 apply.

* 1. Parsing process

The specifications in clause 9 apply.

* 1. Specification of bitstream subsets

The specifications in clause 10 apply.

* 1. (Void)
  2. Byte stream format

The specifications in Annex B apply.

* 1. Hypothetical reference decoder

The specifications in Annex C and its subclauses apply.

* 1. SEI messages

The specifications in Annex D together with the extensions and modifications specified in this subclause apply.

[Ed. (CY): to check the semantics in D.3 and that in F.14.2 to make them align with the AU definition.]

* + 1. SEI message syntax
       1. Layers not present SEI message syntax

|  |  |
| --- | --- |
| layers\_not\_present( payloadSize ) { | **Descriptor** |
| **lp\_sei\_active\_vps\_id** | u(4) |
| for( i = 0; i <= MaxLayersMinus1; i++ ) |  |
| **layer\_not\_present\_flag**[ i ] | u(1) |
| } |  |

* + - 1. Inter-layer constrained tile sets SEI message syntax

|  |  |
| --- | --- |
| inter\_layer\_constrained\_tile\_sets( payloadSize ) { | Descriptor |
| **il\_all\_tiles\_exact\_sample\_value\_match\_flag** | u(1) |
| **il\_one\_tile\_per\_tile\_set\_flag** | u(1) |
| if( !il\_one\_tile\_per\_tile\_set\_flag ) { |  |
| **il\_num\_sets\_in\_message\_minus1** | ue(v) |
| if( il\_num\_sets\_in\_message\_minus1 ) |  |
| **skipped\_tile\_set\_present\_flag** | u(1) |
| numSignificantSets = il\_num\_sets\_in\_message\_minus1  – skipped\_tile\_set\_present\_flag + 1 |  |
| for( i = 0; i < numSignificantSets; i++ ) { |  |
| **ilcts\_id**[ i ] | ue(v) |
| **il\_num\_tile\_rects\_in\_set\_minus1**[ i ] | ue(v) |
| for( j = 0; j <= il\_num\_tile\_rects\_in\_set\_minus1[ i ]; j++ ) { |  |
| **il\_top\_left\_tile\_index[**i **][**j **]** | ue(v) |
| **il\_bottom\_right\_tile\_index**[ i ]**[**j **]** | ue(v) |
| } |  |
| **ilc\_idc**[ i ] | u(2) |
| if ( !il\_all\_tiles\_exact\_sample\_value\_match\_flag ) |  |
| **il\_exact\_sample\_value\_match\_flag**[ i ] | u(1) |
| } |  |
| } else |  |
| **all\_tiles\_ilc\_idc** | u(2) |
| } |  |

* + - 1. Bitstream partition nesting SEI message syntax

|  |  |
| --- | --- |
| bsp\_nesting( payloadSize ) { | **Descriptor** |
| **bsp\_idx** | ue(v) |
| while( !byte\_aligned( ) ) |  |
| **bsp\_nesting\_zero\_bit** /\* equal to 0 \*/ | u(1) |
| do |  |
| sei\_message( ) |  |
| while( more\_rbsp\_data( ) ) |  |
| } |  |

* + - 1. Bitstream partition initial arrival time SEI message syntax

|  |  |
| --- | --- |
| bsp\_initial\_arrival\_time( payloadSize ) { | **Descriptor** |
| if( NalHrdBpPresentFlag ) |  |
| for( i = 0; i < SchedCombCnt; i++ ) |  |
| **nal\_initial\_arrival\_delay**[ i ] | u(v) |
| else |  |
| for( i = 0; i < SchedCombCnt; i++ ) |  |
| **vcl\_initial\_arrival\_delay**[ i ] | u(v) |
| } |  |

* + - 1. Bitstream partition HRD parameters SEI message syntax

|  |  |
| --- | --- |
| bsp\_hrd( payloadSize ) { | Descriptor |
| **sei\_num\_bsp\_hrd\_parameters\_minus1** | ue(v) |
| for( i = 0; i <= sei\_num\_bsp\_hrd\_parameters\_minus1; i++ ) { |  |
| if( i > 0 ) |  |
| **sei\_bsp\_cprms\_present\_flag**[ i ] | u(1) |
| hrd\_parameters( sei\_bsp\_cprms\_present\_flag[ i ],   nesting\_max\_temporal\_id\_plus1[ 0 ] – 1 ) |  |
| } |  |
| for( h=0; h <= nesting\_num\_ops\_minus1; h++ ) { |  |
| lsIdx = nesting\_op\_idx[ h ] |  |
| **sei\_num\_bitstream\_partitions\_minus1**[ lsIdx ] | ue(v) |
| for( i = 0; i <= sei\_num\_bitstream\_partitions\_minus1[ lsIdx ]; i++ ) |  |
| for( j = 0; j <= vps\_max\_layers\_minus1; j++ ) |  |
| if( layer\_id\_included\_flag[ nesting\_op\_idx[ lsIdx ] ][ j ] ) |  |
| **sei\_layer\_in\_bsp\_flag**[ lsIdx ][ i ][ j ] | u(1) |
| **sei\_num\_bsp\_sched\_combinations\_minus1**[ lsIdx ] | ue(v) |
| for( i = 0; i <= sei\_num\_bsp\_sched\_combinations\_minus1[ lsIdx ]; i++ ) |  |
| for( j = 0; j <= sei\_num\_bitstream\_partitions\_minus1[ lsIdx ]; j++ ) { |  |
| **sei\_bsp\_comb\_hrd\_idx**[ lsIdx ][ i ][ j ] | ue(v) |
| **sei\_bsp\_comb\_sched\_idx**[ lsIdx ][ i ][ j ] | ue(v) |
| } |  |
| } |  |
| } |  |

* + 1. SEI message semantics

Table F‑3 – Persistence scope of SEI messages (informative)

|  |  |
| --- | --- |
| SEI message | Persistence scope |
| Layers not present | The access unit containing the SEI message and up to but not including the next access unit, in decoding order, that contains a layers not present SEI message or the end of the CVS, whichever is earlier in decoding order |
| Inter-layer constrained tile sets | The CVS containing the SEI message |
| Bitstream partition nesting | Depending on the nested SEI messages. Each nested SEI  message has the same persistence scope as if the SEI message  was not nested |
| Bitstream partition initial arrival time | The remainder of the bitstream partition (specified by the containing bitstream partition nesting SEI message) |
| Bitstream partition HRD parameters | The CVS containing the SEI message |

The constraints of bitstream conformance specified in clause D.3.1 apply with the following additions.

Let prevVclNalUnitInAu of an SEI NAL unit or an SEI message be the preceding VCL NAL unit in decoding order, if any, in the same access unit, and nextVclNalUnitInAu of an SEI NAL unit or an SEI message be the next VCL NAL unit in decoding order, if any, in the same access unit. It is a requirement of bitstream conformance that the following restrictions apply:

– When a bitstream partition HRD parameters SEI message contained in a scalable nesting SEI message is present in an access unit, the scalable nesting SEI message shall not follow any other SEI message that follows the prevVclNalUnitInAu of the scalable nesting SEI message and precedes the nextVclNalUnitInAu of the scalable nesting SEI message, other than an active parameter sets SEI message, a non-nested buffering period SEI message, a non-nested picture timing SEI message, a non-nested decoding unit information SEI message, a scalable nesting SEI message including a buffering period SEI message, a picture timing SEI message or a decoding unit information SEI message, or another scalable nesting SEI message that contains a bitstream partition HRD parameters SEI message.

– When a buffering period SEI message, a picture timing SEI message, a decoding unit information SEI message or a bitstream partition initial arrival time SEI message is present in a bitstream partition nesting SEI message contained in a scalable nesting SEI message, the scalable nesting SEI message shall not follow any other SEI message that follows the prevVclNalUnitInAu of the scalable nesting SEI message and precedes the nextVclNalUnitInAu of the scalable nesting SEI message, other than an active parameter sets SEI message, a non-nested buffering period SEI message, a non-nested picture timing SEI message, a non-nested decoding unit information SEI message, a scalable nesting SEI message including a buffering period SEI message, a picture timing SEI message or a decoding unit information SEI message, a scalable nesting SEI message including a bitstream partition HRD parameters SEI message, or another scalable nesting SEI message that contains a bitstream partition nesting SEI message including a buffering period SEI message, a picture timing SEI message, a decoding unit information SEI message or a bitstream partition initial arrival time SEI message.

* + - 1. Layers not present SEI message semantics

The layers not present SEI message provides a mechanism for signalling that VCL NAL units of particular layers indicated by the VPS are not present in a particular set of access units.

The target access units are defined as the set of access units starting from the access unit containing the layers not present SEI message up to but not including the next access unit, in decoding order, that contains a layers not present change SEI message or the end of the CVS, whichever is earlier in decoding order.

When present, the layers not present SEI message applies to the target access units.

A layers not present SEI message shall not be included in a scalable nesting SEI message.

A layers not present SEI message shall not be included in an SEI NAL unit with TemporalId greater than 0.

**lp\_sei\_active\_vps\_id** identifies the active VPS of the CVS containing the layers not present SEI message. The value of lp\_sei\_active\_vps\_id shall be equal to the value of vps\_video\_parameter\_set\_id of the active VPS for the VCL NAL units of the access unit containing the SEI message.

**layer\_not\_present\_flag**[ i ] equal to 1 indicates that there are no VCL NAL units with nuh\_layer\_id equal to layer\_id\_in\_nuh[ i ] present in the target access units. layer\_not\_present\_flag[ i ] equal to 0 indicates that there may or may not be VCL NAL units with nuh\_layer\_id equal to layer\_id\_in\_nuh[ i ] present in the target access units.

When layer\_not\_present\_flag[ i ] is equal to 0 and i is greater than 0, layer\_not\_present\_flag[ LayerIdxInVps[ RefLayerId[ layer\_id\_in\_nuh[ i ] ][ j ] ] ] shall be equal to 0 for all values of j in the range of 0 to NumDirectRefLayers[ layer\_id\_in\_nuh[ i ] ] − 1, inclusive.

* + - 1. Inter-layer constrained tile sets SEI message semantics

The scope of the inter-layer constrained tile sets SEI message is the complete CVS. When an inter-layer tile sets SEI message is present in any access unit of a CVS, it shall be present for the first access unit of the CVS in decoding order and may also be present for other access units of the CVS.

The inter-layer constrained tile sets SEI message shall not be present for a layer when tiles\_enabled\_flag is equal to 0 for any PPS that is active for the layer.

The inter-layer constrained tile sets SEI message shall not be present for a layer unless every PPS that is active for the layer has tile\_boundaries\_aligned\_flag equal to 1 or fulfills the conditions that would be indicated by tile\_boundaries\_aligned\_flag being equal to 1.

The presence of the inter-layer tile sets SEI message indicates that the inter-layer inter prediction process is constrained such that no sample value outside each identified tile set, and no sample value at a fractional sample position that is derived using one or more sample values outside the identified tile set, is used for inter prediction of any sample within the identified tile set.

NOTE 1 – When loop filtering and resampling filter is applied across tile boundaries, inter-layer prediction of any samples within an inter-layer constrained tile set that refers to samples within 8 samples from an inter-layer constrained tile set boundary that is not also a picture boundary may result in propagation of mismatch error. An encoder can avoid such potential error propagation by avoiding the use of motion vectors that cause such references.

When more than one inter-layer constrained tile sets SEI message is present within the access units of a CVS, they shall contain identical content.

The number of inter-layer constrained tile sets SEI messages in each access unit shall not exceed 5.

**il\_all\_tiles\_exact\_sample\_value\_match\_flag** equal to equal to 1 indicates that, within the CVS, when the coding tree blocks that are outside of any identified tile are not decoded and the boundaries of the identified tile is treated as picture boundaries for purposes of the decoding process, the value of each sample in the identified tile would be exactly the same as the value of the sample that would be obtained when all the coding tree blocks of all pictures in the CVS are decoded. il\_all\_tiles\_exact\_sample\_value\_match\_flag equal to 0 indicates that, within the CVS, when the coding tree blocks that are outside of any identified tile are not decoded and the boundaries of the identified tile is treated as picture boundaries for purposes of the decoding process, the value of each sample in the identified tile may or may not be exactly the same as the value of the same sample when all the coding tree blocks of all pictures in the CVS are decoded.

**il\_one\_tile\_per\_tile\_set\_flag** equal to 1 indicates that each inter-layer constrained tile set contains one tile, and il\_num\_sets\_in\_message\_minus1 is not present. When il\_one\_tile\_per\_tile\_set\_flag is equal to zero, tile sets are signalled explicitly.

**il\_num\_sets\_in\_message\_minus1** plus 1 specifies the number of inter-layer tile sets identified in the SEI message. The value of il\_num\_sets\_in\_message\_minus1 shall be in the range of 0 to 255, inclusive.

**skipped\_tile\_set\_present\_flag** equal to 1 indicates that, within the CVS, the tile set consists of those remaining tiles that are not included in any earlier tile sets in the same message and all the prediction blocks that are inside the identified tile set having nuh\_layer\_id equal to ictsNuhLayerId are inter-layer predicted from aligned inter-layer reference pictures with nuh\_layer\_id equal to RefLayerId[ ictsNuhLayerId ][ NumDirectRefLayers[ ictsNuhLayerId ] – 1 ] and no residual\_coding syntax structure is present in any transform unit of the identified tile set, where ictsNuhLayerId is the value of nuh\_layer\_id of this message. skipped\_tile\_set\_present\_flag equal to 0 does not indicate a bitstream constraint within the CVS. When not present, the value of skipped\_tile\_set\_present\_flag is inferred to be equal to 0.

**ilcts\_id**[ i ] contains an identifying number that may be used to identify the purpose of the i-th identified tile set (for example, to identify an area to be extracted from the coded video sequence for a particular purpose). The value of ilcts\_id[ i ] shall be in the range of 0 to 232 − 2, inclusive.

Values of ilcts\_id[ i ] from 0 to 255 and from 512 to 231 − 1 may be used as determined by the application. Values of ilcts\_id[ i ] from 256 to 511 and from 231 to 232 − 2 are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of ilcts\_id[ i ] in the range of 256 to 511 or in the range of 231 to 232 − 2 shall ignore (remove from the bitstream and discard) it.

**il\_num\_tile\_rects\_in\_set\_minus1**[ i ] plus 1 specifies the number of rectangular regions of tiles in the i-th identified inter-layer constrained tile set. The value of il\_num\_tile\_rects\_in\_set\_minus1[ i ] shall be in the range of 0 to (num\_tile\_columns\_minus1 + 1) \* (num\_tile\_rows\_minus1 + 1) − 1, inclusive.

**il\_top\_left\_tile\_index**[ i ][ j ] and **il\_bottom\_right\_tile\_index**[ i ][ j ] identify the tile position of the top-left tile and the tile position of the bottom-right tile in a rectangular region of the i-th identified inter-layer constrained tile set, respectively, in tile raster scan order.

**il\_exact\_sample\_value\_match\_flag**[ i ] equal to 1 indicates that, within the CVS, when the coding tree blocks that do not belong to the inter-layer constrained tile set are not decoded and the boundaries of the inter-layer constrained tile set are treated as picture boundaries for purposes of the decoding process, the value of each sample in the inter-layer constrained tile set would be exactly the same as the value of the sample that would be obtained when all the coding tree blocks of all pictures in the coded video sequence are decoded. il\_exact\_sample\_value\_match\_flag[ i ] equal to 0 indicates that, within the CVS, when the coding tree blocks that are outside of the i-th identified inter-layer constrained tile set are not decoded and the boundaries of the inter-layer constrained tile set are treated as picture boundaries for purposes of the decoding process, the value of each sample in the identified tile set may or may not be exactly the same as the value of the same sample when all the coding tree blocks of the picture are decoded.

NOTE 2 – It should be feasible to use il\_exact\_sample\_value\_match\_flag equal to 1 when using certain combinations of loop\_filter\_across\_tiles\_enabled\_flag, pps\_loop\_filter\_across\_slices\_enabled\_flag, pps\_deblocking\_filter\_disabled\_flag, slice\_loop\_filter\_across\_slices\_enabled\_flag, slice\_deblocking\_filter\_disabled\_flag, sample\_adaptive\_offset\_enabled\_flag, slice\_sao\_luma\_flag, and slice\_sao\_chroma\_flag.

**ilc\_idc**[ i ] equal to 1 indicates that, within the CVS, no samples outside of the i-th identified tile set and no samples at a fractional sample position that is derived using one or more samples outside of the i-th identified tile set are used for inter-layer prediction of any sample within the i-th identified tile set with nuh\_layer\_id equal to ictsNuhLayerId, where ictsNuhLayerId is the value of nuh\_layer\_id of this message. ilc\_idc[ i ][ j ] equal to 2 indicates that, within the CVS, no prediction block in the i-th identified tile set with nuh\_layer\_id equal to ictsNuhLayerId is predicted from an inter-layer reference picture. ilc\_idc[ i ] equal to 0 indicates that, within the CVS, the inter-layer prediction process may or may not be constrained for the prediction block in the i-th identified tile set having nuh\_layer\_id equal to ictsNuhLayerId. The value of ilc\_idc[ i ] equal to 3 is reserved.

**all\_tiles\_ilc\_idc** equal to 1 indicates that, within the CVS, no sample value outside of each identified tile and no sample value at a fractional sample position that is derived using one or more samples outside of the identified tile is used for inter-layer prediction of any sample within the identified tile with nuh\_layer\_id equal to ictsNuhLayerId, where ictsNuhLayerId is the value of nuh\_layer\_id of this message. all\_tiles\_ilc\_idc equal to 2 indicates that, within the CVS, no prediction block in each identified tile with nuh\_layer\_id equal to ictsNuhLayerId is predicted from an inter-layer reference picture. all\_tiles\_ilc\_idc equal to 0 indicates that, within the CVS, the inter-layer prediction process may or may not be constrained for the tile having nuh\_layer\_id equal to ictsNuhLayerId. The value of all\_tiles\_ilc\_idc equal to 3 is reserved.

* + - 1. Bitstream partition nesting SEI message semantics

The bitstream partition nesting SEI message provides a mechanism to associate SEI messages with a bitstream partition of a layer set.

When present, this SEI message shall be contained within a scalable nesting SEI message. When this SEI message is contained in a scalable nesting SEI message, it shall be the only nested SEI message. In the scalable nesting SEI message containing this SEI message bitstream\_subset\_flag shall be equal to 1, nesting\_op\_flag is equal to 1, default\_op\_flag shall be equal to 0 and nesting\_num\_ops\_minus1 shall be equal to 0. The nuh\_layer\_id of the SEI NAL unit shall be equal to the highest value within the list nestingLayerIdList[ 0 ].

A bitstream partition nesting SEI message contains one or more SEI messages.

**bsp\_idx** specifies the bitstream partition index to which the contained SEI message apply as follows:

– If vps\_vui\_bsp\_hrd\_present\_flag is equal to 1, bsp\_idx is an index among the bitstream partitions specified for the layer set with index nesting\_op\_idx[ 0 ] in the vps\_vui\_bsp\_hrd\_parameters( ) syntax structure.

– Otherwise, an associated bitstream partition HRD parameters SEI message shall be present. The associated bitstream partition HRD parameter SEI message for the bitstream partition nesting SEI message is the preceding bitstream partition HRD parameters SEI message, in decoding order, that is nested in a scalable nesting SEI message with nesting\_op\_idx[ i ] that, with any value of i in the range of 0 to nesting\_num\_ops\_minus1 of the scalable nesting SEI message containing the bitstream partition HRD parameters SEI message, is equal to nesting\_op\_idx[ 0 ] of the scalable nesting SEI message containing the bitstream partition nesting SEI message. It is a requirement of bitstream conformance that when bitstream partition nesting SEI message is present, it shall have an associated bitstream partition HRD message within the same coded video sequence. bsp\_idx is an index among the bitstream partitions specified in the associated bitstream partition HRD parameters SEI message.

* + - 1. Bitstream partition initial arrival time SEI message semantics

The bitstream partition initial arrival time SEI message specifies the initial arrival times to be used in the bitstream-partition-specific CPB operation.

When present, this SEI message shall be contained within bitstream partition nesting SEI message that is contained in a scalable nesting SEI message. The same bitstream partition SEI message shall also contain a buffering period SEI message.

**nal\_initial\_arrival\_delay**[ i ] specifies the initial arrival time for the i-th schedule combination of the bitstream partition to which this SEI message applies, when NAL HRD parameters are in use.

**vcl\_initial\_arrival\_delay**[ i ] specifies the initial arrival time for the i-th schedule combination of the bitstream partition to which this SEI message applies, when VCL HRD parameters are in use.

* + - 1. Bitstream partition HRD parameters SEI message semantics

The bitstream partition HRD parameters SEI message specifies HRD parameters for bitstream-partition-specific CPB operation.

When present, this SEI message shall be contained within a scalable nesting SEI message in an initial IRAP access unit. When this SEI message is contained in a scalable nesting SEI message, it shall be the only nested SEI message. In the scalable nesting SEI message containing this SEI message, bitstream\_subset\_flag shall be equal to 1, nesting\_op\_flag shall be equal to 1 and default\_op\_flag shall be equal to 0. The nuh\_layer\_id of the SEI NAL unit shall be equal to the highest value within the lists nestingLayerIdList[ h ] with h in the range of 0 to nesting\_num\_ops\_minus1, inclusive.

**sei\_num\_bsp\_hrd\_parameters\_minus1** plus 1 specifies the number of hrd\_parameters( ) syntax structures present within this SEI message.

**sei\_bsp\_cprms\_present\_flag**[ i ] equal to 1 specifies that the HRD parameters that are common for all sub-layers are present in the i-th hrd\_parameters( ) syntax structure in this SEI message. sei\_bsp\_cprms\_present\_flag[ i ] equal to 0 specifies that the HRD parameters that are common for all sub-layers are not present in the i-th hrd\_parameters( ) syntax structure in this SEI message and are derived to be the same as the ( i − 1 )-th hrd\_parameters( ) syntax structure in the in this SEI message. sei\_bsp\_cprms\_present\_flag[ 0 ] is inferred to be equal to 1.

For the subsequent syntax elements of this SEI message, the variable lsIdx is set equal to nesting\_op\_idx[ h ].

**sei\_num\_bitstream\_partitions\_minus1**[ lsIdx ] plus 1 specifies the number of bitstream partitions for which HRD parameters are specified for the layer set with index nesting\_op\_idx[ h ].

**sei\_layer\_in\_bsp\_flag**[ lsIdx ][ i ][ j ] specifies that the layer with index j is a part of bitstream partition with index i within the layer set with index lsIdx.

It is a requirement of bitstream conformance that bitstream partition with index j shall not include direct or indirect reference layers of any layers in bitstream partition i for any values of i and j in the range of 0 to sei\_num\_bitstream\_partitions\_minus1[ h ], inclusive, such that i is less than j.

**sei\_num\_bsp\_sched\_combinations\_minus1**[ lsIdx ]plus 1 specifies the number of combinations of delivery schedules and hrd\_parameters( ) specified for bitstream partitions for the layer set with index lsIdx.

**sei\_bsp\_comb\_hrd\_idx**[ lsIdx ][ i ][ j ] specifies the index of hrd\_parameters( ) within this SEI message used in the i-th combination of a delivery schedule and hrd\_parameters( ) specified for the bitstream partition with index j and for the layer set with index lsIdx.

**sei\_bsp\_comb\_sched\_idx**[ lsIdx ][ i ][ j ] specifies the index of a delivery schedule within the hrd\_parameters( ) syntax structure with the index sei\_bsp\_comb\_hrd\_idx[ lsIdx ][ i ][ j ] that is used in the i-th combination of a delivery schedule and hrd\_parameters( ) specified for the bitstream partition with index j and for the layer set with index lsIdx.

* 1. Video usability information
     1. General

The specifications in clause E.1 apply.

* + 1. VUI syntax

The specifications in clause E.2 apply.

* + 1. VUI semantics
       1. VUI parameters semantics

The specifications in clause E.3.1 apply with the following modifications and additions.

vui\_timing\_info\_present\_flag equal to 1 specifies that vui\_num\_units\_in\_tick, vui\_time\_scale, vui\_poc\_proportional\_to\_timing\_flag, and vui\_hrd\_parameters\_present\_flag are present in the vui\_parameters( ) syntax structure. vui\_timing\_info\_present\_flag equal to 0 specifies that vui\_num\_units\_in\_tick, vui\_time\_scale, vui\_poc\_proportional\_to\_timing\_flag, and vui\_hrd\_parameters\_present\_flag are not present in the vui\_parameters( ) syntax structure. It is a requirement of bitstream conformance that, when nuh\_layer\_id is greater than 0, vui\_timing\_info\_present\_flag shall be equal to 0.

* + - 1. HRD parameters semantics

The specifications in clause E.3.2 apply.

* + - 1. Sub-layer HRD parameters semantics

The specifications in clause E.3.3 apply.

1. Annex H   
     
   Syntax, semantics and decoding processes for scalable extension

(This annex forms an integral part of this Recommendation | International Standard)

This annex specifies syntax, semantics and decoding processes,for scalable exetnsion that use the syntax, semantics, and decoding process specified in clauses 2-9 and Annex A-F.

* 1. Scope

Decoding process and bitstreams conforming to this annex are completely specified in this annex with reference made to clauses 2-9 and Annexes A-F.

* 1. Normative references

The specifications in clause 2 apply.

* 1. Definitions

The specifications in clause F.3 apply.

* 1. Abbreviations

The specifications in clause 4 apply.

* 1. Conventions

The specifications in clause 5 apply.

* 1. Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships
     1. Derivation process for reference layer sample location

The specification in clause 6 and all its subclauses apply with the following additions.

Input to this process is a luma location ( xP, yP ) relative to the top-left luma sample of the current picture.

Output of this process is a luma location ( xRef, yRef ) relative to the top-left luma sample of the reference layer picture.

The variables xRef and yRef are derived as follows:

xRef = ( ( xP ‑ ScaledRefLayerLeftOffset ) \* ScaleFactorX + ( 1 << 15 ) ) >> 16 (H‑1)

yRef = ( ( yP ‑ ScaledRefLayerTopOffset ) \* ScaleFactorY + ( 1 << 15 ) ) >> 16 (H‑2)

* + 1. Derivation process for reference layer sample location used in resampling

Inputs to this process are

– a variable cIdx specifying the color component index,

– a sample location ( xP, yP ) relative to the top-left sample of the color component of the current picture specified by cIdx.

Output of this process is a sample location ( xRef16, yRef16 ) specifying the reference layer sample location in units of 1/16-th sample relative to the top-left sample of the reference layer picture.

The variables offsetX and offsetY are derived as follows:

offsetX = ScaledRefLayerLeftOffset / ( ( cIdx = = 0)  ?  1 :  SubWidthC) (H‑3)  
offsetY = ScaledRefLayerTopOffset / ( ( cIdx = = 0)  ?  1 :  SubHeightC) (H‑4)

The variables phaseX, phaseY, addX and addY are derived as follows:

phaseX = ( cIdx = = 0 ) ? ( cross\_layer\_phase\_alignment\_flag << 1 ) : cross\_layer\_phase\_alignment\_flag (H‑5)  
phaseY = ( cIdx = = 0 ) ? ( cross\_layer\_phase\_alignment\_flag << 1 ) : cross\_layer\_phase\_alignment\_flag + 1 (H‑6)

addX = ( ScaleFactorX \* phaseX + 2 ) >> 2 (H‑7)   
addY = ( ScaleFactorY \* phaseY + 2 ) >> 2 (H‑8)

The variables xRef16 and yRef16 are derived as follows:

xRef16 = ( ( ( xP – offsetX ) \* ScaleFactorX  + addX + ( 1 << 11 ) ) >> 12 ) – ( phaseX << 2 ) (H‑9)  
yRef16 = ( ( ( yP – offsetY ) \* ScaleFactorY + addY + ( 1 << 11 ) ) >> 12 ) – ( phaseY << 2 ) (H‑10)

* 1. Syntax and semantics

The specifications in subclause F.7 and all its subclauses apply.

* 1. Decoding processes
     1. General decoding process

The specifications of subclause F.8.1 apply.

* + - 1. Decoding process for a coded picture with nuh\_layer\_id greater than 0

The decoding process operates as follows for the current picture CurrPic:

1. The decoding of NAL units is specified in subclause 8.2.
2. The processes in subclause H.8.1.2 and H.8.3.4 specify the following decoding processes using syntax elements in the slice segment layer and above:

– Prior to decoding the first slice of the current picture, subclause H.8.1.2 is invoked.

– At the beginning of the decoding process for each P or B slice, the decoding process for reference picture lists construction specified in subclause H.8.3.4 is invoked for derivation of reference picture list 0 (RefPicList0), and when decoding a B slice, reference picture list 1 (RefPicList1).

1. The processes in subclauses H.8.4, H.8.5, H.8.6, and H.8.7 specify decoding processes using syntax elements in all syntax structure layers. It is a requirement of bitstream conformance that the coded slices of the picture shall contain slice segment data for every coding tree unit of the picture, such that the division of the picture into slices, the division of the slices into slice segments, and the division of the slice segments into coding tree units each form a partitioning of the picture.
2. After all slices of the current picture have been decoded, the marking process for ending the decoding of a coded picture with nuh\_layer\_id greater than 0 specified in subclause H.8.1.3 is invoked.
   * + 1. Decoding process for inter-layer reference picture set

Outputs of this process are updated lists of aligned inter-layer reference pictures RefPicSetInterLayer0 and RefPicSetInterLayer1 and the variables NumActiveRefLayerPics0 and NumActiveRefLayerPics1.

The variable currLayerId is set equal to nuh\_layer\_id of the current decoded picture

The lists RefPicSetInterLayer0 and RefPicSetInterLayer1 are first emptied, NumActiveRefLayerPics0 and NumActiveRefLayerPics1 are set equal to 0 and the following applies:

for( i = 0; i < NumActiveRefLayerPics; i++ ) {  
 if( there is a picture picX in the DPB that is in the same access unit as the current picture and has  
 nuh\_layer\_id equal to RefPicLayerId[ i ] ) {  
 an interlayer reference picture rsPic is derived by invoking the subclause H.8.1.4 with picX and  
 RefPicLayerId[ i ] given as inputs   
 if( ( ViewId[ nuh\_layer\_id ] <= ViewId[ 0 ]  &&  
 ViewId[ nuh\_layer\_id ] <= ViewId[ RefPicLayerId[ i ] ] ) | |  
 ( ViewId[ nuh\_layer\_id ] >= ViewId[ 0 ] &&  
 ViewId[ nuh\_layer\_id ] >= ViewId[ RefPicLayerId[ i ] ] ) ) {  
 RefPicSetInterLayer0[ NumActiveRefLayerPics0 ] = rsPic  
 RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] is marked as "used for long-term reference"  
 } else {  
 RefPicSetInterLayer1[ NumActiveRefLayerPics1 ] = rsPic  
 RefPicSetInterLayer1[ NumActiveRefLayerPics1++ ] is marked as "used for long-term reference"  
 }  
 } else  
 RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] = "no reference picture"  
}

There shall be no entry equal to "no reference picture" in RefPicSetInterLayer0 or RefPicSetInterLayer1.

NOTE – For the profiles defined in Annex H, RefPicSetInterLayer1 is always empty since the value of ViewId[ i ] is equal to zero for all layers.

If the current picture is a RADL picture, there shall be no entry in the RefPicSetInterLayer0 or RefPicSetInterLayer1 that is a RASL picture.

NOTE – An access unit may contain both RASL and RADL pictures.

* + - 1. Marking process for ending the decoding of a coded picture with nuh\_layer\_id greater than 0

Output of this process is:

– a potentially updated marking as "used for short-term reference" for some decoded pictures.

The following applies.

for( i = 0; i < NumActiveRefLayerPics0; i++ )   
 RefPicSetInterLayer0[ i ] is marked as "used for short-term reference"

for( i = 0; i < NumActiveRefLayerPics1; i++ )  
 RefPicSetInterLayer1[ i ] is marked as "used for short-term reference"

* + - 1. Resampling process for inter layer reference pictures

Input to this process is:

– a decoded reference layer picture rlPic

– a variable rLId specifies the layer id of reference layer picture .

Output of this process is the resampled reference layer picture rsPic.

The variables PicWidthInSamplesY and PicHeightInSamplesY are set equal to pic\_width\_in\_luma\_samples and pic\_height\_in\_luma\_samples, respectively.

The variables RefLayerPicWidthInSamplesY and RefLayerPicHeightInSamplesY are set equal to the width and height of the decoded reference layer picture rlPic in units of luma samples, respectively. The variables RefLayerBitDepthY and RefLayerBitDepthC are set equal to BitDepthY and BitDepthC of the decoded reference layer picture rlPic, respectively.

The variables PicWidthInSamplesC, PicHeightInSamplesC, RefLayerPicWidthInSamplesC, and RefLayerPicHeightInSamplesC are derived as follows:

PicWidthInSamplesC = PicWidthInSamplesY / subWidthC (H‑11)  
PicHeightInSamplesC = PicHeightInSamplesY / subHeightC (H‑12)  
RefLayerPicWidthInSamplesC = RefLayerPicWidthInSamplesY / subWidthC (H‑13)  
RefLayerPicHeightInSamplesC = RefLayerPicHeightInSamplesY / subHeightC (H‑14)

The variable currLayerId is set equal to nuh\_layer\_id of the current picture..

The variables ScaledRefLayerLeftOffset, ScaledRefLayerTopOffset, ScaledRefLayerRightOffset and ScaledRefLayerBottomOffset are derived as follows:

ScaledRefLayerLeftOffset = scaled\_ref\_layer\_left\_offset[ rLId ] << 1 (H‑15)  
ScaledRefLayerTopOffset = scaled\_ref\_layer\_top\_offset[ rLId ] << 1 (H‑16)  
ScaledRefLayerRightOffset = scaled\_ref\_layer\_right\_offset[ rLId  ] << 1 (H‑17)  
ScaledRefLayerBottomOffset = scaled\_ref\_layer\_bottom\_offset[ rLId ] << 1 (H‑18)

The variables ScaledRefLayerPicWidthInSamplesY and ScaledRefLayerPicHeightInSamplesY are derived as follows:

ScaledRefLayerPicWidthInSamplesY = PicWidthInSamplesY –   
 ScaledRefLayerLeftOffset  – ScaledRefLayerRightOffset (H‑19)  
ScaledRefLayerPicHeightInSamplesY = PicHeightInSamplesY –   
 ScaledRefLayerTopOffset – ScaledRefLayerBottomOffset (H‑20)

The variables ScaleFactorX and ScaleFactorY are derived as follows:

ScaleFactorX = ( ( RefLayerPicWidthInSamplesY << 16 ) + ( ScaledRefLayerPicWidthInSamplesY >> 1 ) ) /   
 ScaledRefLayerPicWidthInSamplesY (H‑21)  
ScaleFactorY = ( ( RefLayerPicHeightInSamplesY << 16 ) + ( ScaledRefLayerPicHeightInSamplesY >> 1 ) ) /   
 ScaledRefLayerPicHeightInSamplesY (H‑22)

The following steps are applied to derive the resampled inter layer reference picture rsPic.

– if PicWidthInSamplesY is equal to RefLayerPicWidthInSamplesY, PicHeightInSamplesY is equal to RefLayerPicHeightInSamplesY, the values of ScaledRefLayerLeftOffset, ScaledRefLayerTopOffset, ScaledRefLayerRightOffset and ScaledRefLayerBottomOffset are all equal to 0, RefLayerBitDepthY is equal to BitDepthY, and RefLayerBitDepthC is equal to BitDepthC.

* + rsPic is set equal to rlPic.

– otherwise, rsPic is derived as follows:

* + The PicOrderCntVal value of rsPic is set equal to the PicOrderCntVal value of rlPic.
  + When VpsInterLayerSamplePredictionEnabled[ LayerIdxInVps[ currLayerId ] ][ LayerIdxInVps[ rLId ] ] is equal to 1, the picture sample resampling process as specified in subclause H.8.1.4.1 is invoked with the sample arrays, rlPicSampleL, rlPicSampleCb and rlPicSampleCr, of the reference layer picture rlPic as inputs, and with the sample arrays of the resampled picture rsPic as outputs.
  + When VpsInterLayerMotionPredictionEnabled[ LayerIdxInVps[ currLayerId ] ][ LayerIdxInVps[ rLId ] ] is equal to 1, the following steps apply:
    - A single slice rsSlice of the resampled picture rsPic is generated as follows:
      * The values of slice\_type, num\_ref\_idx\_l0\_active\_minus1 and num\_ref\_idx\_l1\_active\_minus1 for the generated slice rsSlice are inferred to be equal to the slice\_type, num\_ref\_idx\_l0\_active\_minus1 and num\_ref\_idx\_l1\_active\_minus1, respectively, of the first slice in rlPic
      * When rsSlice is a P or B slice, for i in the range of 0 to num\_ref\_idx\_l0\_active\_minus1 of rsSlice, inclusive, the reference picture with index i in reference picture list 0 of rsSlice is set equal to reference picture with index i in reference picture list 0 of the first slice of rlPic
      * When rsSlice is a B slice, for i in the range of 0 to num\_ref\_idx\_l1\_active\_minus1 of rsSlice, inclusive, the reference picture with index i in reference picture list 1 of rsSlice is set equal to reference picture with index i in reference picture list 1 of the first slice of rlPic

NOTE: When the resampled picture is used as collocated picture for temporal motion vector prediction, all slices of rlPic are constrained to have the same values of slice\_type, num\_ref\_idx\_l0\_active\_minus1 and num\_ref\_idx\_l1\_active\_minus1.

* + - The picture motion field resampling process as specified in subclause H.8.1.4.2 is invoked with reference layer picture rlPic and its motion field data including an array rlPredMode specifying the prediction modes of the reference layer picture, two arrays rlRefIdxLX specifying the reference indices of the reference layer picture, two arrays rlMvLX specifying the luma motion vectors of the reference layer picture and two arrays rlPredFlagLX specifying the prediction list utilization flags of the reference layer picture, with X = 0,1 as inputs, and with the motion field data of the resampled picture rsPic as output.
      * 1. Resampling process of picture sample values

Inputs to this process are:

– a ( RefLayerPicWidthInSamplesY ) x ( RefLayerPicHeightInSamplesY ) array rlPicSampleL of luma samples,

– a ( RefLayerPicWidthInSamplesC ) x ( RefLayerPicHeightInSamplesC ) array rlPicSampleCb of chroma samples of the component Cb,

– a ( RefLayerPicWidthInSamplesC ) x ( RefLayerPicHeightInSamplesC ) array rlPicSampleCr of chroma samples of the component Cr.

Outputs of this process are:

– a ( PicWidthInSamplesY ) x ( PicHeightInSamplesY ) array rsPicSampleL of luma samples,

– a (  PicWidthInSamplesC ) x ( PicHeightInSamplesC ) array rsPicSampleCb of chroma samples of the component Cb,

– a ( PicWidthInSamplesC) x ( PicHeightInSamplesC ) array rsPicSampleCr of chroma samples of the component Cr.

The luma sample array rsPicSampleL is derived by invoking the luma sample resampling process specified in subclause H.8.1.4.1.1 with the reference luma sample array rlPicSampleL given as input.

The chroma sample array rsPicSampleCb of the chroma component Cb is derived by invoking the chroma sample resampling process specified in subclause H.8.1.4.1.2 with the reference chroma sample array rlPicSampleCb given as input.

The chroma sample array rsPicSampleCr of the chroma component Cr is derived by invoking the chroma sample resampling process specified in subclause H.8.1.4.1.2 with the reference sample array rlPicSampleCr given as input.

Resampling process of luma sample values

Input to this process is the reference luma sample array rlPicSampleL.

Output of this process is the resampled luma sample array rsPicSampleL.

The variables leftStartL, rightEndL, topStartL, and bottomEndL are derived as follows:

leftStartL = ScaledRefLayerLeftOffset  
rightEndL = PicWidthInSamplesY – ScaledRefLayerRightOffset  
topStartL = ScaledRefLayerTopOffset  
bottomEndL = PicHeightInSamplesY – ScaledRefLayerBottomOffset

The luma samples rsPicSampleL [ xP ][ yP ] with ( xP = 0 ... PicWidthInSamplesY – 1, yP = 0 ... PicHeightInSamplesY – 1) are derived by invoking the luma sample interpolation process specified in subclause H.8.1.4.1.3 with rlPicSampleL and luma sample location ( Clip3( leftStartL, rightEndL – 1, xP ), Clip3( topStartL, bottomEndL – 1, yP ) ) given as inputs and rsPicSampleL[ xP ][ yP ] as output.

Resampling process of chroma sample values

Input to this process is the reference chroma sample array rlPicSampleC,

Output of this process is the resampled chroma sample array rsPicSampleC.

The variables leftStartC, rightEndC, topStartC, and bottomEndC are derived as follows:

leftStartC = ScaledRefLayerLeftOffset / SubWidthC  
rightEndC = ( PicWidthInSamplesY– ScaledRefLayerRightOffset ) / SubWidthC  
topStartC = ScaledRefLayerTopOffset / SubHeightC  
bottomEndC = ( PicHeightInSamplesY– ScaledRefLayerBottomOffset ) / SubHeightC

The chroma samples rsPicSampleC[ xPC ][ yPC ] with ( xPC = 0 ... PicWidthInSamplesC – 1, yPC = 0 ... PicHeightInSamplesC – 1) are derived by invoking the chroma sample interpolation process specified in subclause H.8.1.4.1.4 with rlPicSampleC and chroma sample location ( Clip3( leftStartC, rightEndC ‑ 1, xPC ), Clip3( topStartC, bottomEndC – 1, yPC ) ) given as inputs and rsPicSampleC[ xPC ][ yPC ] as output.

Luma sample interpolation process

Inputs to this process are

– the luma reference sample array rlPicSampleL,

– a luma sample location ( xP, yP ) relative to the top-left luma sample of the current picture.

Output of this process is a interpolated luma sample value intLumaSample.

Table H‑1 specifies the 8-tap filter coefficients fL[ p, x ] with p = 0 ... 15 and x = 0 ... 7 used for the luma resampling process.

Table H‑1 – 16-phase luma resampling filter

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| phase p | interpolation filter coefficients | | | | | | | |
| fL[ p, 0 ] | fL[ p, 1 ] | fL[ p, 2 ] | fL[ p, 3 ] | fL[ p, 4 ] | fL[ p, 5 ] | fL[ p, 6 ] | fL[ p, 7 ] |
| 0 | 0 | 0 | 0 | 64 | 0 | 0 | 0 | 0 |
| 1 | 0 | 1 | −3 | 63 | 4 | −2 | 1 | 0 |
| 2 | −1 | 2 | −5 | 62 | 8 | −3 | 1 | 0 |
| 3 | −1 | 3 | −8 | 60 | 13 | −4 | 1 | 0 |
| 4 | −1 | 4 | −10 | 58 | 17 | −5 | 1 | 0 |
| 5 | −1 | 4 | −11 | 52 | 26 | −8 | 3 | −1 |
| 6 | −1 | 3 | −9 | 47 | 31 | −10 | 4 | −1 |
| 7 | −1 | 4 | −11 | 45 | 34 | −10 | 4 | −1 |
| 8 | −1 | 4 | −11 | 40 | 40 | −11 | 4 | −1 |
| 9 | −1 | 4 | −10 | 34 | 45 | −11 | 4 | −1 |
| 10 | −1 | 4 | −10 | 31 | 47 | −9 | 3 | −1 |
| 11 | −1 | 3 | −8 | 26 | 52 | −11 | 4 | −1 |
| 12 | 0 | 1 | −5 | 17 | 58 | −10 | 4 | −1 |
| 13 | 0 | 1 | −4 | 13 | 60 | −8 | 3 | −1 |
| 14 | 0 | 1 | −3 | 8 | 62 | −5 | 2 | −1 |
| 15 | 0 | 1 | −2 | 4 | 63 | −3 | 1 | 0 |

The value of the interpolated luma sample IntLumaSample  is derived by applying the following ordered steps:

1. The derivation process for reference layer sample location used in resampling as specified in subclause H.6.2 is invoked with cIdx equal to 0 and luma sample location ( xP, yP ) given as the inputs and ( xRef16, yRef16 ) in units of 1/16-th sample as output.
2. The variables xRef and xPhase are derived as follows:

xRef     = ( xRef16 >> 4 ) (H‑23)

xPhase = ( xRef16 ) % 16 (H‑24)

1. The variables yRef and yPhase are derived as follows:

yRef     = ( yRef16 >> 4 ) (H‑25)

yPhase = ( yRef16 ) % 16 (H‑26)

1. The variables shift1, shift2 and offset are derived as follows:

shift1 = RefLayerBitDepthY – 8 (H‑27)

shift2 = 20 – BitDepthY (H‑28)

offset = 1 << ( shift2 – 1) (H‑29)

1. The sample value tempArray[ n ] with n = 0 … 7, is derived as follows:

yPosRL = Clip3( 0, RefLayerPicHeightInSamplesY – 1, yRef + n – 1 ) (H‑30)

refW      = RefLayerPicWidthInSamplesY

tempArray[ n ] = ( fL[ xPhase, 0 ] \* rlPicSampleL[ Clip3( 0, refW – 1, xRef – 3), yPosRL ] +  
 fL[ xPhase, 1 ] \* rlPicSampleL[ Clip3( 0, refW – 1, xRef – 2), yPosRL ] +  
 fL[ xPhase, 2 ] \* rlPicSampleL[ Clip3( 0, refW – 1, xRef – 1), yPosRL ] +  
 fL[ xPhase, 3 ] \* rlPicSampleL[ Clip3( 0, refW – 1, xRef      ), yPosRL ] +  
 fL[ xPhase, 4 ] \* rlPicSampleL[ Clip3( 0, refW – 1, xRef + 1), yPosRL ] + (H‑31)  
 fL[ xPhase, 5 ] \* rlPicSampleL[ Clip3( 0, refW – 1, xRef + 2), yPosRL ] +  
 fL[ xPhase, 6 ] \* rlPicSampleL[ Clip3( 0, refW – 1, xRef + 3), yPosRL ] +  
 fL[ xPhase, 7 ] \* rlPicSampleL[ Clip3( 0, refW – 1, xRef + 4), yPosRL ] ) >> shift1

1. The interpolated luma sample value intLumaSample is derived as follows:

intLumaSample = ( fL[ yPhase, 0 ] \* tempArray [ 0 ] +   
 fL[ yPhase, 1 ] \* tempArray [ 1 ] +   
 fL[ yPhase, 2 ] \* tempArray [ 2 ] +   
 fL[ yPhase, 3 ] \* tempArray [ 3 ] +   
 fL[ yPhase, 4 ] \* tempArray [ 4 ] + (H‑32)  
 fL[ yPhase, 5 ] \* tempArray [ 5 ] +   
 fL[ yPhase, 6 ] \* tempArray [ 6 ] +   
 fL[ yPhase, 7 ] \* tempArray [ 7 ] + offset ) >> shift2

intLumaSample = Clip3( 0, ( 1 << BitDepthY) – 1 , intLumaSample ) (H‑33)

Chroma sample interpolation process

Inputs to this process are:

– the chroma reference sample array rlPicSampleC,

– a chroma sample location ( xPC, yPC ) relative to the top-left chorma sample of the current picture.

Output of this process is a interpolated chroma sample value intChromaSample.

Table H‑2 specifies the 4-tap filter coefficients fC[ p, x ] with p = 0 ... 15 and x = 0 ... 3 used for the chroma resampling process.

Table H‑2 – 16-phase chroma resampling filter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| phase p | interpolation filter coefficients | | | |
| fC[ p, 0 ] | fC[ p, 1 ] | fC[ p, 2 ] | fC[ p, 3 ] |
| 0 | 0 | 64 | 0 | 0 |
| 1 | −2 | 62 | 4 | 0 |
| 2 | −2 | 58 | 10 | −2 |
| 3 | −4 | 56 | 14 | −2 |
| 4 | −4 | 54 | 16 | −2 |
| 5 | −6 | 52 | 20 | −2 |
| 6 | −6 | 46 | 28 | −4 |
| 7 | −4 | 42 | 30 | −4 |
| 8 | −4 | 36 | 36 | −4 |
| 9 | −4 | 30 | 42 | −4 |
| 10 | −4 | 28 | 46 | −6 |
| 11 | −2 | 20 | 52 | −6 |
| 12 | −2 | 16 | 54 | −4 |
| 13 | −2 | 14 | 56 | −4 |
| 14 | −2 | 10 | 58 | −2 |
| 15 | 0 | 4 | 62 | −2 |

The value of the interpolated chroma sample value intChromaSample is derived by applying the following ordered steps:

1. The derivation process for reference layer sample location in resampling as specified in subclause H.6.2 is invoked with cIdx and chroma sample location ( xPC, yPC ) given as the inputs and ( xRef16, yRef16 ) in units of 1/16-th sample as output.
2. The variables xRef and xPhase are derived as follows:

xRef     = ( xRef16 >> 4 ) (H‑34)

xPhase = ( xRef16 ) % 16 (H‑35)

1. The variables yRef and yPhase are derived as follows:

yRef     = ( yRef16 >> 4 ) (H‑36)

yPhase = ( yRef16 ) % 16 (H‑37)

1. The variables shift1, shift2 and offset are derived as follows:

shift1 = RefLayerBitDepthC – 8 (H‑38)

shift2 = 20 – BitDepthC (H‑39)

offset =1 << ( shift2 – 1) (H‑40)

1. The sample value tempArray[ n ] with n = 0 … 3, is derived as follows:

yPosRL = Clip3( 0 , RefLayerPicHeightInSamplesC – 1, yRef + n – 1 ) (H‑41)

refWC   = RefLayerPicWidthInSamplesC (H‑42)

tempArray[ n ] = ( fC[ xPhase, 0 ] \* rlPicSampleC[ Clip3( 0, refWC – 1, xRef – 1), yPosRL ] +  
 fC[ xPhase, 1 ] \* rlPicSampleC[ Clip3( 0, refWC – 1, xRef       ), yPosRL ] +  
 fC[ xPhase, 2 ] \* rlPicSampleC[ Clip3( 0, refWC – 1, xRef + 1 ), yPosRL ] + (H‑43)  
 fC[ xPhase, 3 ] \* rlPicSampleC[ Clip3( 0, refWC – 1, xRef + 2 ), yPosRL ]  ) >> shift1

1. The interpolated chroma sample value intChromaSample is derived as follows:

intChromaSample = (fC[ yPhase, 0 ] \* tempArray [ 0 ] +  
 fC[ yPhase, 1 ] \* tempArray [ 1 ] +  
 fC[ yPhase, 2 ] \* tempArray [ 2 ] + (H‑44)  
 fC[ yPhase, 3 ] \* tempArray [ 3 ] + offset ) >> shift2

intChromaSample = Clip3( 0, ( 1 << BitDepthC ) – 1 , intChromaSample ) (H‑45)

* + - * 1. Resampling process of picture motion field

Inputs to this process are:

– the decoded reference layer picture rlPic,

– a ( RefLayerPicWidthInSamplesY ) x ( RefLayerPicHeightInSamplesY ) array rlPredMode specifies the prediction modes of the reference layer picture,

– two ( RefLayerPicWidthInSamplesY ) x ( RefLayerPicHeightInSamplesY ) arrays rlRefIdxLX specify the reference indices of the reference layer picture, with X = 0,1,

– two ( RefLayerPicWidthInSamplesY ) x ( RefLayerPicHeightInSamplesY ) arrays rlMvLX specify the luma motion vectors of the reference layer picture, with X = 0,1,

– two ( RefLayerPicWidthInSamplesY ) x ( RefLayerPicHeightInSamplesY ) arrays rlPredFlagLX specify the prediction list utilization flags of the reference layer picture, with X = 0,1.

Outputs of this process are:

– a ( PicWidthInSamplesY ) x ( PicHeightInSamplesY ) array predMode specifies the prediction modes of the resampled picture,

– two ( PicWidthInSamplesY ) x ( PicHeightInSamplesY ) arrays refIdxLX specify the reference indexes of the resampled picture, with X = 0,1,

– two ( PicWidthInSamplesY ) x ( PicHeightInSamplesY ) arrays mvLX specify the luma motion vectors of the resampled picture, with X = 0,1,

* two ( PicWidthInSamplesY ) x ( PicHeightInSamplesY ) arrays predFlagLX specify the prediction list utilization flags of the resampled picture, with X = 0,1.

The motion data of each 16 x 16 prediction block of the resampled picture are derived by applying the following ordered steps with xPb = 0 ... ( ( PicWidthInSamplesY + 15 ) >> 4 ) − 1 and yPb = 0 … ( ( PicHeightInSamplesY + 15 ) >> 4) – 1:

– The top-left luma sample location of the (16 x 16) prediction block xP and yP are set equal to ( xPb << 4 ) and ( yPb << 4 ), respectively,

– The variables predMode[ xP ][ yP ], refIdxLX[ xP ][ yP ], mvLX[ xP ][ yP ] and predFlagLX[ xP ][ yP ], with X = 0,1, are derived by invoking inter layer motion parameters derivation process specified in subclause H.8.1.4.2.1 with the luma location ( xP, yP ), rlPredMode, rlRefIdxLX, rlMvLX and rlPredFlagLX, with X = 0,1, given as input.

Derivation process for inter layer motion parameters

Inputs to this process are

– a luma location ( xP, yP ) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the resampled picture,

– the reference layer prediction mode array rlPredMode,

– the reference layer reference index arrays rlRefIdxL0 and rlRefIdxL1,

– the reference layer motion vector arrays rlMvL0 and rlMvL1,

– the reference layer prediction list utilization flag arrays rlPredFlagL0 and rlPredFlagL1.

Outputs of this process are

– a derived prediction mode predMode[ xP ][ yP ],

– two derived motion vectors mvL0 and mvL1[ xP ][ yP ]

– two derived reference indices refIdxL0 and refIdxL1[ xP ][ yP ]

– two derived prediction list utilization flags predFlagL0 and predFlagL1[ xP ][ yP ].

predMode[ xP ][ yP ], mvLX[ xP ][ yP ], refIdxLX[ xP ][ yP ], and predFlagLX[ xP ][ yP ], with X = 0, 1, are derived as follows:

1. The center location (xPCtr, yPCtr) of the luma prediction block is derived as follows:

xPCtr = xP + 8 (H‑44)

yPCtr = yP + 8 (H‑45)

1. The derivation process for reference layer luma sample location specified in subclause H.6.1 is invoked with luma location ( xPCtr, yPCtr ) given as the inputs and ( xRef, yRef ) as output.
2. The collocated position (xRL, yRL) is derived as follows:

xRL = ( ( xRef + 4 ) >> 4 ) << 4 (H‑46)

yRL = ( ( yRef + 4 ) >> 4 ) << 4 (H‑47)

1. The prediction mode predMode[ xP ][ yP ] is derived as follows:

* If ( xRL < 0 ) or ( xRL >= RefLayerPicWidthInSamplesY ) or ( yRL < 0 ) or ( yRL >= RefLayerPicHeightInSamplesY ), predMode[ xP ][ yP ] is set equal to MODE\_INTRA.
* Otherwise, the following applies:

predMode[ xP ][ yP ] = rlPredMode[ xRL ][ yRL ] (H‑48)

1. mvL0[ xP ][ yP ], mvL1[ xP ][ yP ], refIdxL0[ xP ][ yP ], refIdxL1[ xP ][ yP ], predFlagL0[ xP ][ yP ] and predFlagL1[ xP ][ yP ] are derived as follows:

* If predMode[ xP ][ yP ] is equal to MODE\_INTER, the following applies
* For each X = 0, 1, the following applies:
  + - * + refIdxLX[ xP ][ yP ] and predFlagLX[ xP ][ yP ] are derived as follows:

refIdxLX[ xP ][ yP ] = rlRefIdxLX[ xRL ][ yRL ] (H‑49)

predFlagLX[ xP ][ yP ] = rlPredFlagLX[ xRL ][ yRL ] (H‑50)

* + - * + mvLX[ xP ][ yP ][ 0 ] is derived as follows:
        + If ScaledRefLayerPicWidthInSamplesY is not equal to RefLayerPicWidthInSamplesY, the following applies:

scaleFactorMVX = Clip3( −4096, 4095, ( ( ScaledRefLayerPicWidthInSamplesY << 8 ) + ( RefLayerPicWidthInSamplesY >> 1 ) ) / RefLayerPicWidthInSamplesY) (H‑51)

mvLX[ xP ][ yP ][0] = Clip3( −32768, 32767, Sign(scaleFactorMVX \*   
rlMvLX[ xRL ][ yRL ][ 0 ] ) \* ( ( Abs ( scaleFactorMVX \* rlMvLX[ xRL ][ yRL ][ 0 ] )  
 + 127 ) >> 8 ) ) (H‑52)

* + - * + Otherwise, the following applies:

mvLX[ xP ][ yP ][ 0 ] = rlMvLX[ xRL ][ yRL ][ 0 ] (H‑53)

* + - * + mvLX[ xP ][ yP ][ 1 ] is derived as follows:
        + If ScaledRefLayerPicHeightInSamplesY is not equal to RefLayerPicHeightInSamplesY, the following applies:

scaleFactorMVY = Clip3( −4096, 4095, ( ( ScaledRefLayerPicHeightInSamplesY << 8 ) + ( RefLayerPicHeightInSamplesY >> 1 ) ) / RefLayerPicHeightInSamplesY) (H‑54)

mvLX[ xP ][ yP ][ 1 ] = Clip3( −32768, 32767, Sign(scaleFactorMVY \*   
rlMvLX[ xRL ][ yRL ][ 1 ] ) \* ( ( Abs  ( scaleFactorMVY \* rlMvLX[ xRL ][ yRL ][ 1 ] )  
 + 127 ) >> 8 ) ) (H‑55)

* + - * + Otherwise, the following applies:

mvLX[ xP ][ yP ][ 1 ] = rlMvLX[ xRL ][ yRL ][ 1 ] (H‑56)

* Otherwise (predMode[ xP ][ yP ] is equal to MODE\_INTRA), the following applies:
* both components of mvL0[ xP ][ yP ] and mvL1[ xP ][ yP ] are set to 0, refIdxL0[ xP ][ yP ] and refIdxL1[ xP ][ yP ] are set to –1, predFlagL0[ xP ][ yP ] and predFlagL1[ xP ][ yP ] are set to 0.
  + 1. NAL unit decoding process

The specification in subclause 8.2 apply.

* + 1. Slice decoding processes
       1. Decoding process for picture order count

The specifications in subclause F.8.3.1 apply.

* + - 1. Decoding process for reference picture set

The specifications in subclause F.8.3.2 apply.

* + - 1. Decoding process for generating unavailable reference pictures

The specifications in subclause F.8.3.3 apply.

* + - 1. Decoding process for reference picture lists construction

This process is invoked at the beginning of the decoding process for each P or B slice.

Reference pictures are addressed through reference indices as specified in subclause 8.5.3.3.2. A reference index is an index into a reference picture list. When decoding a P slice, there is a single reference picture list RefPicList0. When decoding a B slice, there is a second independent reference picture list RefPicList1 in addition to RefPicList0.

At the beginning of the decoding process for each slice, the reference picture lists RefPicList0 and, for B slices, RefPicList1 are derived as follows:

The variable NumRpsCurrTempList0 is set equal to Max( num\_ref\_idx\_l0\_active\_minus1 + 1, NumPicTotalCurr ) and the list RefPicListTemp0 is constructed as follows:

rIdx = 0  
while( rIdx < NumRpsCurrTempList0 ) {  
 for( i = 0; i < NumPocStCurrBefore && rIdx < NumRpsCurrTempList0; rIdx++, i++ )  
 RefPicListTemp0[ rIdx ] = RefPicSetStCurrBefore[ i ]  
 for( i = 0; i < NumActiveRefLayerPics0; rIdx++, i++ )  
 RefPicListTemp0[ rIdx ] = RefPicSetInterLayer0[ i ]  
 for( i = NumDirectRefLayers[ nuh\_layer\_id ] – 1; i >= 0; i-- ) {  
 rLId = RefLayerId[ nuh\_layer\_id ][ i ]  
 if( ref\_layer\_rps\_present\_flag[ i ] )  
 for( j = 0; j < RefLayerNumPocStCurrBefore[ rLId ]; rIdx++, j++ )  
 RefPicListTemp0[ rIdx ] = RefLayerRefPicSetCurrBefore[ rLId ] ][ j ]  
 }  
 for( i = 0; i < NumPocStCurrAfter && rIdx < NumRpsCurrTempList0; rIdx++, i++ ) (H‑57)  
 RefPicListTemp0[ rIdx ] = RefPicSetStCurrAfter[ i ]  
 for( i = 0; i < NumPocLtCurr && rIdx < NumRpsCurrTempList0; rIdx++, i++ )  
 RefPicListTemp0[ rIdx ] = RefPicSetLtCurr[ i ]  
 for( i = 0; i < NumActiveRefLayerPics1; rIdx++, i++ )  
 RefPicListTemp0[ rIdx ] = RefPicSetInterLayer1[ i ]  
 for( i = NumDirectRefLayers[ nuh\_layer\_id ] – 1; i >= 0; i-- ) {  
 rLId = RefLayerId[ nuh\_layer\_id ][ i ]  
 if( ref\_layer\_rps\_present\_flag[ i ] )  
 for( j = 0; j < RefLayerNumPocStCurrAfter[ rLId ]; rIdx++, j++ )  
 RefPicListTemp0[ rIdx ] = RefLayerRefPicSetCurrAfter[ rLId ] ][ j ]  
 }  
}

The list RefPicList0 is constructed as follows:

for( rIdx = 0; rIdx <= num\_ref\_idx\_l0\_active\_minus1; rIdx++) (H‑58)  
 RefPicList0[ rIdx ] = ref\_pic\_list\_modification\_flag\_l0 ? RefPicListTemp0[ list\_entry\_l0[ rIdx ] ] :  
 RefPicListTemp0[ rIdx ]

When the slice is a B slice, the variable NumRpsCurrTempList1 is set equal to Max( num\_ref\_idx\_l1\_active\_minus1 + 1, NumPicTotalCurr ) and the list RefPicListTemp1 is constructed as follows:

rIdx = 0  
while( rIdx < NumRpsCurrTempList1 ) {  
 for( i = 0; i < NumPocStCurrAfter && rIdx < NumRpsCurrTempList1; rIdx++, i++ )  
 RefPicListTemp1[ rIdx ] = RefPicSetStCurrAfter[ i ]  
 for( i = 0; i< NumActiveRefLayerPics1; rIdx++, i++ )  
 RefPicListTemp1[ rIdx ] = RefPicSetInterLayer1[ i ]  
 for( i = NumDirectRefLayers[ nuh\_layer\_id ] – 1; i >= 0; i-- ) {  
 rLId = RefLayerId[ nuh\_layer\_id ][ i ]  
 if( ref\_layer\_rps\_present\_flag[ i ] )  
 for( j = 0; j < RefLayerNumPocStCurrAfter[ rLId ]; rIdx++, j++ )  
 RefPicListTemp0[ rIdx ] = RefLayerRefPicSetCurrAfter[ rLId ][ j ]  
 }  
 for( i = 0; i < NumPocStCurrBefore && rIdx < NumRpsCurrTempList1; rIdx++, i++ ) (H‑59)  
 RefPicListTemp1[ rIdx ] = RefPicSetStCurrBefore[ i ]  
 for( i = 0; i < NumPocLtCurr && rIdx < NumRpsCurrTempList1; rIdx++, i++ )  
 RefPicListTemp1[ rIdx ] = RefPicSetLtCurr[ i ]  
 for( i = 0; i< NumActiveRefLayerPics0; rIdx++, i++ )  
 RefPicListTemp1[ rIdx ] = RefPicSetInterLayer0[ i ]  
 for( i = NumDirectRefLayers[ nuh\_layer\_id ] – 1; i >= 0; i-- ) {  
 if( ref\_layer\_rps\_present\_flag[ i ] )  
 for( j = 0; j < RefLayerNumPocStCurrBefore[ rLId ]; rIdx++, j++ )  
 RefPicListTemp0[ rIdx ] = RefLayerRefPicSetCurrBefore[ rLId ][ j ]  
 }  
}

When the slice is a B slice, the list RefPicList1 is constructed as follows:

for( rIdx = 0; rIdx <= num\_ref\_idx\_l1\_active\_minus1; rIdx++) (H‑60)  
 RefPicList1[ rIdx ] = ref\_pic\_list\_modification\_flag\_l1 ? RefPicListTemp1[ list\_entry\_l1[ rIdx ] ] :  
 RefPicListTemp1[ rIdx ]

NOTE – Because motion vectors from aligned inter-layer reference pictures are constrained to be zero motion only, an SHVC encoder should disable temporal motion vector prediction for the current picture, by setting slice\_temporal\_mvp\_enabled\_flag to zero, when only aligned inter-layer reference pictures exist in the reference picture lists of all slices in the current picture. This avoids the need to send any additional syntax elements such as collocated\_from\_l0\_flag and collocated\_ref\_idx. [Ed. (JC && YY): Consider finding a better location for this note, for example in the semantics part of slice\_temporal\_mvp\_enabled\_flag.]

* + - 1. Decoding process for collocated picture and no backward prediction flag

The specifications in subclause F.8.3.5 apply.

* + 1. Decoding process for coding units coded in intra prediction mode

The specifications in subclause F.8.4 apply.

* + 1. Decoding process for coding units coded in inter prediction mode

The specifications in subclause F.8.5 apply with the following addtions.

It is a requirement of bitstream conformance that, for X being replaced by either 0 or 1, the variables mvLX[ 0 ] and mvLX[ 1 ] as an output of the subclause 8.5.3.1 shall be equal to 0 if the value of refIdxLX as an output of the subclause 8.5.3.1 corresponds to an aligned inter-layer reference picture or to a diagonal inter-layer reference picture for which the picture resampling process of picture sample values as specified in subclause H.8.1.4.1 has been applied. That is, in any conformant bitstream, for X being replaced by either 0 or 1, upon invoking the decoding process in subclause 8.5.3.1, the values of the syntax elements merge\_idx, mvp\_lX\_flag, ref\_idx\_lX, MvdLX, and mvd\_l1\_zero\_flag shall always result in zero values for mvLX[ 0 ] and mvLX[ 1 ] when the value of refIdxLX of the reference picture list RefPicListX indicates an aligned inter-layer reference picture or to a diagonal inter-layer reference picture for which the picture resampling process of picture sample values as specified in subclause H.8.1.4.1 has been applied. [Ed. (MH): It is an open issue whether to have these constraints on mvLX[ 0 ] and mvLX[ 1 ] being equal to 0 for resampled diagonal inter-layer reference pictures.]

It is a requirement of bitstream conformance that the collocated picture colPic, used for temporal motion vector prediction and derived by invoking the subclause 8.5.3.2.7, shall not be an inter-layer reference picture if the reference layer picture is coded using two or more slice segments, and any of the following conditions is true:

* + The slice segment header syntax element slice\_type of at least one of the slice segments of the reference layer picture is different from the slice segment header syntax element slice\_type of another slice segment of the reference layer picture;
  + The slice segment header syntax element, num\_ref\_idx\_lX\_active\_minus1, for X being replaced by either 0 or 1, of at least one of the slice segments of the reference layer picture is different from the slice segment header syntax element num\_ref\_idx\_lX\_active\_minus1, for X being replaced by either 0 or 1, of another slice segment of the reference layer picture.
  + The reference picture list, RefPicListX[ i ], for X being replaced by either 0 or 1, of at least one of the slice segments of the reference layer picture is different from the reference picture list RefPicListX[ i ], for X being replaced by either 0 or 1, of another slice segment of the reference layer picture.
    1. Scaling, transformation and array construction process prior to deblocking filter process

The specifications in subclause F.8.6 apply.

* + 1. In-loop filter process

The specifications in subclause F.8.7 apply.

* 1. Parsing process

The specifications in clause F.9 apply.

* 1. Specification of bitstream subsets

The specifications in clause F.10 apply.

* 1. Profiles, tiers, and levels
     1. Profiles
        1. General

TBD.

* + - 1. Scalable Main profile

Bitstreams containing output layer sets conforming to the Scalable Main profile shall obey the following constraints, with layerSetIdx being the layer set for an output layer set conforming to the Scalable Main profile:

– The base layer bitstream, which is derived by invoking the sub-bitstream extraction process as specified in subclause F.10 with tIdTarget equal 7 and with TargetDecLayerIdList containing only one nuh\_layer\_id value that is equal to 0 as inputs, shall obey the following constraints:

– The base layer bitstream shall obey all constraints of the Main profile specified in subclause A.3.2.

– SPSs of the base layer bitstream should have general\_profile\_idc equal to 1 or general\_profile\_compatibility\_flag[ 1 ] equal to 1. [Ed. (JB): Should we add this “should” condition? I copied the language from SVC. Shall this ‘should’ be a ‘shall’]

– The sub-bitstream that is derived by invoking the sub-bitstream extraction process as specified in subclause F.10 with tIdTarget equal 7 and with TargetDecLayerIdList containing the nuh\_layer\_id values of the layer set with the index layerSetIdx shall obey the following constraints:

– All active SPSs and active layer SPSs for the sub-bitstream shall have chroma\_format\_idc equal to 1 only.

– All active SPSs and active layer SPSs for the sub-bitstream shall have bit\_depth\_luma\_minus8 equal to 0 only.

– All active SPSs and active layer SPSs shall have bit\_depth\_chroma\_minus8 equal to 0 only.

– CtbLog2SizeY derived from any active SPS or active layer SPS for the sub-bitstream shall be in the range of 4 to 6, inclusive.

– The picture resampling process of picture sample values as specified in subclause H.8.1.4.1 shall not be invoked more than once for decoding of each particular picture and the resampling process of picture motion field as specified in subclause H.8.1.4.2 shall not be invoked more than once for decoding of each particular picture with nuh\_layer\_id included in the layer set layerSetIdx. When both picture sample values and picture motion field resampling processes are invoked for decoding of a particular picture, they shall be applied to the same reference layer picture.

– ScalabilityId[ j ][ smIdx ] shall be equal to 0 for any smIdx value not equal to 2 and for any value of j for which layer\_id\_included\_flag[ layerSetIdx ][ j ] is equal to 1.

– For a layer with layer id iNuhLId equal to any of nuh\_layer\_id included in the layer set layerSetIdx, the value of NumRefLayers[ iNuhLId ], which specifies the total number of direct and indirect dependent layers and is derived as in F.7.4.3.1, shall be less than or equal to 7.

– All active SPSs shall have sps\_extension\_type\_flag[ i ] equal to 0 only for i equal to 0, and in the range of 2 to 6, inclusive.

* + 1. Tiers and levels

TBD

* 1. Byte stream format

The specifications in subclause F.12 apply.

* 1. Hypothetical reference decoder

The specifications in subclause F.13 and its subclauses apply.

* 1. SEI messages

The specifications in Annex D and subclause F.14 and its subclauses apply.

* 1. Video usability information

The specifications in Annex F.15 apply.