|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  17th Meeting: Valencia, ES, 27 March – 4 April 2014 | Document: JCTVC-Q1008\_v5 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **High efficiency video coding (HEVC) scalable extension Draft 6** | | |
| *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  Gary J. Sullivan, Microsoft  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)  [garysull@microsoft.com](mailto:garysull@microsoft.com)  [yekuiw@qti.qualcomm.com](mailto:yekuiw@qti.qualcomm.com) |
| *Source:* | Editors | | |

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

# Abstract

This document contains a preliminary version of SHVC Draft 6 text.

Remarks:

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 6) (changes to [JCTVC-P1008](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=8839) version 3)

* ----------- Release v4 -----------
* (Review GS01): editorial improvement for MCTS SEI message and chroma resampling filter SEI message
* (Review JC07 ~ 10): editorial improvement and bug fixes for chroma resampling filter SEI message
* (Review JB03): editorial improvement of chroma resampling filter SEI message
* (Proponent's update of chroma resampling filter SEI message)
* ----------- Release v3 -----------
* (Review YY05): small bug fixes and improvements in colour remapping SEI.
* (Review MH01): The semantics of the temporal motion-constrained tile sets SEI message were modified to apply layer-wise along the spirit of JCTVC-Q0183. The Scalable Main and Scalable Main 10 profile constraints were modified to concern the layers included in the derived sub-bitstream for the OLS indicated to conform to Scalable Main (10) in subclause H.11.1.2. Editorial fixes and improvements.
* (Review JC06): editorial improvements
* (Review YY05): editorial improvements to colour remapping SEI, etc.
* (Review YK00) : editorial improvements
* (Review JB): editorial improvements
* (Common HLS from [JCT3V-H1002](http://phenix.it-sudparis.eu/jct2/doc_end_user/current_document.php?id=2097)): Port common specifications for multi-layer extensions from JCT3V-H1004\_v5 and trace marks for the updates of common specifications are not recorded in this document. See [JCT3V-G1004](http://phenix.it-sudparis.eu/jct2/doc_end_user/current_document.php?id=1881) for the integration detail of each proposal.
* (Review JC05): addressing ticket #16, #19, #30, editorial improvement for knee function SEI, and other editorial improvements
* S13 (Hybrid scalability): Adding minor SHVC-specific change of Hybrid Scalabiblity
* S14 (Profile constraint on DependencyId): The Scalable Main, Scalable Main 10, and Stereo Main profiles against allowing layers with duplicate values of DependencyId when AuxId equal to 0
* (Review YY04): editorial fixes to the colour mapping process, editorial improvements to knee function SEI, etc
* ----------- Release v2 -----------
* (Review JB02): editorial improvements
* (Review JC04): editorial improvements
* (Review YY03): editorial improvements to the colour mapping process.
* (Review JB): editorial improvements to Deinterlaced picture info SEI message
* (Review YY02): editorial fixes.
* (Review JC03): editorial improvements for MCTS tile sets and editorial fixes for Q0200 integration and motion resampling invoking
* ([JCTVC-Q0078](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=8882), conformance for auxiliary picture)
* ([JCTVC-Q0078](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=8882), item 4): bug fix in the indexing of layer\_id\_included\_flag in H.11.1.2
* ([JCTVC-Q0117](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=8921)): 1-bit deinterlaced picture SEI message with field\_seq\_flag required to be zero
* ([JCTVC-Q0074](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=8878)): Colour Mapping Information SEI message
* (Chroma resampling filter SEI message 2): Proponent’s input for addressing editor’s note
* (Chroma resampling filter SEI message): Migration of Chroma resampling filter SEI message from RExt draft (JCTVC-P1005\_v4)
* (MCTS SEI message): Migration of MCTS SEI message from RExt draft (JCTVC-Q1005\_v1)
* (Knee Function SEI message): Migration of Knee Function SEI message from RExt draft (JCTVC-Q1005\_v1)
* ----------- Release v1 -----------
* (Review JC02): editorial fixes for Q0200 integration
* (Review JB01): editorial improvements
* (Review YY01): editorial improvements
* (Review JC01): editorial fixes
* ([JCTVC-Q0247](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=9081)): field\_frame\_info SEI message
* ([JCTVC-Q0200](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=9020)): use the cropping window parameters of the reference layer in the calculation of ScalingFactor and the starting phase position
* ([JCTVC-Q0120](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=8925)): introduce a calculated delta (which is calculated once per picture) for use in the reference sample location calculation
* ([JCTVC-Q0048](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=8852)): colour gamut scalability with asymmetric colour mapping LUT

Ed. Notes (Draft 5) (changes to [JCTVC-O1008](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=8527) version 3)

* ----------- Release v3 -----------
* (Review YY04): editorial improvement
* (Review MH01): editorial improvements (definitions, bitstream partition related text, etc.)
* (Review AR01): editorial improvement
* (Review JC03): editorial improvement
* (Definition of spatial and quality scalability): add the definition of spatial and quality scalability
* (Common HLS from JCT3V-G1004\_v5): Port common specifications for multi-layer extensions from JCT3V-G1004\_v5 and trace marks for the updates of common specifications are not recorded in this document. See [JCT3V-G1004](http://phenix.it-sudparis.eu/jct2/doc_end_user/current_document.php?id=1881) for the integration detail of each proposal.
* ----------- Release v2 -----------
* (Review YY03): editorial fixes
* (Review JC02): editorial fixes and move the process in H.8.3.4 to F.8.3.4 which is shared by MV-HEVC and SHVC spec.
* (Review YY02): editorial fixes
* (Review JO01): Small wording consistency adjustment, removal of reference to non-existing subclause
* (Review GJS01): Note numbering fixes and typographical issues (esp. spaces before left brackets on array indexes, uses of non-breaking hyphens and dashes instead of minus signs, and "smart" quote marks).
* ----------- Release v1 -----------
* ([JCTVC-P0134](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=8622)): Level constraints
* (Review JC01)
* (Review JB01)
* (Review YY01)
* (BL and EL bit depth restriction): EL bit depth shall not be smaller than BL bit depth
* (Scalable Main 10 profile): Profile for up to 10 bits in each layer
* ([JCTVC-P0312](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=8826)): vertical phase adjustment for resampling process
* (Fix Ticket #14): Maximum layer id to be 62
* (Fix Ticket #11): remove "active layer SPS" in definistion of Scalable Main profile

Ed. Notes (Draft 4) (changes to [JCTVC-N1008](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=8148))

* --------- Release v3 -----------
* Accepted all change marks.
* ----------- Release v2 -----------
* (Ticket Fix): #3, #4
* (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 are not recorded in this document. See [JCT3V-F1004](http://phenix.it-sudparis.eu/jct2/doc_end_user/current_document.php?id=1631) for the integration detail of each proposal
* ----------- Release v1 -----------
* (Review MH02)
* (Review MH01):
  + editorial improvements and fixes, editor's notes
  + Make sure that for all profile constraints the following decision is obeyed: ([JCTVC-O0253](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=8371), 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](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=8206), Scalable Main profile decision 2): layer number in any dependency layer chain shall be less than or equal to 8
* ([JCTVC-O0253](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=8371), Scalable Main profile decision 1): profile constraints apply to an output layer set
* ([JCTVC-O0216](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=8334)): Slice information derivation for inter-layer reference picture
* ([JCTVC-O0215](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=8333)): Signaling a flag to specify the phase alignment between layers (zero or center phase shift) for upsampling process
* ([JCTVC-O0199](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=8316)): 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](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=8311)): 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](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=8409) tables 2 and 3, first column

Ed. Notes (Draft 3) (changes to [JCTVC-N0242](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=7958))

* ----------- 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](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=7876)): 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](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=7824)): 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](http://phenix.it-sudparis.eu/jct2/doc_end_user/current_document.php?id=1354) for the integration detail of each proposal
* (Motion mapping text completion): Picture and slice level information derivation for resampled interlayer reference picture
* ([JCTVC-N0214](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=7930)):Intermediate data dynamic range control for the cases of 10-bits or higher input
* ([JCTVC-N0139](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=7855)): Adding a rounding offset for the colocated position derivation in reference layer motion derivation
* ([JCTVC-N0111](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=7827)): Using scaling factor to calculate the rounding offset for reference layer sample location derivation

Ed. Notes (JCTVC-N0242) (changes to [JCTVC-M1008](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=7751))

* ----------- Editorial improvement of Wording Draft 2 (submmited as to [JCTVC-N0242](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=7958))
* (Restructured Annexes) Annex F contain common parts of MV-HEVC and SHVC, Annex H contain SHVC specific text

Ed. Notes (Working Draft 2) (changes to [JCTVC-L1008](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=7279))

* ----------- 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](http://phenix.it-sudparis.eu/jct2/doc_end_user/current_document.php?id=1028) for the integration detail of each proposal
* ----------- Release v2 -----------
* Modifications to [JCTVC-M0309](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=7560): 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](http://phenix.it-sudparis.eu/jct2/doc_end_user/current_document.php?id=1028) 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](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=7520)): 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](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=7560)): scaled reference layer picture offsets
  + ([JCTVC-M0274](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=7525)): inter-layer referencing outside of conformance cropping window
  + ([JCTVC-M0449](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=7725) (JCTVC-M0188, JCTVC-M0322 and JCTVC-M0425)): division-free reference layer sample location derivation used in re-sampling process
  + ([JCTVC-M0133](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=7384)): the division-free reference layer sample location derivation
  + ([JCTVC-M0133](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=7384)): 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](http://phenix.it-sudparis.eu/jct2/doc_end_user/current_document.php?id=1028) for the integration detail for of each proposal

**CONTENTS**

*Page*

[Abstract i](#_Toc389494725)

[7.4.2.4.4 Order of NAL units and coded pictures and their association to access units 3](#_Toc389494726)

[8 Decoding process 4](#_Toc389494727)

[8.1 General decoding process 4](#_Toc389494728)

[8.1.1 Decoding process for a coded picture with nuh\_layer\_id equal to 0 6](#_Toc389494729)

[8.2 NAL unit decoding process 7](#_Toc389494730)

[8.3 Slice decoding process 7](#_Toc389494731)

[8.3.1 Decoding process for picture order count 7](#_Toc389494732)

[8.3.2 Decoding process for reference picture set 8](#_Toc389494733)

[8.3.3 Decoding process for generating unavailable reference pictures 12](#_Toc389494734)

[8.3.3.1 General decoding process for generating unavailable reference pictures 12](#_Toc389494735)

[A.3.6 Monochrome 8 profile 13](#_Toc389494736)

[Annex C Hypothetical reference decoder 14](#_Toc389494737)

[C.1 General 14](#_Toc389494738)

[C.2 Operation of coded picture buffer (CPB) and bitstream partition buffer (BPB) 21](#_Toc389494739)

[C.2.1 General 21](#_Toc389494740)

[C.2.2 Timing of decoding unit arrival 21](#_Toc389494741)

[C.2.3 Timing of decoding unit removal and decoding of decoding unit 23](#_Toc389494742)

[C.3 Operation of the decoded picture buffer (DPB) 26](#_Toc389494743)

[C.3.1 General 26](#_Toc389494744)

[C.3.2 Removal of pictures from the DPB 27](#_Toc389494745)

[C.3.3 Picture output 28](#_Toc389494746)

[C.3.4 Current decoded picture marking and storage 28](#_Toc389494747)

[C.4 Bitstream conformance 28](#_Toc389494748)

[C.5 Decoder conformance 30](#_Toc389494749)

[C.5.1 General 30](#_Toc389494750)

[C.5.2 Operation of the output order DPB 31](#_Toc389494751)

[C.5.2.1 General 31](#_Toc389494752)

[C.5.2.2 Output and removal of pictures from the DPB 32](#_Toc389494753)

[C.5.2.3 Picture decoding, marking, additional bumping, and storage 33](#_Toc389494754)

[C.5.2.4 "Bumping" process 34](#_Toc389494755)

[C.6 Demultiplexing process for deriving a bitstream partition 34](#_Toc389494756)

[Annex D Supplemental enhancement information 36](#_Toc389494757)

[D.2.25 Temporal motion-constrained tile sets SEI message syntax 39](#_Toc389494758)

[D.2.26 Chroma resampling filter hint SEI message syntax 40](#_Toc389494759)

[D.2.27 Knee function information SEI message syntax 41](#_Toc389494760)

[D.2.28 Colour mapping information SEI message syntax 42](#_Toc389494761)

[D.2.29 Deinterlaced picture information SEI message syntax 43](#_Toc389494762)

[D.3.25 Temporal motion-constrained tile sets SEI message simantics 57](#_Toc389494763)

[D.3.26 Chroma resampling filter hint SEI message semantics 60](#_Toc389494764)

[D.3.27 Knee function information SEI message semantics 65](#_Toc389494765)

[D.3.28 Colour mapping information SEI message semantics 66](#_Toc389494766)

[D.3.29 Deinterlaced picture information SEI message semantics 69](#_Toc389494767)

[Annex E Video usability information 70](#_Toc389494768)

[E.2 VUI semantics 70](#_Toc389494769)

[E.2.1 VUI parameters semantics 70](#_Toc389494770)

[E.3.2 HRD parameters semantics 70](#_Toc389494771)

[Annex F Common specifications for multi-layer extensions 71](#_Toc389494772)

[F.1 Scope 71](#_Toc389494773)

[F.2 Normative references 71](#_Toc389494774)

[F.3 Definitions 71](#_Toc389494775)

[F.4 Abbreviations 73](#_Toc389494776)

[F.5 Conventions 73](#_Toc389494777)

[F.6 Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships 73](#_Toc389494778)

[F.7 Syntax and semantics 73](#_Toc389494779)

[F.7.1 Method of specifying syntax in tabular form 73](#_Toc389494780)

[F.7.2 Specification of syntax functions, categories, and descriptors 73](#_Toc389494781)

[F.7.3 Syntax in tabular form 73](#_Toc389494782)

[F.7.3.1 NAL unit syntax 73](#_Toc389494783)

[F.7.3.2 Raw byte sequence payloads and RBSP trailing bits syntax 74](#_Toc389494784)

[F.7.3.3 Profile, tier and level syntax 88](#_Toc389494785)

[F.7.3.4 Scaling list data syntax 88](#_Toc389494786)

[F.7.3.5 Colour mapping table syntax 88](#_Toc389494787)

[F.7.3.6 Supplemental enhancement information message syntax 89](#_Toc389494788)

[F.7.3.7 Slice segment header syntax 90](#_Toc389494789)

[F.7.3.8 Short-term reference picture set syntax 93](#_Toc389494790)

[F.7.3.9 Slice segment data syntax 93](#_Toc389494791)

[F.7.4 Semantics 93](#_Toc389494792)

[F.7.4.1 General 93](#_Toc389494793)

[F.7.4.2 NAL unit semantics 93](#_Toc389494794)

[F.7.4.3 Raw byte sequence payloads, trailing bits, and byte alignment semantics 97](#_Toc389494795)

[F.7.4.4 Profile, tier and level semantics 119](#_Toc389494796)

[F.7.4.5 Scaling list data semantics 120](#_Toc389494797)

[F.7.4.6 Colour mapping table semantics 120](#_Toc389494798)

[F.7.4.7 Supplemental enhancement information message semantics 121](#_Toc389494799)

[F.7.4.8 Slice segment header semantics 121](#_Toc389494800)

[F.7.4.9 Short-term reference picture set semantics 125](#_Toc389494801)

[F.7.4.10 Slice segment data semantics 126](#_Toc389494802)

[F.8 Decoding process 126](#_Toc389494803)

[F.8.1 General decoding process 126](#_Toc389494804)

[F.8.1.1 Decoding process for a coded picture with nuh\_layer\_id equal to 0 128](#_Toc389494805)

[F.8.1.2 Decoding process for a coded picture with nuh\_layer\_id greater than 0 128](#_Toc389494806)

[F.8.1.3 Decoding process for starting the decoding of a coded picture with nuh\_layer\_id greater than 0 128](#_Toc389494807)

[F.8.1.4 Decoding process for ending the decoding of a coded picture with nuh\_layer\_id greater than 0 129](#_Toc389494808)

[F.8.1.5 Generation of unavailable reference pictures for pictures first in decoding order within a layer 129](#_Toc389494809)

[F.8.2 NAL unit decoding process 130](#_Toc389494810)

[F.8.3 Slice decoding processes 130](#_Toc389494811)

[F.8.3.1 Decoding process for picture order count 130](#_Toc389494812)

[F.8.3.2 Decoding process for reference picture set 132](#_Toc389494813)

[F.8.3.3 Decoding process for generating unavailable reference pictures 132](#_Toc389494814)

[F.8.3.4 Decoding process for reference picture lists construction 132](#_Toc389494815)

[F.8.4 Decoding process for coding units coded in intra prediction mode 133](#_Toc389494816)

[F.8.5 Decoding process for coding units coded in inter prediction mode 133](#_Toc389494817)

[F.8.6 Scaling, transformation and array construction process prior to deblocking filter process 133](#_Toc389494818)

[F.8.7 In-loop filter process 133](#_Toc389494819)

[F.9 Parsing process 133](#_Toc389494820)

[F.10 Specification of bitstream subsets 133](#_Toc389494821)

[F.10.1 Sub-bitstream extraction process 133](#_Toc389494822)

[F.10.2 Non-base layer subtree extraction process 134](#_Toc389494823)

[F.11 (Void) 135](#_Toc389494824)

[F.12 Byte stream format 135](#_Toc389494825)

[F.13 Hypothetical reference decoder 135](#_Toc389494826)

[F.14 SEI messages 135](#_Toc389494827)

[F.14.1 SEI message syntax 136](#_Toc389494828)

[F.14.1.1 Layers not present SEI message syntax 136](#_Toc389494829)

[F.14.1.2 Inter-layer constrained tile sets SEI message syntax 137](#_Toc389494830)

[F.14.1.3 Bitstream partition nesting SEI message syntax 137](#_Toc389494831)

[F.14.1.4 Bitstream partition initial arrival time SEI message syntax 138](#_Toc389494832)

[F.14.1.5 Bitstream partition HRD parameters SEI message syntax 138](#_Toc389494833)

[F.14.1.6 Sub-bitstream property SEI message syntax 139](#_Toc389494834)

[F.14.1.7 Alpha channel information SEI message syntax 139](#_Toc389494835)

[F.14.1.8 Overlay information SEI message syntax 139](#_Toc389494836)

[F.14.1.9 Temporal motion vector prediction constraints SEI message syntax 140](#_Toc389494837)

[F.14.1.10 Frame-field information SEI message syntax 140](#_Toc389494838)

[F.14.1.11 OLS nesting SEI message syntax 141](#_Toc389494839)

[F.14.1.12 VPS rewriting SEI message syntax 141](#_Toc389494840)

[F.14.2 SEI message semantics 142](#_Toc389494841)

[F.14.2.1 Layers not present SEI message semantics 143](#_Toc389494842)

[F.14.2.2 Inter-layer constrained tile sets SEI message semantics 143](#_Toc389494843)

[F.14.2.3 Bitstream partition nesting SEI message semantics 145](#_Toc389494844)

[F.14.2.4 Bitstream partition initial arrival time SEI message semantics 145](#_Toc389494845)

[F.14.2.5 Bitstream partition HRD parameters SEI message semantics 145](#_Toc389494846)

[F.14.2.6 Sub-bitstream property SEI message semantics 146](#_Toc389494847)

[F.14.2.7 Alpha channel information SEI message semantics 147](#_Toc389494848)

[F.14.2.8 Overlay information SEI message semantics 149](#_Toc389494849)

[F.14.2.9 Temporal motion vector prediction constraints SEI message semantics 150](#_Toc389494850)

[F.14.2.10 Frame-field information SEI message semantics 151](#_Toc389494851)

[F.14.2.11 OLS nesting SEI message semantics 152](#_Toc389494852)

[F.14.2.12 VPS rewriting SEI message semantics 152](#_Toc389494853)

[F.15 Video usability information 152](#_Toc389494854)

[F.15.1 General 152](#_Toc389494855)

[F.15.2 VUI syntax 152](#_Toc389494856)

[F.15.3 VUI semantics 152](#_Toc389494857)

[F.15.3.1 VUI parameters semantics 152](#_Toc389494858)

[F.15.3.2 HRD parameters semantics 153](#_Toc389494859)

[F.15.3.3 Sub-layer HRD parameters semantics 153](#_Toc389494860)

[Annex H Scalable high efficiency video coding 154](#_Toc389494861)

[H.1 Scope 154](#_Toc389494862)

[H.2 Normative references 154](#_Toc389494863)

[H.3 Definitions 154](#_Toc389494864)

[H.4 Abbreviations 154](#_Toc389494865)

[H.5 Conventions 154](#_Toc389494866)

[H.6 Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships 154](#_Toc389494867)

[H.6.1 Derivation process for reference layer sample location 154](#_Toc389494868)

[H.6.2 Derivation process for reference layer sample location used in resampling 154](#_Toc389494869)

[H.7 Syntax and semantics 155](#_Toc389494870)

[H.8 Decoding processes 155](#_Toc389494871)

[H.8.1 General decoding process 155](#_Toc389494872)

[H.8.1.1 Decoding process for a coded picture with nuh\_layer\_id greater than 0 155](#_Toc389494873)

[H.8.1.2 Decoding process for inter-layer reference picture set 155](#_Toc389494874)

[H.8.1.3 Marking process for ending the decoding of a coded picture with nuh\_layer\_id greater than 0 156](#_Toc389494875)

[H.8.1.4 Derivation process for inter-layer reference pictures 156](#_Toc389494876)

[H.8.2 NAL unit decoding process 168](#_Toc389494877)

[H.8.3 Slice decoding processes 168](#_Toc389494878)

[H.8.3.1 Decoding process for picture order count 168](#_Toc389494879)

[H.8.3.2 Decoding process for reference picture set 168](#_Toc389494880)

[H.8.3.3 Decoding process for generating unavailable reference pictures 168](#_Toc389494881)

[H.8.3.4 Decoding process for reference picture lists construction 168](#_Toc389494882)

[H.8.4 Decoding process for coding units coded in intra prediction mode 168](#_Toc389494883)

[H.8.5 Decoding process for coding units coded in inter prediction mode 168](#_Toc389494884)

[H.8.6 Scaling, transformation and array construction process prior to deblocking filter process 169](#_Toc389494885)

[H.8.7 In-loop filter process 169](#_Toc389494886)

[H.9 Parsing process 169](#_Toc389494887)

[H.10 Specification of bitstream subsets 169](#_Toc389494888)

[H.11 Profiles, tiers, and levels 169](#_Toc389494889)

[H.11.1 Profiles 169](#_Toc389494890)

[H.11.1.1 General 169](#_Toc389494891)

[H.11.1.2 Scalable Main and Scalable Main 10 profiles 169](#_Toc389494892)

[H.11.2 Tiers and levels 170](#_Toc389494893)

[H.11.2.1 Profile specific tier and level limits for the Scalable Main and Scalable Main 10 profiles 170](#_Toc389494894)

[H.12 Byte stream format 171](#_Toc389494895)

[H.13 Hypothetical reference decoder 171](#_Toc389494896)

[H.14 SEI messages 171](#_Toc389494897)

[H.15 Video usability information 171](#_Toc389494898)

*Replace the definition of access unit in clause 3 with the following:*

**3.X 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 X – Pictures in the same access unit are associated with the same picture order count.

*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**: A *layer* of an *output layer set* that is output when TargetOlsIdx is equal to the index of the *output layer set*.

**3.X output layer set (OLS)**: A set of *layers* consisting of the *layers* of one of the specified *layer sets*, where one or more *layers* in the set of layers are indicated to be output layers.

**3.X output 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 *output layers*.

**3.X picture 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 a *coded picture* and their *associated non-VCL NAL units*.

**3.X target output layer set**: The *output layer set* for which the index is equal to TargetOlsIdx.

*Add the following abbreviations to clause 4:*

OLS Output Layer Set

*Add the definition of the following mathematical function to subclause 5.8:*

GetCurrMsb( cl, pl, pm, ml ) = 

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

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

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

NOTE 1 – 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.

NOTE 2 – All PPSs, regardless of their values of nuh\_layer\_id or TemporalId, share the same value space for pps\_pic\_parameter\_set\_id. In other words, a PPS with nuh\_layer\_id equal to X, TemporalId equal to Y, and pps\_pic\_parameter\_set\_id equal to A would update the previously received PPS with nuh\_layer\_id not equal to X, and/or TemporalId not equal to Y, and pps\_pic\_parameter\_set\_id equal to A.

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 3 – 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.

NOTE 4 – All SPSs, regardless of their values of 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.

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 5 – 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 6 – 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 subclauses 7.4.2.4.4 with the following (with differences indicated in turquois):*

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

This subclause specifies the order of NAL units and coded pictures and their association to access units for CVSs that conform to one or more of the profiles specified in Annex A and that are decoded using the decoding process specified in clauses 2 through 10.

NOTE 1 – The association of NAL units to access units may differ depending on which profiles the CVSs conform to or which decoding process is used for decoding the bitstream.

An access unit consists of one coded picture with nuh\_layer\_id equal to 0, zero or more VCL NAL units with nuh\_layer\_id greater than 0, and zero or more non-VCL NAL units. The association of VCL NAL units to coded pictures is described in subclause 7.4.2.4.5.

The first access unit in the bitstream starts with the first NAL unit of the bitstream.

The first of any of the following NAL units after the last VCL NAL unit of a coded picture with nuh\_layer\_id equal to 0 specifies the start of a new access unit: [Ed. (YK): Within an access unit at least an SPS, PPS, or prefix SEI NAL unit may be present after the last VCL NAL unit of the picture with nuh\_layer\_id equal to 0. Maybe other NAL units as well. Check the need of more changes.]

– access unit delimiter NAL unit (when present),

– VPS NAL unit (when present),

– SPS NAL unit (when present),

– PPS NAL unit (when present),

– Prefix SEI NAL unit (when present),

– NAL units with nal\_unit\_type in the range of RSV\_NVCL41..RSV\_NVCL44 (when present), [Ed. (MH): It should be clarified if only the NAL units with nuh\_layer\_id equal to 0 are considered here or if nuh\_layer\_id can have any value in this condition. (YK): Also for other NAL units that may have nuh\_layer\_id greater than 0.]

– NAL units with nal\_unit\_type in the range of UNSPEC48..UNSPEC55 (when present),

– first VCL NAL unit of a coded picture with nuh\_layer\_id equal to 0 (always present).

The order of the coded pictures and non-VCL NAL units within an access unit shall obey the following constraints:

– When an access unit delimiter NAL unit is present, it shall be the first NAL unit. There shall be at most one access unit delimiter NAL unit in any access unit.

– When any prefix SEI NAL units are present, they shall not follow the last VCL NAL unit of the access unit.

– NAL units having nal\_unit\_type equal to FD\_NUT or SUFFIX\_SEI\_NUT, or in the range of RSV\_NVCL45..RSV\_NVCL47 or UNSPEC56..UNSPEC63 shall not precede the first VCL NAL unit of the coded picture with nuh\_layer\_id equal to 0.

– When an end of sequence NAL unit is present, it shall be the last NAL unit in the access unit other than an end of bitstream NAL unit (when present).

– When an end of bitstream NAL unit is present, it shall be the last NAL unit in the access unit.

NOTE 2 – VPS NAL units, SPS NAL units, PPS NAL units, prefix SEI NAL units, or NAL units with nal\_unit\_type in the range of RSV\_NVCL41..RSV\_NVCL44 or UNSPEC48..UNSPEC55, may be present in an access unit, but cannot follow the last VCL NAL unit of the coded picture within the access unit, as this condition would specify the start of a new access unit.

The structure of access units not containing any NAL units with nal\_unit\_type equal to FD\_NUT, VPS\_NUT, SPS\_NUT, PPS\_NUT, RSV\_VCL\_N10, RSV\_VCL\_R11, RSV\_VCL\_N12, RSV\_VCL\_R13, RSV\_VCL\_N14, or RSV\_VCL\_R15, RSV\_IRAP\_VCL22, or RSV\_IRAP\_VCL23, or in the range of RSV\_VCL24..RSV\_VCL31, RSV\_NVCL41..RSV\_NVCL47, or UNSPEC48..UNSPEC63 is shown in Figure 8‑1.



Figure 8‑1 – Structure of an access unit not containing any NAL units with nal\_unit\_type equal to FD\_NUT, SUFFIX\_SEI\_NUT, VPS\_NUT, SPS\_NUT, PPS\_NUT, RSV\_VCL\_N10, RSV\_VCL\_R11, RSV\_VCL\_N12, RSV\_VCL\_R13, RSV\_VCL\_N14, RSV\_VCL\_R15, RSV\_IRAP\_VCL22, or RSV\_IRAP\_VCL23, or in the range of RSV\_VCL24..RSV\_VCL31, RSV\_NVCL41..RSV\_NVCL47, or UNSPEC48..UNSPEC63

*Replace clause 8, subclauses 8.1, 8.2, 8.3, 8.3.1, 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 TargetOlsIdx, which specifies the index to the list of the OLSs specified by the VPS, of the target OLS, is specified as follows:

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

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

– Otherwise, TargetOlsIdx 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 = OlsIdxToLsIdx[ TargetOlsIdx ]  
 lsIdx = TargetDecLayerSetIdx  
 for( k = 0, j = 0; j < NumLayersInIdList[ lsIdx ]; j++ ) {  
 TargetDecLayerIdList[ j ] = LayerSetLayerIdList[ lsIdx ][ j ] (8‑1)  
 if( OutputLayerFlag[ TargetOlsIdx ][ 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 picture with HandleCraAsBlaFlag equal to 1, NoClrasOutputFlag is set equal to 1.

– Otherwise, if the current picture is an IDR picture with 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 0 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.

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 as specified 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 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 12).

Each coded picture is associated with a picture order count variable, denoted as PicOrderCntVal.

When the current picture is not an IRAP picture with NoRaslOutputFlag equal to 1, the variables prevPicOrderCntLsb and prevPicOrderCntMsb are derived as follows:

* Let prevTid0Pic be 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, and let PrevPicOrderCnt[ nuh\_layer\_id ] be the PicOrderCntVal of prevTid0Pic.
* 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.

The variable PicOrderCntMsb of the current picture is derived as follows:

* If the current picture is an IRAP picture with NoRaslOutputFlag equal to 1, PicOrderCntMsb is set equal to 0.
* Otherwise, PicOrderCntMsb is derived as follows:

if( ( slice\_pic\_order\_cnt\_lsb < prevPicOrderCntLsb ) &&  
 ( ( prevPicOrderCntLsb − slice\_pic\_order\_cnt\_lsb ) >= ( MaxPicOrderCntLsb / 2 ) ) )  
 PicOrderCntMsb = prevPicOrderCntMsb + MaxPicOrderCntLsb (8‑2)  
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 (8‑3)

NOTE 1 – All IDR pictures will have PicOrderCntVal equal to 0 since slice\_pic\_order\_cnt\_lsb is inferred to be 0 for IDR pictures and prevPicOrderCntLsb and prevPicOrderCntMsb are both set equal to 0.

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 shall not be the same.

The function PicOrderCnt( picX ) is specified as follows:

PicOrderCnt( picX ) = PicOrderCntVal of the picture picX (8‑4)

The function DiffPicOrderCnt( picA, picB ) is specified as follows:

DiffPicOrderCnt( picA, picB ) = PicOrderCnt( picA ) − PicOrderCnt( picB ) (8‑5)

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 2 – Let X be the current picture and Y and Z be two other pictures in the same CVS, 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.

### 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, all reference pictures with any value of nuh\_layer\_id currently in the DPB (if any) are marked as "unused for reference" when at least one of the following conditions is true:

– The current picture has NoClrasOutputFlag is equal to 1.

– The current picture activates a new VPS.

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 (8‑6)  
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" (8‑7)  
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" (8‑8)

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.
* There shall be no picture in the RPS that has discardable\_flag equal to 1.

### 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.

*Add a new subclause "A.3.6 Monochrome 8 profile".* [Ed (MH): If a RExt profile\_idc is used, the Monochrome 8 profile specification should be appended into A.3.5 instead.]

A.3.6 Monochrome 8 profile

Bitstreams conforming to the Monochrome 8 profile shall obey all constraints of the Main profile with the following exceptions:

* SPSs shall have chroma\_format\_idc equal to 0 only.

Conformance of a bitstream to the Monochrome 8 profile is indicated by general\_profile\_idc being equal to X or general\_profile\_compatibility\_flag[ X ] being equal to 1. [Ed. (JB): Consider whether to use the RExt profile\_idc or a separate profile\_idc for Monochrome 8 profile. This section requires further editorial changes after integration with RExt specification.]

*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 output operation point under test, denoted as TargetOp, is selected by selecting a value for TargetOlsIdx identifying a target OLS and selecting a target highest TemporalId value HighestTid. The value of TargetOlsIdx shall be in the range of 0 to NumOutputLayerSets − 1, inclusive. The value of HighestTid shall be in the range of 0 to MaxSubLayersInLayerSetMinus1[ OlsIdxToLsIdx [ TargetOlsIdx ] ], inclusive. The variables TargetDecLayerSetIdx, TargetOptLayerIdList, and TargetDecLayerIdList are then derived as specified by Equation 8‑1. The output 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 num\_sei\_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.

The variable MultiLayerCpbOperationFlag is derived as follows:

– If the selected buffering period SEI message is non-nested or is included in a scalable nesting SEI message that applies only to the sub-bitstream that contains only the base layer, MultiLayerCpbOperationFlag is set equal to 0.

– Otherwise, MultiLayerCpbOperationFlag is set equal to 1.

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\_minus1[ TargetDecLayerSetIdx ], 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. The variable initialAltParamSelectionFlag is derived as follows:

– If all of the following conditions are true, initialAltParamSelectionFlag is set equal to 1:

– The coded picture with nuh\_layer\_id equal to 0 in access unit 0 has nal\_unit\_type equal to CRA\_NUT or BLA\_W\_LP.

– MultiLayerCpbOperationFlag is equal to 0.

– irap\_cpb\_params\_present\_flag in the selected buffering period SEI message is equal to 1.

– Otherwise, if all of the following conditions are true, initialAltParamSelectionFlag is set equal to 1:

– The coded picture with nuh\_layer\_id equal to 0 in access unit 0 is an IRAP picture,

– MultiLayerCpbOperationFlag is equal to 1.

– irap\_cpb\_params\_present\_flag in the selected buffering period SEI message is equal to 1.

– Otherwise, initialAltParamSelectionFlag is set equal to 0.

– When initialAltParamSelectionFlag is equal to 1, the following applies:

– If the selected buffering period SEI message is included in a scalable nesting SEI message that applies at least to one sub-bitstream that contains more than one layer, a set of skipped leading pictures skippedPictureList consists of the CL-RAS pictures and the RASL pictures associated with the IRAP pictures with nuh\_layer\_id equal to nuhLayerId for which LayerInitializedFlag[ nuhLayerId ] is equal to 0 at the start of decoding the IRAP picture and for which nuhLayerId is among TargetDecLayerIdList. Otherwise (a buffering period SEI message is not nested in a scalable nesting SEI message), skippedPictureList consists of the RASL pictures associated with the coded picture with nuh\_layer\_id equal to 0 in access unit 0.

– Either of the following applies for selection of the initial CPB removal delay and delay offset:

[Ed. (JB): "Either of the following applies" language is unclear. How is it known which one(s) apply? (MH): This phrasing is from version 1. I suppose the intent is to let the HRD to pick either one of the following arbitrarily for its operation. (YK): The intent is to choose either one of the two first, and then to choose the other one, as each possible combination needs to be tested. However, this intent is indeed not clearly specified. We can try to improve it at the next editing session in Sapporo.]

– 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 all the pictures in skippedPictureList 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 output 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 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**

[Ed. (JO): Is the scheduler part of the Decoder? To me it seems that the decoder should start with the demultiplexer, and a single hypothetical bitstream partition scheduler (responsible in managing all partitions) should be a block in front of that (MH): The order of bitstream demultiplexer and HBPSs is correct in the figure. Bitstream partitions may be delivered by HBPSs to BPBs using different bitrates, hence the data streams are separated in the figure. The intent is also to indicate that the bitstream demultiplexer is optionally present and that the coded data may be readily organized in bitstream partitions rather than a united bitstream.]

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. 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[ TargetOlsIdx ][ layerIdx ][ HighestTid ] + 1, where layerIdx is equal to the value such that LayerSetLayerIdList[ TargetDecLayerSetIdx ][ layerIdx ] is equal to currLayerId.

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**

The variable altParamSelectionFlag is derived as follows:

– If all of the following conditions are true, altParamSelectionFlag is set equal to 1:

– The current picture is a BLA picture that has nal\_unit\_type equal to BLA\_W\_LP and nuh\_layer\_id equal to 0 or is a CRA picture that has nuh\_layer\_id equal to 0.

– MultiLayerCpbOperationFlag is equal to 0.

– Otherwise, if all of the following conditions are true, altParamSelectionFlag is set equal to 1:

– The current picture is an IRAP picture with nuh\_layer\_id equal to 0 and with NoClrasOutputFlag equal to 1.

– MultiLayerCpbOperationFlag is equal to 1.

– Otherwise, altParamSelectionFlag is set equal to 0.

When altParamSelectionFlag is equal to 1, 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, UseAltCpbParamsFlag is set equal to the value of use\_alt\_cpb\_params\_flag of the buffering period SEI message selected as specified in subclause C.1.

If SubPicHrdFlag is equal to 0, the variable subPicParamsFlag is set equal to 0, and the process 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 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 includes a BLA picture with nuh\_layer\_id equal to 0 and nal\_unit\_type equal to BLA\_W\_RADL or BLA\_N\_LP, MultiLayerCpbOperationFlag is equal to 0 and the value of irap\_cpb\_params\_present\_flag of the buffering period SEI message is equal to 1.

– Access unit 0 includes a BLA picture with nuh\_layer\_id equal to 0 and nal\_unit\_type equal to BLA\_W\_LP or includes a CRA picture with nuh\_layer\_id equal to 0, MultiLayerCpbOperationFlag is equal to 0, 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.

– Access unit 0 includes an IRAP picture with nuh\_layer\_id equal to 0, MultiLayerCpbOperationFlag is equal to 1 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 (present in BitstreamToDecode or available through external means not specified in this Specification) 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 includes a BLA picture with nuh\_layer\_id equal to 0 and nal\_unit\_type equal to BLA\_W\_RADL or BLA\_N\_LP, MultiLayerCpbOperationFlag is equal to 0 and the value of irap\_cpb\_params\_present\_flag of the buffering period SEI message is equal to 1.

– Access unit 0 includes a BLA picture with nuh\_layer\_id equal to 0 and nal\_unit\_type equal to BLA\_W\_LP or includes a CRA picture with nuh\_layer\_id equal to 0, MultiLayerCpbOperationFlag is equal to 0 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.

– Access unit 0 includes an IRAP picture with nuh\_layer\_id equal to 0, MultiLayerCpbOperationFlag is equal to 1 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 includes a BLA picture with nuh\_layer\_id equal to 0 and nal\_unit\_type equal to BLA\_W\_RADL or BLA\_N\_LP, MultiLayerCpbOperationFlag is equal to 0 and the value of irap\_cpb\_params\_present\_flag of the buffering period SEI message is equal to 1.

– Access unit includes a BLA picture with nuh\_layer\_id equal to 0 and nal\_unit\_type equal to BLA\_W\_LP or includes a CRA picture with nuh\_layer\_id equal to 0, MultiLayerCpbOperationFlag is equal to 0 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.

– Access unit n includes an IRAP picture with nuh\_layer\_id equal to 0, MultiLayerCpbOperationFlag is equal to 1, 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. 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 AltOptLayerFlag[ TargetOlsIdx ] is equal to 1 and an access unit either does not contain a picture at the output layer or contains a picture at the output layer that has PicOutputFlag equal to 0, the following ordered steps apply:

– The list nonOutputLayerPictures is the list of the pictures of the access unit with PicOutputFlag equal to 1 and with nuh\_layer\_id values among the nuh\_layer\_id values of the direct and indirect reference layers of the output layer.

– 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 an 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, with nuh\_layer\_id equal to currLayerId, 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 8.3.1 is invoked.

– The variable crossLayerBufferEmptyFlag is derived as follows:

– If a new VPS is activated by the current access unit or the current picture is an IRAP picture with nuh\_layer\_id equal to 0, NoRaslOutputFlag equal to 1, and NoClrasOutputFlag equal to 1, crossLayerBufferEmptyFlag is set equal to 1.

– Otherwise, crossLayerBufferEmptyFlag is set equal to 0.

– When the current picture is an IRAP picture with NoRaslOutputFlag equal to 1 and nuh\_layer\_id equal to 0, 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, NoOutputOfPriorPicsFlag is set equal to 1 (regardless of the value of no\_output\_of\_prior\_pics\_flag).

– Otherwise, if the value of pic\_width\_in\_luma\_samples, pic\_height\_in\_luma\_samples, chroma\_format\_idc, bit\_depth\_luma\_minus8, bit\_depth\_chroma\_minus8, separate\_colour\_plane\_flag, 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, chroma\_format\_idc, bit\_depth\_luma\_minus8, bit\_depth\_chroma\_minus8, separate\_colour\_plane\_flag, 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 equal 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, NoOutputOfPriorPicsFlag is set equal to no\_output\_of\_prior\_pics\_flag.

2. When the value of NoOutputOfPriorPicsFlag derived for the decoder under test is equal to 1, the following applies for the HRD:

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

– Otherwise (the value of crossLayerBufferEmptyFlag is equal to 0), all non-empty picture storage buffers that contain pictures with nuh\_layer\_id equal to 0 are emptied without output of the pictures they contain, and the sub-DPB fullness of the sub-DPB associated with nuh\_layer\_id equal to 0 is decremented by the number of picture storage buffers that were emptied.

– 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.

When vps\_base\_layer\_internal\_flag is equal to 0, all the following bitstream conformance constraints apply without considering pictures with nuh\_layer\_id equal to 0, for which there is no coded picture in the bitstream and the decoded pictures are provided by external means. [Ed. (YK): Check for possible wording improvements.] Ed. (JB): How about this?: When vps\_base\_layer\_internal\_flag is equal to 0, all the following bitstream conformance constraints apply only to coded pictures present in the bitstream, and do not apply to pictures with nuh\_layer\_id equal to 0 which are provided by external means.]

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 max\_vps\_dec\_pic\_buffering\_minus1[ TargetOlsIdx ][ layerIdx ][ HighestTid ], where layerIdx is equal to the value such that LayerSetLayerIdList[ TargetDecLayerSetIdx ][ layerIdx ] is equal to currPicLayerId.
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 by the VCL NAL units, appropriate buffering period, picture timing, and decoder unit information 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, and, when vps\_base\_layer\_internal\_flag is equal to 0, the decoded pictures with nuh\_layer\_id equal to 0 and their properties as specified in subclause F.8.1 are conveyed to the decoder in a timely manner 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). When vps\_base\_layer\_internal\_flag is equal to 0, decoded pictures with nuh\_layer\_id equal to 0 and their properties as specified in subclause F.8.1 are also conveyed both to the HRD and to the DUT in a timely manner by external means not specified in this Specification. 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. The flag BaseLayerOutputFlag and all flags BaseLayerPicOutputFlag output by the HRD shall also be output by the DUT, and the values that are output shall be equal to the values 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 need to be recalculated.

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. 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, CurrLayerMaxLatencyPictures, and MaxDecPicBufferingMinus1 are derived as follows:

– MaxNumReorderPics is set equal to max\_vps\_num\_reorder\_pics[ TargetOlsIdx ][ HighestTid ] of the active VPS.

– MaxLatencyIncreasePlus1 is set equal to the value of the syntax element max\_vps\_latency\_increase\_plus1[ TargetOlsIdx ][ HighestTid ] of the active VPS.

– CurrLayerMaxLatencyPictures is set equal to VpsMaxLatencyPictures[ TargetOlsIdx ][ HighestTid ] of the active VPS.

– MaxDecPicBufferingMinus1 is set equal to the value of the syntax element max\_vps\_dec\_pic\_buffering\_minus1[ TargetOlsIdx ][ layerIdx ][ HighestTid ] of the active VPS, where layerIdx is equal to the value such that LayerSetLayerIdList[ TargetDecLayerSetIdx ][ layerIdx ] is equal to currLayerId.

* + - 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, with nuh\_layer\_id equal to currLayerId, 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 and before the invocation of the decoding process for picture order count, happens instantaneously when the first decoding unit of the current picture is removed from the CPB and proceeds as follows:

– When the current picture is a POC resetting picture, all pictures in the DPB that do not belong to the current access unit and that are marked as "needed for output" are output, starting with pictures with the smallest value of PicOrderCntVal of all pictures excluding those in the current access unit in the DPB, in ascending order of the PicOrderCntVal values, and pictures with the same value of PicOrderCntVal are output in ascending order of the nuh\_layer\_id values. When a picture is output, it is 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".

– The decoding processes for picture order count and RPS are invoked. 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, the decoding processes for picture order count and RPS that are invoked are as specified in subclauses 8.3.1and 8.3.2, respectively. When decoding a CVS conforming to one or more of the profiles specified in Annex G or H using the decoding process specified in Annex F, and Annex G or H, the decoding processes for picture order count and RPS that are invoked are as specified in subclauses F.8.3.1 and F.8.3.2, respectively.

– The variable crossLayerBufferEmptyFlag is derived as follows:

– If a new VPS is activated by the current access unit or the current picture is IRAP picture with nuh\_layer\_id equal to 0, NoRaslOutputFlag equal to 1, and NoClrasOutputFlag equal to 1, crossLayerBufferEmptyFlag is set equal to 1.

– Otherwise, crossLayerBufferEmptyFlag is set equal to 0.

– If the current picture is an IRAP picture with NoRaslOutputFlag equal to 1 and nuh\_layer\_id equal to 0, 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, NoOutputOfPriorPicsFlag is set equal to 1 (regardless of the value of no\_output\_of\_prior\_pics\_flag).

– Otherwise, if the value of pic\_width\_in\_luma\_samples, pic\_height\_in\_luma\_samples, chroma\_format\_idc, bit\_depth\_luma\_minus8, bit\_depth\_chroma\_minus8, separate\_colour\_plane\_flag, 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, chroma\_format\_idc, bit\_depth\_luma\_minus8, bit\_depth\_chroma\_minus8, separate\_colour\_plane\_flag, 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 equal 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, NoOutputOfPriorPicsFlag is set equal to no\_output\_of\_prior\_pics\_flag.

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

– If NoOutputOfPriorPicsFlag is equal to 0, all non-empty picture storage buffers in all the sub-DPBs are output by repeatedly invoking the "bumping" process specified in subclause C.5.2.4 until all these pictures are marked as "not needed for output".

– Otherwise if crossLayerBufferEmptyFlag is equal to 1, all picture storage buffers in all the sub-DPBs are emptied, and the sub-DPB fullness of all the sub-DPBs is set equal to 0.

– Otherwise (crossLayerBufferEmptyFlag 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), all pictures that have nuh\_layer\_id equal to 0 in the sub-DPB containing the layer with nuh\_layer\_id equal to 0 are emptied, and the sub-DPB fullness of each sub-DPB is decremented by the number of picture storage buffers emptied in that sub-DPB.

– 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 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 unit that contains 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 CurrLayerMaxLatencyPictures.
* The number of pictures 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 AltOptLayerFlag[ TargetOlsIdx ] is equal to 1 and the current access unit either does not contain a picture at the output layer or contains a picture at the output layer that has PicOutputFlag equal to 0, the following ordered steps apply:

– The list nonOutputLayerPictures is the list of the pictures of the access unit with PicOutputFlag equal to 1 and with nuh\_layer\_id values among the nuh\_layer\_id values of the direct and indirect reference layers of the output layer.

– 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 an 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 unit that contains 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 CurrLayerMaxLatencyPictures.

* + - 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 and the fullness of the associated sub-DPB is decremented by one.
   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 Annex D as follows:*

1. Annex D  
     
   Supplemental enhancement information

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

*Modify subclause D.2.1 as follows:*

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

|  |  |
| --- | --- |
| sei\_payload( payloadType, payloadSize ) { | Descriptor |
| if( nal\_unit\_type = = PREFIX\_SEI\_NUT ) |  |
| if( payloadType = = 0 ) |  |
| **...** |  |
| else if( payloadType = = XXX ) |  |
| temporal\_motion\_constrained\_tile\_sets( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| chroma\_resampling\_filter\_hint( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| knee\_function\_info( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| colour\_remapping\_info( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| deinterlaced\_picture\_info( payloadSize ) |  |
| 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 ) |  |
| sub\_bitstream\_property( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| three\_dimensional\_reference\_displays\_info( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| depth\_representation\_info\_sei( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| multiview\_scene\_info( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| multiview\_acquisition\_info( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| multiview\_view\_position( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| alpha\_channel\_info( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| overlay\_info( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| temporal\_motion\_vector\_prediction\_constraints( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| frame\_field\_info( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| ols\_nesting( payloadSize ) |  |
| else if( payloadType = = XXX ) |  |
| vps\_rewriting( 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 D.2.2 as follows:*

|  |  |
| --- | --- |
| buffering\_period( payloadSize ) { | **Descriptor** |
| **bp\_seq\_parameter\_set\_id** | ue(v) |
| if( !sub\_pic\_hrd\_params\_present\_flag ) |  |
| **irap\_cpb\_params\_present\_flag** | u(1) |
| if( irap\_cpb\_params\_present\_flag ) { |  |
| **cpb\_delay\_offset** | u(v) |
| **dpb\_delay\_offset** | u(v) |
| } |  |
| **concatenation\_flag** | u(1) |
| **au\_cpb\_removal\_delay\_delta\_minus1** | u(v) |
| if( NalHrdBpPresentFlag ) { |  |
| for( i = 0; i <= CpbCnt; i++ ) { |  |
| **nal\_initial\_cpb\_removal\_delay[** i **]** | u(v) |
| **nal\_initial\_cpb\_removal\_offset[** i **]** | u(v) |
| if( sub\_pic\_hrd\_params\_present\_flag | | irap\_cpb\_params\_present\_flag ) { |  |
| **nal\_initial\_alt\_cpb\_removal\_delay[** i **]** | u(v) |
| **nal\_initial\_alt\_cpb\_removal\_offset[** i **]** | u(v) |
| } |  |
| } |  |
| } |  |
| if( VclHrdBpPresentFlag ) { |  |
| for( i = 0; i <= CpbCnt; i++ ) { |  |
| **vcl\_initial\_cpb\_removal\_delay[** i **]** | u(v) |
| **vcl\_initial\_cpb\_removal\_offset[** i **]** | u(v) |
| if( sub\_pic\_hrd\_params\_present\_flag | | irap\_cpb\_params\_present\_flag) { |  |
| **vcl\_initial\_alt\_cpb\_removal\_delay[** i **]** | u(v) |
| **vcl\_initial\_alt\_cpb\_removal\_offset[** i **]** | u(v) |
| } |  |
| } |  |
| } |  |
| if( payload\_extension\_present( ) ) |  |
| **use\_alt\_cpb\_params\_flag** | u(1) |
| } |  |

*Note: subclauses D.2.25, D.2.26, D.2.27, D2.28, D.2.29, D.3.25, D.3.26, D.3.27, D3.28 and D.3.29 are additional to MV-HEVC spec text.*

*Add subclauses D.2.25, D.2.26, D.2.27, D2.28 and D.2.29 as follows and the original subclause index D.2.25 is modified as D.2.30:*

[Ed. (JC): Subclause indices of D.2.25 ~ D.2.29 and D.3.25 ~ D.3.29 are determined based on version 1 specification without considering the new SEI messages in RExt specification. This requires further editorial changes after integration with RExt specification.]

D.2.25 Temporal motion-constrained tile sets SEI message syntax

|  |  |
| --- | --- |
| temporal\_motion\_constrained\_tile\_sets( payloadSize ) { | Descriptor |
| **mc\_all\_tiles\_exact\_sample\_value\_match\_flag** | u(1) |
| **each\_tile\_one\_tile\_set\_flag** | u(1) |
| if( !each\_tile\_one\_tile\_set\_flag ) { |  |
| **limited\_tile\_set\_display\_flag** | u(1) |
| **num\_sets\_in\_message\_minus1** | ue(v) |
| for( i = 0; i <= num\_sets\_in\_message\_minus1; i++) { |  |
| **mcts\_id**[ i ] | ue(v) |
| if( limited\_tile\_set\_display\_flag ) |  |
| **display\_tile\_set\_flag**[ i ] | u(1) |
| **num\_tile\_rects\_in\_set\_minus1**[ i ] | ue(v) |
| for( j = 0; j <= num\_tile\_rects\_in\_set\_minus1[ i ]; j++) { |  |
| **top\_left\_tile\_index**[i ][j ] | ue(v) |
| **bottom\_right\_tile\_index**[ i ][j ] | ue(v) |
| } |  |
| if( !mc\_all\_tiles\_exact\_sample\_value\_match\_flag ) |  |
| **mc\_exact\_sample\_value\_match\_flag**[ i ] | u(1) |
| **mcts\_tier\_level\_idc\_present\_flag**[ i ] | u(1) |
| if( mcts\_tier\_level\_idc\_present\_flag[ i ] ) { |  |
| **mcts\_tier\_flag**[ i ] | u(1) |
| **mcts\_level\_idc**[ i ] | u(8) |
| } |  |
| } |  |
| } else { |  |
| **max\_mcs\_tier\_level\_idc\_present\_flag** | u(1) |
| if( max\_mcts\_tier\_level\_idc\_present\_flag ) { |  |
| **max\_mcts\_tier\_flag** | u(1) |
| **max\_mcts\_level\_idc** | u(8) |
| } |  |
| } |  |
| } |  |

D.2.26 Chroma resampling filter hint SEI message syntax

|  |  |
| --- | --- |
| chroma\_resampling\_filter\_hint( payloadSize ) { | Descriptor |
| **ver\_chroma\_filter\_idc** | u(8) |
| **hor\_chroma\_filter\_idc** | u(8) |
| **ver\_filtering\_field\_processing\_flag** | u(1) |
| if( ver\_chroma\_filter\_idc = = 1 | | hor\_chroma\_filter\_idc = = 1 ) { |  |
| **target\_\_format\_idc** | ue(v) |
| if( ver\_chroma\_filter\_idc = = 1 ) { |  |
| **num\_vertical\_filters** | ue(v) |
| for( i = 0; i < num\_vertical\_filters; i++) { |  |
| **ver\_tap\_length\_minus1**[ i ] | ue(v) |
| for( j = 0; j <= ver\_tap\_length\_minus1[ i ]; j++) |  |
| **ver\_filter\_coeff**[ i ][ j ] | se(v) |
| } |  |
| } |  |
| if( hor\_chroma\_filter\_idc = = 1 ) { |  |
| **num\_horizontal\_filters** | ue(v) |
| for( i=0; i < num\_horizontal\_filters; i++) { |  |
| **hor\_tap\_length\_minus1**[ i ] | ue(v) |
| for( j=0; j <= hor\_tap\_length\_minus1[ i ]; j++) |  |
| **hor\_filter\_coeff**[ i ][ j ] | se(v) |
| } |  |
| } |  |
| } |  |
| } |  |

D.2.27 Knee function information SEI message syntax

|  |  |
| --- | --- |
| knee\_function\_info( payloadSize ) { | Descriptor |
| **knee\_function\_id** | ue(v) |
| **knee\_function\_cancel\_flag** | u(1) |
| if( !knee\_function\_cancel\_flag ) { |  |
| **knee\_function\_persistence\_flag** | u(1) |
| **mapping\_flag** | u(1) |
| **input\_d\_range** | u(32) |
| **input\_disp\_luminance** | u(32) |
| **output\_d\_range** | u(32) |
| **output\_disp\_luminance** | u(32) |
| **num\_knee\_points\_minus1** | ue(v) |
| for( i = 0; i <= num\_knee\_points\_minus1; i++ ) { |  |
| **input\_knee\_point**[ i ] | u(10) |
| **output\_knee\_point**[ i ] | u(10) |
| } |  |
| } |  |
| } |  |

D.2.28 Colour remapping information SEI message syntax

|  |  |
| --- | --- |
| colour\_remapping\_info( payloadSize ) { | Descriptor |
| **colour\_remap\_id** | ue(v) |
| **colour\_remap\_cancel\_flag** | u(1) |
| if( !colour\_remap\_cancel\_flag ) { |  |
| **colour\_remap\_persistence\_flag** | u(1) |
| **colour\_remap\_video\_signal\_type\_present\_flag** | u(1) |
| if(colour\_remap\_video\_signal\_type\_present\_flag ) { |  |
| **colour\_remap\_video\_full\_range\_flag** | u(1) |
| **colour\_remap\_primaries** | u(8) |
| **colour\_remap\_transfer\_characteristics** | u(8) |
| **colour\_remap\_matrix\_coeffs** | u(8) |
| } |  |
| **colour\_remap\_coded\_data\_bit\_depth** | u(5) |
| **colour\_remap\_target\_bit\_depth** | u(5) |
| **colour\_remap\_model\_id** | ue(v) |
| if( colour\_remap\_model\_id = = 0 ) { |  |
| for( c = 0; c < 3; c++ ) { |  |
| **pre\_lut\_num\_pivots\_minus1**[ c ] | u(8) |
| if( pre\_lut\_num\_pivots\_minus1[ c ] > 0 ) { |  |
| for( i = 0; i <= pre\_lut\_num\_pivots\_minus1[ c ]; i++ ) { |  |
| **pre\_lut\_coded\_pivot\_value**[ c ][ i ] | u(v) |
| **pre\_lut\_target\_pivot\_value**[ c ][ i ] | u(v) |
| } |  |
| } |  |
| } |  |
| **colour\_remap\_matrix\_present\_flag** | u(1) |
| if( colour\_remap\_matrix\_present\_flag ) { |  |
| **log2\_matrix\_denom** | u(4) |
| for( i = 0; i<3; i++) |  |
| for( j = 0; j<3; j++) |  |
| **colour\_remap\_coeffs**[ i ][ j ] | se(v) |
| } |  |
| for( c = 0; c < 3; c++ ) { |  |
| **post\_lut\_num\_pivots\_minus1**[ c ] | u(8) |
| if( post\_lut\_num\_pivots\_minus1[ c ] > 0 ) { |  |
| for( i = 0; i <= post\_lut\_num\_pivots\_minus1[ c ]; i++ ) { |  |
| **post\_lut\_coded\_pivot\_value**[ c ][ i ] | u(v) |
| **post\_lut\_target\_pivot\_value**[ c ][ i ] | u(v) |
| } |  |
| } |  |
| } |  |
| } |  |
| } |  |
| } |  |

D.2.29 Deinterlaced picture information SEI message syntax

|  |  |
| --- | --- |
| deinterlaced\_picture\_info( payloadSize ) { | Descriptor |
| **deinterlaced\_picture\_source\_parity\_flag** | u(1) |
| } |  |

*Modify Table D-1 as follows:*

Table D‑1 – Persistence scope of SEI messages (informative)

|  |  |
| --- | --- |
| SEI message | Persistence scope |
| Buffering period | The remainder of the bitstream |
| Picture timing | The access unit containing the SEI message |
| Pan-scan rectangle | Specified by the syntax of the SEI message |
| Filler payload | The access unit containing the SEI message |
| User data registered by Rec. ITU-T T.35 | Unspecified |
| User data unregistered | Unspecified |
| Recovery point | Specified by the syntax of the SEI message |
| Scene information | The access unit containing the SEI message and up to but not including the next access unit, in decoding order, that contains a scene information SEI message |
| Picture snapshot | The access unit containing the SEI message |
| Progressive refinement segment start | Specified by the syntax of the SEI message |
| Progressive refinement segment end | The access unit containing the SEI message |
| Film grain characteristics | Specified by the syntax of the SEI message |
| Post-filter hint | The access unit containing the SEI message |
| Tone mapping information | Specified by the syntax of the SEI message |
| Frame packing arrangement | Specified by the syntax of the SEI message |
| Display orientation | Specified by the syntax of the SEI message |
| Structure of pictures information | The set of access units in the CVS that correspond to entries listed in the SEI message |
| Decoded picture hash | The access unit containing the SEI message |
| Active parameter sets | The CVS containing the SEI message |
| Decoding unit information | The decoding unit containing the SEI message |
| Temporal sub-layer zero index | The access unit containing the SEI message |
| Scalable nesting | Depending on the nested SEI messages. Each nested SEI message has the same persistence scope as if the SEI message was not nested |
| Region refresh information | The set of VCL NAL units within the access unit starting from the VCL NAL unit following the SEI message up to but not including the VCL NAL unit following the next SEI NAL unit containing a region refresh information SEI message (if any) |
| Temporal motion-constrained tile sets | The CVS containing the SEI message |
| Chroma resampling filter hint | The CVS containing the SEI message |
| Knee function information | Specified by the syntax of the SEI message |
| Colour remapping information | Specified by the syntax of the SEI message |
| Deinterlaced picture information | One or more pictures of the access unit containing the SEI message |

*Modify the text after Table D-1 in subclause D.3.1 as follows:*

It is a requirement of bitstream conformance that when a prefix SEI message with payloadType equal to 17 (progressive refinement segment end) or 22 (post-filter hint) is present in an access unit, a suffix SEI message with the same value of payloadType shall not be present in the same access unit.

It is a requirement of bitstream conformance that the following restrictions apply on containing of SEI messages in SEI NAL units:

– An SEI NAL unit containing an active parameter sets SEI message shall contain only one active parameter sets SEI message and shall not contain any other SEI messages.

– When an SEI NAL unit contains a non-nested buffering period SEI message, a non-nested picture timing SEI message, or a non-nested decoding unit information SEI message, the SEI NAL unit shall not contain any other SEI message with payloadType not equal to 0 (buffering period), 1 (picture timing), or 130 (decoding unit information).

– When an SEI NAL unit contains a nested buffering period SEI message, a nested picture timing SEI message, or a nested decoding unit information SEI message, the SEI NAL unit shall not contain any other SEI message with payloadType not equal to 0 (buffering period), 1 (picture timing), 130 (decoding unit information), or 133 (scalable nesting).

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 on decoding order of SEI messages:

– When an SEI NAL unit containing an active parameter sets SEI message is present in an access unit, it shall be the first SEI NAL unit that follows the prevVclNalUnitInAu of the SEI NAL unit and precedes the nextVclNalUnitInAu of the SEI NAL unit.

– When a non-nested buffering period SEI message is present in an access unit, it shall not follow any other SEI message that follows the prevVclNalUnitInAu of the buffering period SEI message and precedes the nextVclNalUnitInAu of the buffering period SEI message, other than an active parameter sets SEI message.

– When a non-nested picture timing SEI message is present in an access unit, it shall not follow any other SEI message that follows the prevVclNalUnitInAu of the picture timing SEI message and precedes the nextVclNalUnitInAu of the picture timing SEI message, other than an active parameter sets SEI message or a non-nested buffering period SEI message.

– When a non-nested decoding unit information SEI message is present in an access unit, it shall not follow any other SEI message in the same access unit that follows the prevVclNalUnitInAu of the decoding unit information SEI message and precedes the nextVclNalUnitInAu of the decoding unit information SEI message, other than an active parameter sets SEI message, a non-nested buffering period SEI message, or a non-nested picture timing SEI message.

– When a nested buffering period SEI message, a nested picture timing SEI message, or a nested decoding unit information SEI message is contained in a scalable nesting SEI message 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, or another scalable nesting SEI message that contains a buffering period SEI message, a picture timing SEI message, or a decoding unit information SEI message.

– When payloadType is equal to 0 (buffering period), 1 (picture timing), or 130 (decoding unit information) for an SEI message, nested or non-nested, within the access unit, the SEI NAL unit containing the SEI message shall precede all NAL units of any picture unit that has nuh\_layer\_id greater than highestAppLayerId, where highestAppLayerId is the greatest value of nuh\_layer\_id of all the layers in all the operation points that the SEI message applies to.

– When payloadType is equal to 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 128, 131, 132, ~~or~~ 134, or XXX(i.e. one of the SEI messages that have payloadType not equal to any of 0, 1, 4, 5, 130, and 133) for an SEI message, nested or non-nested, within the access unit, the SEI NAL unit containing the SEI message shall precede all NAL units of any picture unit that has nuh\_layer\_id greater than highestAppLayerId, where highestAppLayerId is the greatest value of nuh\_layer\_id of all the layers that the SEI message applies to.

The following applies on the applicable operation points or layers of SEI messages:

– For a non-nested SEI message, when payloadType is equal to 0 (buffering period) or 130 (decoding unit information), the non-nested SEI message applies to the operation point that has OpTid equal to the greatest value of nuh\_temporal\_id\_plus1 among all VCL NAL units in the bitstream, and that has OpLayerIdList containing all values of nuh\_layer\_id in all VCL units in the bitstream.

– For a non-nested SEI message, when payloadType is equal to 1 (picture timing), the frame field information carried in the syntax elements pic\_struct, source\_scan\_type, and duplicate\_flag, when present, in the non-nested picture timing SEI message applies to the base layer only, while the picture timing information carried in other syntax elements, when present, in the non-nested picture timing SEI message applies to the operation point that has OpTid equal to the greatest value of nuh\_temporal\_id\_plus1 among all VCL NAL units in the bitstream, and that has OpLayerIdList containing all values of nuh\_layer\_id in all VCL units in the bitstream.

– For a non-nested SEI message, when payloadType is equal to 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 128, 131, 132, ~~or~~ 134, or XXX (i.e. one of the SEI messages that have payloadType not equal to any of 0, 1, 4, 5, 130, and 133), the non-nested SEI message applies to the layer for which the VCL NAL units have nuh\_layer\_id equal to the nuh\_layer\_id of the SEI NAL unit containing the SEI message. [Ed. (MH): Editorial suggestion: as this list is repeated multiple times and as any new SEI messages would have to be included in the list, it would be better to assign the list to a global variable and refer to the variable. The same applies for the other list 0, 1, 4, 5, 130 and other similar lists.]

– An active parameter sets SEI message, which cannot be nested, applies to all layers in the bitstream.

It is a requirement of bitstream conformance that the following restrictions apply on nesting of SEI messages:

– An SEI message that has payloadType equal to 129 (active parameter sets), 132 (decoded picture hash), and 133 (scalable nesting) shall not be nested in a scalable nesting SEI message.

– When a scalable nesting SEI message contains a buffering period SEI message, a picture timing SEI message, or a decoding unit information SEI message, the scalable nesting SEI message shall not contain any other SEI message with payloadType not equal to 0 (buffering period), 1 (picture timing), or 130 (decoding unit information).

– When a scalable nesting SEI message contains a buffering period SEI message, a picture timing SEI message, or a decoding unit information SEI message, the value of bitstream\_subset\_flag of the scalable nesting SEI message shall be equal to 1.

– When a scalable nesting SEI message contains an SEI message that has payloadType equal to 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 128, 131, 132, ~~or~~ 134, or XXX (i.e. one of the SEI messages that have payloadType not equal to any of 0, 1, 4, 5, 130, and 133), the value of bitstream\_subset\_flag of the scalable nesting SEI message shall be equal to 0.

It is a requirement of bitstream conformance that the following restrictions apply on the values of nuh\_layer\_id and TemporalId of SEI NAL units:

– When a non-nested SEI message has payloadType equal to 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 128, 131, 132, ~~or~~ 134, or XXX (i.e. one of the SEI messages that have payloadType not equal to any of 0, 1, 4, 5, 129, 130, and 133), the SEI NAL unit containing the non-nested SEI message shall have TemporalId equal to the TemporalId of the access unit containing the SEI NAL unit.

– When a non-nested SEI message has payloadType equal to 0, 1, 129, or 130, the SEI NAL unit containing the non-nested SEI message shall have nuh\_layer\_id equal to 0.

– When a non-nested SEI message has payloadType equal to 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 128, 131, 132, ~~or~~ 134, or XXX (i.e. one of the SEI messages that have payloadType not equal to any of 0, 1, 129, 130, and 133), the SEI NAL unit containing the non-nested SEI message shall have nuh\_layer\_id equal to the nuh\_layer\_id of VCL NAL unit associated with the SEI NAL unit.

NOTE 4 – For an SEI NAL unit containing a scalable nesting SEI message, the values of TemporalId and nuh\_layer\_id should be set equal to the lowest value of TemporalId and nuh\_layer\_id, respectively, of all the sub-layers or operation points the nested SEI messages apply to.

It is a requirement of bitstream conformance that the following restrictions apply on the presence of SEI messages between two VCL NAL units of a picture:

– When there is a prefix SEI message that has payloadType equal to 0, 1, 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 128, 129, ~~or~~ 131, or XXX (i.e. one of the prefix SEI messages that are not user data registered by Rec. ITU-T T.35 SEI message, user data unregistered SEI message, decoding unit information SEI message, scalable nesting SEI message, or region refresh information SEI message) and applies to a picture of a layer layerA present between two VCL NAL units of the picture in decoding order, there shall be a prefix SEI message that is of the same type and applies to the layer layerA present in the same access unit preceding the first VCL NAL unit of the picture.

– When there is a suffix SEI message that has payloadType equal to 3 (filler payload), 17 (progressive refinement segment end), 22 (post filter hint), or 132 (decoded picture hash) and applies to a picture of a layer layerA present between two VCL NAL units of the picture in decoding order, there shall be a suffix SEI message that is of the same type and applies to the layer layerA present in the same access unit succeeding the last VCL NAL unit of the picture.

It is a requirement of bitstream conformance that the following restrictions apply on repetition of SEI messages:

– For each of the following payloadType values, there shall be less than or equal to 8 identical sei\_payload( ) syntax structures within a picture unit: 0, 1, 2, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 128, 129, 131, 132, ~~and~~ 133, and XXX.

– There shall be less than or equal to 8 identical sei\_payload( ) syntax structures with payloadType equal to 130 within a decoding unit.

– The number of identical sei\_payload( ) syntax structures with payloadType equal to 134 in a picture unit shall be less than or equal to the number of slice segments in the picture unit.

In the following subclauses of this annex, when a particular SEI message applies to a set of one or more layers (instead of a set of operation points), i.e. when the payloadType value is not equal to one of 0 (buffering period), 1 (picture timing), and 130 (decoding unit information), the following applies:

– The semantics apply independently to each particular layer with nuh\_layer\_id equal to targetLayerId of the layers to which the particular SEI message applies.

– The current SEI message refers to the particular SEI message.

– The current access unit refers to the access unit containing the current SEI message.

– The current layer refers to the layer with nuh\_layer\_id equal to targetLayerId.

– The current picture or the current decoded picture refers to the picture with nuh\_layer\_id equal to targetLayerId (i.e. in the current layer) in the current access unit.

– The direct and indirect reference layers of the current layer are referred to as the reference layers of the current layer.

In the following subclauses of this annex, when a particular SEI message applies to a set of one or more operation points (instead of a set of one or more layers), i.e. when the payloadType value is equal to 0 (buffering period), 1 (picture timing), or 130 (decoding unit information), the following applies:

– The semantics apply independently to each particular operation point of the set of operation points to which the particular SEI message applies.

– The current SEI message refers to the particular SEI message.

– The current operation point refers to the particular operation point.

– The terms "access unit" and "CVS" apply to the bitstream BitstreamToDecode that is the sub-bitstream of the particular operation point.

– The terms "access unit" and "CVS" apply to the bitstream BitstreamToDecode that is the sub-bitstream of the particular operation point.

*Modify subclause D.3.2 as follows:*

A buffering period SEI message provides initial CPB removal delay and initial CPB removal delay offset information for initialization of the HRD at the position of the associated access unit in decoding order.

A buffering period SEI message may be included in a scalable nesting SEI message with nesting\_op\_flag equal to 1. [Ed. (YK): The sentence does not say anything normative/essential. Is it trying to say that when buffering period SEI message is nested, the containing scalable nesting SEI message shall have nesting\_op\_flag equal to 1? Either remove it, just put it to the semantics of the SN SEI message to, and it should be consistent for PT and DUI SEIs.]

If a buffering period SEI message is included in a scalable nesting SEI message, a set of skipped leading pictures skippedPictureList consists of the CL-RAS pictures and the RASL pictures associated with the IRAP pictures with nuh\_layer\_id equal to nuhLayerId for which LayerInitializedFlag[ nuhLayerId ] is equal to 0 at the start of decoding the IRAP picture and for which nuhLayerId is among the nestingLayerIdList[ i ] for any value of i in the range of 0 to nesting\_num\_ops\_minus1, inclusive. Otherwise (a buffering period SEI message is not nested in a scalable nesting SEI message), skippedPictureList consists of the RASL pictures associated with which the IRAP picture which the buffering period SEI message is associated.

The following applies for the buffering period SEI message syntax and semantics:

– The syntax elements initial\_cpb\_removal\_delay\_length\_minus1, au\_cpb\_removal\_delay\_length\_minus1, dpb\_output\_delay\_length\_minus1, and sub\_pic\_hrd\_params\_present\_flag, and the variables NalHrdBpPresentFlag and VclHrdBpPresentFlag are found in or derived from syntax elements found in the hrd\_parameters( ) syntax structure that is applicable to at least one of the operation points to which the buffering period SEI message applies.

– The variables CpbSize[ i ], BitRate[ i ] and CpbCnt are derived from syntax elements found in the sub\_layer\_hrd\_parameters( ) syntax structure that is applicable to at least one of the operation points to which the buffering period SEI message applies.

– Any two operation points that the buffering period SEI message applies to having different OpTid values tIdA and tIdB indicate that the values of cpb\_cnt\_minus1[ tIdA ] and cpb\_cnt\_minus1[ tIdB ] coded in the hrd\_parameters( ) syntax structure(s) applicable to the two operation points are identical.

– Any two operation points that the buffering period SEI message applies to having different OpLayerIdList values layerIdListA and layerIdListB indicate that the values of nal\_hrd\_parameters\_present\_flag and vcl\_hrd\_parameters\_present\_flag, respectively, for the two hrd\_parameters( ) syntax structures applicable to the two operation points are identical.

– The bitstream (or a part thereof) refers to the bitstream subset (or a part thereof) associated with any of the operation points to which the buffering period SEI message applies.

The presence of buffering period SEI messages for an operation point is specified as follows:

– If NalHrdBpPresentFlag is equal to 1 or VclHrdBpPresentFlag is equal to 1, the following applies for each access unit in the CVS:

* If the access unit is an IRAP access unit, a buffering period SEI message applicable to the operation point shall be associated with the access unit.
* Otherwise, if both of the following conditions apply, a buffering period SEI message applicable to the operation point may or may not be present for the access unit:
  + The access unit has TemporalId equal to 0.
  + The access unit has at least one picture that has discardable\_flag equal to 0 and is not a RASL, RADL or sub-layer non-reference picture.
* Otherwise, the access unit shall not be associated with a buffering period SEI message applicable to the operation point.

– Otherwise (NalHrdBpPresentFlag and VclHrdBpPresentFlag are both equal to 0), no access unit in the CVS shall be associated with a buffering period SEI message applicable to the operation point.

NOTE 1 – For some applications, frequent presence of buffering period SEI messages may be desirable (e.g. for random access at an IRAP picture or a non-IRAP picture or for bitstream splicing).

**bp\_seq\_parameter\_set\_id** indicates and shall be equal to the sps\_seq\_parameter\_set\_id for the SPS that is active for the coded picture associated with the buffering period SEI message. The value of bp\_seq\_parameter\_set\_id shall be equal to the value of pps\_seq\_parameter\_set\_id in the PPS referenced by the slice\_pic\_parameter\_set\_id of the slice segment headers of the coded picture associated with the buffering period SEI message. The value of bp\_seq\_parameter\_set\_id shall be in the range of 0 to 15, inclusive.

**irap\_cpb\_params\_present\_flag** equal to 1 specifies the presence of the initial\_alt\_cpb\_removal\_delay[ i ] and initial\_alt\_cpb\_removal\_offset[ i ] syntax elements. When not present, the value of irap\_cpb\_params\_present\_flag is inferred to be equal to 0. When the associated picture is neither a CRA picture nor a BLA picture, the value of irap\_cpb\_params\_present\_flag shall be equal to 0.

NOTE 2 – The values of sub\_pic\_hrd\_params\_present\_flag and irap\_cpb\_params\_present\_flag cannot be both equal to 1.

**cpb\_delay\_offset** specifies an offset to be used in the derivation of the nominal CPB removal times of pictures following, in decoding order, the CRA or BLA picture associated with the buffering period SEI message when no picture in skippedPictureList is present. The syntax element has a length in bits given by au\_cpb\_removal\_delay\_length\_minus1 + 1. When not present, the value of cpb\_delay\_offset is inferred to be equal to 0.

**dpb\_delay\_offset** specifies an offset to be used in the derivation of the DPB output times of the CRA or BLA picture associated with the buffering period SEI message when no picture in skippedPictureList is present. The syntax element has a length in bits given by dpb\_output\_delay\_length\_minus1 + 1. When not present, the value of dpb\_delay\_offset is inferred to be equal to 0.

When the current picture is not the first picture in the bitstream in decoding order, let prevNonDiscardablePic be the preceding picture in decoding order with TemporalId equal to 0 that is not a RASL, RADL or sub-layer non-reference picture.

**concatenation\_flag** indicates, when the current picture is not the first picture in the bitstream in decoding order, whether the nominal CPB removal time of the current picture is determined relative to the nominal CPB removal time of the preceding picture with a buffering period SEI message or relative to the nominal CPB removal time of the picture prevNonDiscardablePic.

**au\_cpb\_removal\_delay\_delta\_minus1** plus 1, when the current picture is not the first picture in the bitstream in decoding order, specifies a CPB removal delay increment value relative to the nominal CPB removal time of the picture prevNonDiscardablePic. This syntax element has a length in bits given by au\_cpb\_removal\_delay\_length\_minus1 + 1.

When the current picture contains a buffering period SEI message and concatenation\_flag is equal to 0 and the current picture is not the first picture in the bitstream in decoding order, it is a requirement of bitstream conformance that the following constraint applies:

* If the picture prevNonDiscardablePic is not associated with a buffering period SEI message, the au\_cpb\_removal\_delay\_minus1 of the current picture shall be equal to the au\_cpb\_removal\_delay\_minus1 of prevNonDiscardablePic plus au\_cpb\_removal\_delay\_delta\_minus1 + 1.
* Otherwise, au\_cpb\_removal\_delay\_minus1 shall be equal to au\_cpb\_removal\_delay\_delta\_minus1.

NOTE 3 – When the current picture contains a buffering period SEI message and concatenation\_flag is equal to 1, the au\_cpb\_removal\_delay\_minus1 for the current picture is not used. The above-specified constraint can, under some circumstances, make it possible to splice bitstreams (that use suitably-designed referencing structures) by simply changing the value of concatenation\_flag from 0 to 1 in the buffering period SEI message for an IRAP picture at the splicing point. When concatenation\_flag is equal to 0, the above-specified constraint enables the decoder to check whether the constraint is satisfied as a way to detect the loss of the picture prevNonDiscardablePic.

**nal\_initial\_cpb\_removal\_delay**[ i ] and **nal\_initial\_alt\_cpb\_removal\_delay**[ i ] specify the default and the alternative initial CPB removal delays, respectively, for the i-th CPB when the NAL HRD parameters are in use. The syntax elements have a length in bits given by initial\_cpb\_removal\_delay\_length\_minus1 + 1, and are in units of a 90 kHz clock. The values of the syntax elements shall not be equal to 0 and shall be less than or equal to 90000 \* ( CpbSize[ i ] ÷ BitRate[ i ] ), the time-equivalent of the CPB size in 90 kHz clock units.

**nal\_initial\_cpb\_removal\_offset**[ i ] and **nal\_initial\_alt\_cpb\_removal\_offset**[ i ] specify the default and the alternative initial CPB removal offsets, respectively, for the i-th CPB when the NAL HRD parameters are in use. The syntax elements have a length in bits given by initial\_cpb\_removal\_delay\_length\_minus1 + 1 and are in units of a 90 kHz clock.

Over the entire CVS, the sum of nal\_initial\_cpb\_removal\_delay[ i ] and nal\_initial\_cpb\_removal\_offset[ i ] shall be constant for each value of i, and the sum of nal\_initial\_alt\_cpb\_removal\_delay[ i ] and nal\_initial\_alt\_cpb\_removal\_offset[ i ] shall be constant for each value of i.

**vcl\_initial\_cpb\_removal\_delay**[ i ] and **vcl\_initial\_alt\_cpb\_removal\_delay**[ i ] specify the default and the alternative initial CPB removal delays, respectively, for the i-th CPB when the VCL HRD parameters are in use. The syntax elements have a length in bits given by initial\_cpb\_removal\_delay\_length\_minus1 + 1, and are in units of a 90 kHz clock. The values of the syntax elements shall not be equal to 0 and shall be less than or equal to 90000 \* ( CpbSize[ i ] ÷ BitRate[ i ] ), the time-equivalent of the CPB size in 90 kHz clock units.

**vcl\_initial\_cpb\_removal\_offset**[ i ] and **vcl\_initial\_alt\_cpb\_removal\_offset**[ i ] specify the default and the alternative initial CPB removal offsets, respectively, for the i-th CPB when the VCL HRD parameters are in use. The syntax elements have a length in bits given by initial\_cpb\_removal\_delay\_length\_minus1 + 1 and are in units of a 90 kHz clock.

Over the entire CVS, the sum of vcl\_initial\_cpb\_removal\_delay[ i ] and vcl\_initial\_cpb\_removal\_offset[ i ] shall be constant for each value of i, and the sum of vcl\_initial\_alt\_cpb\_removal\_delay[ i ] and vcl\_initial\_alt\_cpb\_removal\_offset[ i ] shall be constant for each value of i.

**use\_alt\_cpb\_params\_flag** is used to derive the value of UseAltCpbParamsFlag. When irap\_cpb\_params\_present\_flag is equal to 0, use\_alt\_cpb\_params\_flag shall not be equal to 1. When use\_alt\_cpb\_params\_flag is not present, it is inferred to be equal to 0.

NOTE 4 – Encoders are recommended either to set use\_alt\_cpb\_params\_flag equal to 0 or not to include irap\_cpb\_params\_present\_flag equal to 1 in non-nested buffering period SEI messages associated with a CRA or BLA picture for which at least one of its associated RASL pictures follows one or more of its associated RADL pictures in decoding order. Encoders are recommended either to set use\_alt\_cpb\_params\_flag equal to 0 or not to include irap\_cpb\_params\_present\_flag equal to 1 in a nested buffering period SEI message that is nested in a scalable nesting SEI message and associated with an IRAP picture with NoClrasOutputFlag equal to 1 for which at least one of RASL pictures associated with an IRAP picture with nuh\_layer\_id equal to nuhLayerId such that LayerInitializedFlag[ nuhLayerId ] equal to 0 (at the beginning of decoding the IRAP picture) follows one or more of its associated RADL pictures in decoding order or for which CL-RAS pictures are present.

*Add in subclause D.3.3 before the semantics of pic\_struct:*

If the picture timing SEI message is nested in a scalable nesting SEI message, the semantics of pic\_struct, source\_scan\_type, and duplicate\_flag are unspecified. [Ed. (YK): Should it be specified that decoders shall ignore the values of these fields in this case? (MH): "shall ignore" would not make any practical difference, as these syntax elements affect decoding.] Otherwise, the semantics of pic\_struct, source\_scan\_type, and duplicate\_flag are specified in the following paragraphs.

NOTE X – When an OLS contains multiple layers, the frame-field information SEI message can be used to indicate pic\_struct, source\_scan\_type, and duplicate\_flag for each of the multiple layers.

*Modify subclause D.3.4 as follows:*

**pan\_scan\_rect\_cancel\_flag** equal to 1 indicates that the SEI message cancels the persistence of any previous pan-scan rectangle SEI message in output order that applies to the current layer. pan\_scan\_rect\_cancel\_flag equal to 0 indicates that pan-scan rectangle information follows.

**pan\_scan\_rect\_persistence\_flag** specifies the persistence of the pan-scan rectangle SEI message for the current layer.

pan\_scan\_rect\_persistence\_flag equal to 0 specifies that the pan-scan rectangle information applies to the current decoded picture only.

Let picA be the current picture. pan\_scan\_rect\_persistence\_flag equal to 1 specifies that the pan-scan rectangle information persists for the current layer in output order until any of the following conditions are true:

– A new CVS begins.

– The bitstream ends.

– A picture picB in the current layer in an access unit containing a pan-scan rectangle SEI message with the same value of pan\_scan\_rect\_id and applicable to the current layer is output for which PicOrderCnt( picB ) is greater than PicOrderCnt( picA ), where PicOrderCnt( picB ) and PicOrderCnt( picA ) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

*Modify subclause D.3.8 as follows:*

The semantics below apply independently to each particular layer with nuh\_layer\_id equal to targetLayerId of the layers to which the recovery point SEI message applies. The current picture refers to the picture with nuh\_layer\_id equal to targetLayerId in the access unit containing the current SEI message.

NOTE 1 – If not nested, a recovery point SEI message applies to the layer for which the VCL NAL units have nuh\_layer\_id equal to the nuh\_layer\_id of the SEI NAL unit containing the SEI message. Otherwise, the layers to which a recovery point SEI message applies are specified by the scalable nesting SEI message that contains the SEI message.

The recovery point SEI message assists a decoder in determining when the decoding process will produce acceptable pictures with nuh\_layer\_id equal to targetLayerId for display after the decoder initiates random access or after the encoder indicates a broken link.

NOTE 2 – For a single-layer bitstream, the recovery point SEI message may be used to indicate that it is possible to perform random access from the current access unit even when the current picture is not an IRAP picture. For example, it is possible to code a picture as an intra picture but not an IRAP picture, enable random accessing from the picture, and add an associated recovery point SEI message to indicate random accessibility. This functionality can be used to improve the coding efficiency of a sequence of interlaced field pictures with random accessibility needed, because there is a restriction that pictures before a CRA picture in output order cannot use pictures that follow the CRA picture for inter-prediction reference, while such referencing is possible if the CRA picture was encoded as a non-IRAP picture.

When all decoded pictures in earlier access units in decoding order are removed from the bitstream, the recovery point picture (defined below) and all the subsequent pictures in output order in the current layer can be correctly or approximately correctly decoded, the current picture is referred to as a layer random-accessing picture. When the pictures that belong to any reference layer, that precede, in decoding order, the current picture and that may be used for reference by the current picture or subsequent pictures in decoding order are correctly decoded, and the recovery point picture and all the subsequent pictures in output order in the current layer can be correctly or approximately correctly decoded when no picture prior to the current picture in decoding order in the current layer are present in the bitstream, the current picture is referred to as a layer up-switching picture.

When the recovery point SEI message applies to the current layer and all the reference layers of the current layer, the current picture is indicated as a layer random-accessing picture. When the recovery point SEI message applies to the current layer only, the current picture is indicated as a layer up-switching picture.

Decoded pictures with nuh\_layer\_id equal to targetLayerId produced by random access at or before the access unit containing the recovery point SEI message need not be correct in content until the indicated recovery point, and the operation of the decoding process starting at access unit containing the recovery point SEI message may contain references to pictures unavailable in the decoded picture buffer.

In addition, by use of the broken\_link\_flag, the recovery point SEI message can indicate to the decoder the location of some pictures with nuh\_layer\_id equal to targetLayerId in the bitstream that can result in serious visual artefacts when displayed, even when the decoding process was begun at the location of a previous access unit containing an IRAP picture with nuh\_layer\_id equal to targetLayerId in decoding order.

NOTE 3 – The broken\_link\_flag can be used by encoders to indicate the location of a point after which the decoding process for the decoding of some pictures with nuh\_layer\_id equal to targetLayerId may cause references to pictures that, though available for use in the decoding process, are not the pictures that were used for reference when the bitstream was originally encoded (e.g. due to a splicing operation performed during the generation of the bitstream).

When random access is performed to start decoding from the access unit containing the recovery point SEI message, the decoder operates as if the associated access unit was the first access unit in the bitstream in decoding order, and the variable PrevPicOrderCnt[ nuh\_layer\_id ] used in derivation of PicOrderCntVal for each picture in the access unit is set equal to 0.

NOTE 4 – When HRD information is present in the bitstream, a buffering period SEI message should be associated with the access unit associated with the recovery point SEI message in order to establish initialization of the HRD buffer model after a random access.

Any SPS or PPS RBSP that is referred to by a picture of the access unit containing a recovery point SEI message or by any picture in a subsequent access unit in decoding order shall be available to the decoding process prior to its activation, regardless of whether or not the decoding process is started at the beginning of the bitstream or with the access unit, in decoding order, that contains the recovery point SEI message.

**recovery\_poc\_cnt** specifies the recovery point of decoded pictures with nuh\_layer\_id equal to targetLayerId in output order. If there is a picture picB with nuh\_layer\_id equal to targetLayerId that follows the current picture picA but precedes an access unit containing an IRAP picture with nuh\_layer\_id equal to targetLayerId in decoding order and PicOrderCnt( picB ) is equal to PicOrderCnt( picA ) plus the value of recovery\_poc\_cnt, where PicOrderCnt( picA ) and PicOrderCnt( picB ) are the PicOrderCntVal values of picA and picB, respectively, immediately after the invocation of the decoding process for picture order count for picB, the picture picB is referred to as the recovery point picture. Otherwise, the first picture picC with nuh\_layer\_id equal to targetLayerId in output order for which PicOrderCnt( picC ) is greater than PicOrderCnt( picA ) plus the value of recovery\_poc\_cnt is referred to as the recovery point picture, where PicOrderCnt( picA ) and PicOrderCnt( picC ) are the PicOrderCntVal values of picA and picC, respectively, immediately after the invocation of the decoding process for picture order count for picC. The recovery point picture shall not precede the current picture in decoding order. All decoded pictures with nuh\_layer\_id equal to targetLayerId in output order are indicated to be correct or approximately correct in content starting at the output order position of the recovery point picture. The value of recovery\_poc\_cnt shall be in the range of −MaxPicOrderCntLsb / 2 to MaxPicOrderCntLsb / 2 − 1, inclusive.

**exact\_match\_flag** indicates whether decoded pictures with nuh\_layer\_id equal to targetLayerId at and subsequent to the specified recovery point in output order derived by starting the decoding process at the access unit containing the recovery point SEI message will be an exact match to the pictures with nuh\_layer\_id equal to targetLayerId that would be produced by starting the decoding process at the location of a previous access unit where the picture of the layer with nuh\_layer\_id equal to targetLayerId and the pictures of all the direct and indirect reference layers are IRAP pictures, if any, in the bitstream. The value 0 indicates that the match may not be exact and the value 1 indicates that the match will be exact. When exact\_match\_flag is equal to 1, it is a requirement of bitstream conformance that the decoded pictures with nuh\_layer\_id equal to targetLayerId at and subsequent to the specified recovery point in output order derived by starting the decoding process at the access unit containing the recovery point SEI message shall be an exact match to the pictures with nuh\_layer\_id equal to targetLayerId that would be produced by starting the decoding process at the location of a previous access unit where the picture of the layer with nuh\_layer\_id equal to targetLayerId and the pictures of all the direct and indirect reference layers are IRAP pictures, if any, in the bitstream.

NOTE 5 – When performing random access, decoders should infer all references to unavailable pictures as references to pictures containing only intra coding blocks and having sample values given by Y equal to ( 1  <<  ( BitDepthY − 1 ) ), Cb and Cr both equal to ( 1  <<  ( BitDepthC − 1 ) ) (mid-level grey), regardless of the value of exact\_match\_flag.

When exact\_match\_flag is equal to 0, the quality of the approximation at the recovery point is chosen by the encoding process and is not specified in this Specification.

**broken\_link\_flag** indicates the presence or absence of a broken link in the layer with nuh\_layer\_id equal to targetLayerId at the location of the recovery point SEI message and is assigned further semantics as follows:

– If broken\_link\_flag is equal to 1, pictures with nuh\_layer\_id equal to targetLayerId produced by starting the decoding process at the location of a previous access unit where the picture of the layer with nuh\_layer\_id equal to targetLayerId and the pictures of all the direct and indirect reference layers are IRAP pictures may contain undesirable visual artefacts to the extent that decoded pictures with nuh\_layer\_id equal to targetLayerId at and subsequent to the access unit containing the recovery point SEI message in decoding order should not be displayed until the specified recovery point in output order.

– Otherwise (broken\_link\_flag is equal to 0), no indication is given regarding any potential presence of visual artefacts.

When the current picture is a BLA picture, the value of broken\_link\_flag shall be equal to 1.

Regardless of the value of the broken\_link\_flag, pictures with nuh\_layer\_id equal to targetLayerId subsequent to the specified recovery point in output order are specified to be correct or approximately correct in content.

*Modify subclause D.3.11 as follows:*

The progressive refinement segment start SEI message specifies the beginning of a set of consecutive coded pictures in the current layer in decoding order that consists of the current picture and a sequence of one or more subsequent pictures in the current layer that refine the quality of the current picture, rather than a representation of a continually moving scene.

Let picA be the current picture. The tagged set of consecutive coded pictures refinementPicSet in the current layer starts from the next picture in the current layer after the current picture in decoding order and continues, in decoding order, until one of the following conditions is true: [Ed. (MH): It is suggested to change the phrasing to indicate that the picture for which the condition is true is not included in refinementPicSet.]

– A new CVS begins.

– The bitstream ends.

– pic\_order\_cnt\_delta is greater than 0 and the PicOrderCntVal of the next slice, which belongs to the picture picB in the current layer, to be decoded, i.e. PicOrderCnt( picB ), is greater than PicOrderCnt( picA ) plus pic\_order\_cnt\_delta, where PicOrderCnt( picB ) and PicOrderCnt( picA ) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

– A progressive refinement segment end SEI message that has the same progressive\_refinement\_id as the one in this SEI message and also applies to the current layer is decoded.

The decoding order of pictures within refinementPicSet should be the same as their output order.

**progressive\_refinement\_id** specifies an identification number for the progressive refinement operation. progressive\_refinement\_id shall be in the range of 0 to 232 − 2, inclusive.

Values of progressive\_refinement\_id in the range of 0 to 255, inclusive, and in the range of 512 to 231 − 1, inclusive, may be used as determined by the application. Values of progressive\_refinement\_id in the range of 256 to 511, inclusive, and in the range of 231 to 232 − 2, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of progressive\_refinement\_id in the range of 256 to 511, inclusive, or in the range of 231 to 232 − 2, inclusive, shall ignore it.

**pic\_order\_cnt\_delta** specifies the last picture in refinementPicSet in decoding order as follows:

– If pic\_order\_cnt\_delta is equal to 0, the last picture in refinementPicSet in decoding order is the following picture:

– If the CVS contains one or more pictures in the current layer that follow the current picture in decoding order and are associated with a progressive refinement segment end SEI message that has the same progressive\_refinement\_id and also applies to the current layer, the last picture in refinementPicSet in decoding order is the first of these pictures in decoding order.

– Otherwise, the last picture in refinementPicSet in decoding order is the last picture in the current layer within the CVS in decoding order.

– Otherwise, the last picture in refinementPicSet in decoding order is the following picture:

– If the CVS contains one or more pictures in the current layer that follow the current picture in decoding order, are associated with a progressive refinement segment end SEI message with the same progressive\_refinement\_id and applicable to the current layer, and precede any picture picC in the current layer in the CVS for which PicOrderCnt( picC ) is greater than PicOrderCnt( picA ) plus pic\_order\_cnt\_delta, where PicOrderCnt( picC ) and PicOrderCnt( picA ) are the PicOrderCntVal values of picC and picA, respectively, immediately after the invocation of the decoding process for picture order count for picC, the last picture in refinementPicSet in decoding order is the first of these pictures in decoding order.

– Otherwise, if the CVS contains one or more pictures picD in the current layer that follow the current picture in decoding order for which PicOrderCnt( picD ) is greater than PicOrderCnt( picA) plus pic\_order\_cnt\_delta, where PicOrderCnt( picD ) and PicOrderCnt( picA ) are the PicOrderCntVal values of picD and picA, respectively, immediately after the invocation of the decoding process for picture order count for picD, the last picture in refinementPicSet in decoding order is the last picture in the current layer that precedes the first of these pictures in decoding order.

– Otherwise, the last picture in refinementPicSet in decoding order is the last picture in the current layer within the CVS in decoding order.

The value of pic\_order\_cnt\_delta shall be in the range of 0 to 256, inclusive.

*Modify subclause D.3.13 as follows:*

**film\_grain\_characteristics\_cancel\_flag** equal to 1 indicates that the SEI message cancels the persistence of any previous film grain characteristics SEI message in output order that applies to the current layer. film\_grain\_characteristics\_cancel\_flag equal to 0 indicates that film grain modelling information follows.

…

**film\_grain\_characteristics\_persistence\_flag** specifies the persistence of the film grain characteristics SEI message for the current layer.

film\_grain\_characteristics\_persistence\_flag equal to 0 specifies that the film grain characteristics SEI message applies to the current decoded picture only.

Let picA be the current picture. film\_grain\_characteristics\_persistence\_flag equal to 1 specifies that the film grain characteristics SEI message persists for the current layer in output order until any of the following conditions are true:

– A new CVS begins.

– The bitstream ends.

– A picture picB in the current layer in an access unit containing a film grain characteristics SEI message that is applicable to the current layer is output for which PicOrderCnt( picB ) is greater than PicOrderCnt( picA ), where PicOrderCnt( picB ) and PicOrderCnt( picA ) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

*Modify subclause D.3.15 as follows:*

**tone\_map\_cancel\_flag** equal to 1 indicates that the tone mapping information SEI message cancels the persistence of any previous tone mapping information SEI message in output order that applies to the current layer. tone\_map\_cancel\_flag equal to 0 indicates that tone mapping information follows.

**tone\_map\_persistence\_flag** specifies the persistence of the tone mapping information SEI message for the current layer.

tone\_map\_persistence\_flag equal to 0 specifies that the tone mapping information applies to the current decoded picture only.

Let picA be the current picture. tone\_map\_persistence\_flag equal to 1 specifies that the tone mapping information persists for the current layer in output order until any of the following conditions are true:

– A new CVS begins.

– A picture picB in the current layer in an access unit containing a tone mapping information SEI message with the same value of tone\_map\_id and applicable to the current layer is output for which PicOrderCnt( picB ) is greater than PicOrderCnt( picA ), where PicOrderCnt( picB ) and PicOrderCnt( picA ) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

*Modify subclause D.3.16 as follows:*

**frame\_packing\_arrangement\_cancel\_flag** equal to 1 indicates that the frame packing arrangement SEI message cancels the persistence of any previous frame packing arrangement SEI message in output order that applies to the current layer. frame\_packing\_arrangement\_cancel\_flag equal to 0 indicates that frame packing arrangement information follows.

**…**

**frame\_packing\_arrangement\_persistence\_flag** specifies the persistence of the frame packing arrangement SEI message for the current layer.

frame\_packing\_arrangement\_persistence\_flag equal to 0 specifies that the frame packing arrangement SEI message applies to the current decoded frame only.

Let picA be the current picture. frame\_packing\_arrangement\_persistence\_flag equal to 1 specifies that the frame packing arrangement SEI message persists for the current layer in output order until any of the following conditions are true:

– A new CVS begins.

– The bitstream ends.

– A frame picB in the current layer in an access unit containing a frame packing arrangement SEI message with the same value of frame\_packing\_arrangement\_id and applicable to the current layer is output for which PicOrderCnt( picB ) is greater than PicOrderCnt( picA ), where PicOrderCnt( picB ) and PicOrderCnt( picA ) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

*Modify subclause D.3.17 as follows:*

**display\_orientation\_persistence\_flag** specifies the persistence of the display orientation SEI message for the current layer.

display\_orientation\_persistence\_flag equal to 0 specifies that the display orientation SEI message applies to the current decoded picture only.

Let picA be the current picture. display\_orientation\_persistence\_flag equal to 1 specifies that the display orientation SEI message persists for the current layer in output order until one or more of the following conditions are true:

– A new CVS begins.

– The bitstream ends.

– A picture picB in the current layer in an access unit containing a display orientation SEI message that is applicable to the current layer is output for which PicOrderCnt( picB ) is greater than PicOrderCnt( picA ), where PicOrderCnt( picB ) and PicOrderCnt( picA ) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

*Modify subclause D.3.18 as follows:*

The structure of pictures information SEI message provides information for a list of entries, some of which correspond to a series of pictures in decoding order in the current layer in the CVS.

The first entry in the structure of pictures information SEI message corresponds to the current picture. When there is a picture in the current layer that has PicOrderCntVal equal to the variable entryPicOrderCnt[ i ] as specified below, the entry i corresponds to the picture in the CVS. The decoding order of the pictures in the current layer in the CVS that correspond to entries in the structure of pictures information SEI message corresponds to increasing values of i in the list of entries.

Any picture in the current layer in the CVS that has PicOrderCntVal equal to entryPicOrderCnt[ i ] for any i in the range of 0 to num\_entries\_in\_sop\_minus1, inclusive, shall correspond to an entry in the list of entries.

The structure of pictures information SEI message shall not be present in a CVS and applicable for a layer for which the active SPS has long\_term\_ref\_pics\_present\_flag equal to 1 or num\_short\_term\_ref\_pic\_sets equal to 0.

The structure of pictures information SEI message shall not be present in any access unit that has TemporalId greater than 0 or contains a RASL, RADL or sub-layer non-reference picture in the current layer. Any picture in the current layer in the CVS that corresponds to an entry other than the first entry described in the structure of pictures information SEI message shall not be an IRAP picture.

*Modify subclause D.3.19 as follows:*

This message provides a hash for each colour component of the current decoded picture.

NOTE 1 – The decoded picture hash SEI message is a suffix SEI message and cannot be contained in a scalable nesting SEI message.

*Modify subclause D.3.20 as follows:*

**num\_sps\_ids\_minus1** plus 1 indicates and shall be equal to the number of SPSs that are referred to by the VCL NAL units of the access unit associated with the active parameter sets SEI message. The value of num\_sps\_ids\_minus1 shall be in the range of 0 to 15, inclusive.

*Modify subclause D.3.23 as follows:*

The scalable nesting SEI message provides a mechanism to associate SEI messages with bitstream subsets corresponding to various operation points or with specific layers or sub-layers.

A scalable nesting SEI message contains one or more SEI messages.

**bitstream\_subset\_flag** equal to 0 specifies that the SEI messages contained in the scalable nesting SEI message apply to specific layers or sub-layers. bitstream\_subset\_flag equal to 1 specifies that the SEI messages contained in the scalable nesting SEI message apply to one or more sub-bitstreams resulting from a sub-bitstream extraction process as specified in clause 10 with inputs based on the syntax elements of the scalable nesting SEI message as specified below.

[Ed. (YK): A paragraph is removed here.]

Depending on the value of bitstream\_subset\_flag, the layers or sub-layers, or the operation points to which the SEI messages contained in the scalable nesting SEI message apply are specified by deriving the lists nestingLayedIdList[ i ] and the variables maxTemporalId[ i ] based on syntax element values as specified below.

**nesting\_op\_flag** equal to 0 specifies that the list nestingLayerIdList[ 0 ] is specified by all\_layers\_flag and, when present, nesting\_layer\_id[ i ] for all i values in the range of 0 to nesting\_num\_layers\_minus1, inclusive, and that the variable maxTemporalId[ 0 ] is specified by nesting\_no\_op\_max\_temporal\_id\_plus1. nesting\_op\_flag equal to 1 specifies that the list nestingLayerIdList[ i ] and the variable maxTemporalId[ i ] are specified by nesting\_num\_ops\_minus1, default\_op\_flag, nesting\_max\_temporal\_id\_plus1[ i ], when present, and nesting\_op\_idx[ i ], when present.

**default\_op\_flag** equal to 1 specifies that maxTemporalId[ 0 ] is equal to nuh\_temporal\_id\_plus1 of the current SEI NAL unit minus 1 and that nestingLayerIdList[ 0 ] contains all integer values in the range of 0 to nuh\_layer\_id of the current SEI NAL unit, inclusive, in increasing order of the values.

**nesting\_num\_ops\_minus1** plus 1 minus default\_op\_flag specifies the number of the following nesting\_op\_idx[ i ] syntax elements. The value of nesting\_num\_ops\_minus1 shall be in the range of 0 to 1023, inclusive.

If nesting\_op\_flag is equal to 0, the variable nestingNumOps is set equal to 1. Otherwise, the variable nestingNumOps is set equal to nesting\_num\_ops\_minus1 + 1.

**nesting\_max\_temporal\_id\_plus1**[ i ] is used to specify the variable maxTemporalId[ i ]. The value of nesting\_max\_temporal\_id\_plus1[ i ] shall be greater than or equal to nuh\_temporal\_id\_plus1 of the current SEI NAL unit. The variable maxTemporalId[ i ] is set equal to nesting\_max\_temporal\_id\_plus1[ i ] − 1.

**nesting\_op\_idx**[ i ] is used to specify the list nestingLayerIdList[ i ]. The value of nesting\_op\_idx[ i ] shall be in the range of 0 to 1023, inclusive.

The list nestingLayerIdList[ i ] is set equal to the OpLayerIdList of the nesting\_op\_idx[ i ]-th layer set specified by the active VPS.

**all\_layers\_flag** equal to 0 specifies that the list nestingLayerIdList[ 0 ] is specified by nesting\_layer\_id[ i ] for all i values in the range of 0 to nesting\_num\_layers\_minus1, inclusive. all\_layers\_flag equal to 1 specifies that the list nestingLayerIdList[ 0 ] consists of all values of nuh\_layer\_id present in the current access unit that are greater than or equal to nuh\_layer\_id of the current SEI NAL unit, in increasing order of the values.

NOTE – When nuh\_layer\_id of the SEI NAL unit containing the scalable nesting SEI message is greater than 0, bitstream\_subset\_flag and all\_layers\_flag cannot both be equal to 1, because in this case the applicable operation point of the nested SEI messages would not include the base layer and consequently the sub-bitstream corresponding to the applicable operation point would be a non-conforming bitstream.

When nesting\_op\_flag is equal to 0 and all\_layers\_flag is equal to 1, maxTemporalId[ 0 ] is set equal to 6.

**nesting\_no\_op\_max\_temporal\_id\_plus1** minus 1 specifies the value of maxTemporalId[ 0 ] when nesting\_op\_flag is equal to 0 and all\_layers\_flag is equal to 0. The value of nesting\_no\_op\_max\_temporal\_id\_plus1 shall not be equal to 0.

**nesting\_num\_layers\_minus1** plus 1 specifies the number of the following nesting\_layer\_id[ i ] syntax elements. The value of nesting\_num\_layers\_minus1 shall be in the range of 0 to 63, inclusive.

**nesting\_layer\_id**[ i ] specifies the i-th nuh\_layer\_id value included in the list nestingLayerIdList[ 0 ].

For any i and j in the range of 0 to nesting\_num\_layers\_minus1, inclusive, with i less than j, nesting\_layer\_id[ i ] shall be less than nesting\_layer\_id[ j ].

The list nestingLayerIdList[ 0 ] is set to consist of nesting\_layer\_id[ i ] for all i values in the range of 0 to nesting\_num\_layers\_minus1, inclusive, in increasing order of i values.

When bitstream\_subset\_flag is equal to 0, the following applies:

– The SEI messages contained in the scalable nesting SEI message apply to the sets of layers or sub-layers subLayerSet[ i ] for all i values in the range of 0 to nestingNumOps – 1, inclusive, where the VCL NAL units of the layers or sub-layers in each set subLayerSet[ i ] have nuh\_layer\_id values that are included in the list nestingLayerIdList[ i ] and TemporalId values that are in the range of the TemporalId of the current SEI NAL unit to maxTemporalId[ i ], inclusive.

– When a nested SEI message has payloadType equal to 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 128, 131, 132, or 134 (i.e. one of the SEI messages that have payloadType not equal to any of 0, 1, 4, 5, 130, and 133), the value of TemporalId of the SEI NAL unit containing the scalable nesting SEI message shall be equal to 0 and maxTemporalId[ i ] for all i shall be equal to 6. [Ed. (MH): This constraint seems too restrictive and unintuitive for those SEI messages which could apply to single pictures and hence have the TemporalId of the associated picture unit. (YK): Thanks for the fix – we had a weird typo in the original text. The intention for this constraint is just to simply say that all these SEI messages when nested apply to entire layers instead of sub-layers. Otherwise, quite some clarifications would be needed for at least some of the individual SEI messages to clarify how they apply to sub-layers. Think about how to clarify this for the recovery point SEI message and the region refresh information for example – a whole lot of text changes would be needed – I dared not even to try those though I did think about it.]

– When a nested SEI message has payloadType equal to 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 128, 131, 132, or 134 (i.e. one of the SEI messages that have payloadType not equal to any of 0, 1, 4, 5, 130, and 133) and the value of nestingNumOps is greater than 0, the nested SEI message applies to all layers for which each nuh\_layer\_id is included in at least one of the lists nestingLayerIdList[ i ] with i ranging from 0 to nestingNumOps – 1, inclusive.

When bitstream\_subset\_flag is equal to 1, the SEI messages contained in the scalable nesting SEI message apply to the operation points corresponding to the sub-bitstreams subBitstream[ i ] for all i values in the range of 0 to nestingNumOps − 1, inclusive, where each sub-bitstream subBitstream[ i ] is the output of the sub-bitstream extraction process of clause 10 with the bitstream, maxTemporalId[ i ], and nestingLayerIdList[ i ] as inputs.

**nesting\_zero\_bit** shall be equal to 0.

*Modify subclause D.3.24 as follows:*

The region refresh information SEI message indicates whether the slice segments that the current SEI message applies to belong to a refreshed region of the current picture.

The variable targetLayerIdList is derived as follows:

– If the region refresh information SEI message applies to the current layer and all the reference layers, targetLayerIdList contains the nuh\_layer\_id of the current layer and all the reference layers.

– Otherwise, targetLayerIdList contains the nuh\_layer\_id of the current layer.

The region refresh SEI message is associated with a recovery point SEI message that applies to targetLayerIdList.

A picture that belongs to a layer with nuh\_layer\_id greater than 0 or a picture that is not an IRAP picture and belongs to the layer with nuh\_layer\_id equal to 0, that is contained in an access unit containing a recovery point SEI message where the recovery point SEI message applies to that layer is referred to as a gradual decoding refresh (GDR) picture, and the access unit containing the picture is referred to as a GDR access unit. The access unit corresponding to the indicated recovery point picture is referred to as the recovery point access unit.

If there is a picture picB in the current layer that follows the GDR picture picA in the current layer in decoding order in the CVS and PicOrderCnt( picB ) is equal to PicOrderCnt( picA ) plus the value of recovery\_poc\_cnt in the recovery point SEI message, where PicOrderCnt( picA ) and PicOrderCnt( picB ) are the PicOrderCntVal values of picA and picB, respectively, immediately after the invocation of the decoding process for picture order count for picB, let the variable lastPicInSet be the recovery point picture. Otherwise, let lastPicInSet be the picture in targetLayerIdList that immediately precedes the recovery point picture in output order. The picture lastPicInSet shall not precede the GDR access unit in decoding order.

Let gdrPicSet be the set of pictures in targetLayerIdList starting from a GDR access unit to the access unit containing lastPicInSet, inclusive, in output order. When the decoding process for the current layer is started from a GDR access unit, the refreshed region in each picture of the gdrPicSet is indicated to be the region of the picture that is correct or approximately correct in content, and, when lastPicInSet is contained in the recovery point access unit, the refreshed region in lastPicInSet covers the entire picture.

The slice segments of the current picture to which a region refresh information SEI message applies consist of all slice segments within the picture that follow the SEI NAL unit containing the region refresh information SEI message and precede the next SEI NAL unit, in decoding order, containing a region refresh information SEI message (if any) that has the same targetLayerIdList as the current SEI message. These slice segments are referred to as the slice segments associated with the region refresh information SEI message.

Let gdrAuSet be the set of access units corresponding to gdrPicSet. A gdrAuSet and the corresponding gdrPicSet are referred to as being associated with the recovery point SEI message contained in the GDR access unit.

Region refresh information SEI messages shall not be present in an access unit unless the access unit is included in a gdrAuSet associated with a recovery point SEI message. When any picture that is included in a gdrPicSet is associated with one or more region refresh information SEI messages, all pictures in the gdrPicSet shall be associated with one or more region refresh information SEI messages.

**refreshed\_region\_flag** equal to 1 indicates that the slice segments associated with the current SEI message belong to the refreshed region in the current picture. refreshed\_region\_flag equal to 0 indicates that the slice segments associated with the current SEI message may not belong to the refreshed region in the current picture.

When one or more region refresh information SEI messages are associated with a picture belonging to gdrPicSet and the first slice segment of the picture in decoding order does not have an associated region refresh information SEI message, the value of refreshed\_region\_flag for the slice segments of the picture that precede the first region refresh information SEI message is inferred to be equal to 0.

When lastPicInSet is the recovery point picture, and any region refresh SEI message is associated with the recovery point access unit, the first slice segment of the picture in decoding order shall have an associated region refresh SEI message, and the value of refreshed\_region\_flag shall be equal to 1 in all region refresh SEI messages associated with the picture.

When one or more region refresh information SEI messages are associated with a picture, the refreshed region in the picture is specified as the set of CTUs in all slice segments of the picture that are associated with region refresh information SEI messages that have refreshed\_region\_flag equal to 1. Other slice segments belong to the non-refreshed region of the picture.

It is a requirement of bitstream conformance that when a dependent slice segment belongs to the refreshed region, the preceding slice segment in decoding order shall also belong to the refreshed region.

Let gdrRefreshedSliceSegmentSet be the set of all slice segments that belong to the refreshed regions in the gdrPicSet. The variable upSwitchingRefreshedSliceSegmentSet is derived as follows:

– If targetLayerIdList contains only one non-zero nuh\_layer\_id, upSwitchingRefreshedSliceSegmentSet is defined as the set inclusive of the following:

– all slice segments of all pictures of the reference layers that precede, in decoding order, the current picture and that may be used for reference by the current picture or subsequent pictures of the reference layers.

– all slice segments of all pictures of the reference layers that succeed, in decoding order, the current picture and that belong to gdrAuSet.

– Otherwise, upSwitchingRefreshedSliceSegmentSet is defined as an empty set.

When a gdrPicSet contains one or more pictures associated with region refresh information SEI messages, it is a requirement of bitstream conformance that the following constraints all apply:

– For each layer in targetLayerIdList, the refreshed region in the first picture, in decoding order, that belongs to the layer and that is included in gdrPicSet that contains any refreshed region shall contain only coding units that are coded in an intra coding mode or inter-layer prediction from slice segments belonging to the union of gdrRefreshedSliceSegmentSet and upSwitchingRefreshedSliceSegmentSet.

– For each picture included in the gdrPicSet, the syntax elements in gdrRefreshedSliceSegmentSet shall be constrained such that no samples or motion vector values outside of the union of gdrRefreshedSliceSegmentSet and upSwitchingRefreshedSliceSegmentSet are used for inter prediction or inter-layer prediction in the decoding process of any samples within gdrRefreshedSliceSegmentSet.

– For any picture that follows the picture lastPicInSet in output order, the syntax elements in the slice segments of the picture shall be constrained such that no samples or motion vector values outside of the union of gdrRefreshedSliceSegmentSet and upSwitchingRefreshedSliceSegmentSet are used for inter prediction or inter-layer prediction in the decoding process of the picture other than those of the other pictures that follow the picture lastPicInSet in output order.

*Add subclauses D.3.25, D.3.26, D.3.27, D3.28 and D.3.29 as follows and the original subclause index D.3.25 is modified as D.3.30:*

D.3.25 Temporal motion-constrained tile sets SEI message semantics

The temporal motion-constrained tile sets SEI message indicates that the 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.

Let a set of pictures associatedPicSet be the pictures with nuh\_layer\_id equal to targetLayerId from the access unit containing the SEI message, inclusive, up to but not including the first of any of the following in decoding order:

– The next access unit, in decoding order, that contains a temporal motion-constrained tile sets SEI message applicable to targetLayerId.

– The next IDR or BLA picture, in decoding order, with nuh\_layer\_id equal to targetLayerId.

– The next IRAP access unit, in decoding order, with NoClrasOutputFlag equal to 1.

The scope of the temporal motion-constrained tile sets SEI message is the set of pictures associatedPicSet.

When a temporal motion-constrained tile sets SEI message is present for any picture in associatedPicSet, a temporal motion-constrained tile sets SEI message applicable to targetLayerId shall be present for the first picture of associatedPicSet in decoding order and may also be present for other pictures of associatedPicSet.

The temporal motion-constrained tile sets SEI message applicable to targetLayerId shall not be present for any picture in associatedPicSet when tiles\_enabled\_flag is equal to 0 for any PPS that is active for any picture in associatedPicSet.

The temporal motion-constrained tile sets SEI message applicable to targetLayerId shall not be present for any picture in associatedPicSet unless every PPS that is active for any picture in associatedPicSet has the same values of the syntax elements num\_tile\_columns\_minus1, num\_tile\_rows\_minus1, uniform\_spacing\_flag, column\_width\_minus1[ i ], and row\_height\_minus1[ i ].

NOTE 1 – This constraint is similar to the constraint associated with tiles\_fixed\_structure\_flag equal to 1, and it may be desirable for tiles\_fixed\_structure\_flag to be equal to 1 when the temporal motion-constrained tile sets SEI message is present (although this is not required).

NOTE 2 – When loop filtering is applied across tile boundaries, inter prediction of any samples within a temporal motion-constrained tile set that refers to samples within four samples from a temporal motion-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 temporal motion-constrained tile sets SEI message applicable to targetLayerId is present for the pictures of AssociatedPicSet, they shall contain identical content.

The number of temporal motion-constrained tile sets SEI messages applicable to the same nuh\_layer\_id value in each access unit shall not exceed 5.

**mc\_all\_tiles\_exact\_sample\_value\_match\_flag** equal to 0 indicates that when the coding tree blocks that are outside any identified tile are not decoded and the boundaries of the identified tile are treated as picture boundaries for purposes of the decoding process, the value of each sample in the tile may not be exactly the same as the value of the same sample when all the coding tree blocks of the picture are decoded. mc\_all\_tiles\_exact\_sample\_value\_match\_flag equal to 1 indicates that when the coding tree blocks that do not belong to any identified tile are not decoded and the boundaries of the identified tile are treated as picture boundaries for purposes of the decoding process, the value of each sample in the 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 associatedPicSet are decoded. [Ed. (GJS): What is an "identified tile"?]

**each\_tile\_one\_tile\_set\_flag** equal to 1 indicates that each tile is a separate temporal motion-constrained tile set.

**limited\_tile\_set\_display\_flag** equal to 1 specifies that the display\_tile\_set\_flag[ i ] syntax element is present and indicates that the tiles not included within any tile set with display\_tile\_set\_flag[ i ] equal to 1 are not intended for display. limited\_tile\_set\_display\_flag equal to 0 specifies that the display\_tile\_set\_flag[ i ] syntax element is not present.

**num\_sets\_in\_message\_minus1** plus 1 specifies the number of temporal motion-constrained tile sets identified in the SEI message. The value of num\_sets\_in\_message\_minus1 shall be in the range of 0 to 255, inclusive.

**mcts\_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 associatedPicSet for a particular purpose). The value of pan\_scan\_rect\_id shall be in the range of 0 to 232 − 2, inclusive.

Values of mcts\_id[ i ] from 0 to 255 and from 512 to 231 − 1 may be used as determined by the application. Values of mcts\_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 mcts\_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.

**display\_tile\_set\_flag**[ i ] equal to 1 indicates that the i-th tile set is intended for display. display\_tile\_set\_flag[ i ] equal to 0 indicates that the i-th tile set is not intended for display. When not present, the value of display\_tile\_set\_flag[ i ] is inferred to be equal to 1.

**num\_tile\_rects\_in\_set\_minus1**[ i ] plus 1 specifies the number of rectangular regions of tiles in the i-th identified temporal motion-constrained tile set. The value of 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.

**top\_left\_tile\_index**[ i ][ j ] and **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 temporal motion-constrained tile set, respectively, in tile raster scan order.

**mc\_exact\_sample\_value\_match\_flag**[ i ] equal to 0 indicates that when the coding tree blocks that are outside of the i-th identified temporal motion-constrained tile set are not decoded and the boundaries of the temporal motion-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 not be exactly the same as the value of the same sample when all the coding tree blocks of the picture are decoded. mc\_exact\_sample\_value\_match\_flag[ i ] equal to 1 indicates that when the coding tree blocks that do not belong to the temporal motion-constrained tile set are not decoded and the boundaries of the temporal motion-constrained tile set are treated as picture boundaries for purposes of the decoding process, the value of each sample in the temporal motion-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 associatedPicSet are decoded.

NOTE 3 – It should be feasible to use mc\_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.

**mcts\_tier\_level\_idc\_present\_flag**[ i ] equal to 1 specifies that the mcts\_tier\_flag[ i ] and mcts\_level\_idc[ i ] syntax elements are present. mcts\_tier\_level\_idc\_present\_flag equal to 0 specifies that the mcts\_tier\_flag[ i ] and mcts\_level\_idc[ i ] syntax elements are not present. When mcts\_tier\_level\_idc\_present\_flag[ i ] is equal to 1, the i-th motion-constrained tile set shall form a rectangular shape.

**mcts\_tier\_flag**[ i ] specifies the tier context for the interpretation of mcts\_level\_idc[ i ] corresponding to the i-th motion constrained tile set. mcts\_tier\_flag[ i ] equal to 0 indicates conformance to the Main tier, and mcts\_tier\_flag[ i ] equal to 1 indicates conformance to the High tier, according to the tier constraints specified in Table A‑1. mcts\_tier\_flag[ i ] shall be equal to 0 for levels below level 4 (corresponding to the entries in Table A‑1 marked with "-") indicated by mcts\_level\_idc[ i ]. When not present, the value of mcts\_tier\_flag[ i ] is inferred to be equal to general\_tier\_flag.

**mcts\_level\_idc**[ i ] indicates a level to which the i-th tile set region, corresponding to the i-th motion constrained tile set, conforms. The value of mcts\_level\_idc[ i ] shall be less than or equal to the value of general\_level\_idc in the active SPS RBSP. When not present, the value of mcts\_level\_idc[ i ] is inferred to be equal to general\_level\_idc.

**max\_mcts\_tier\_level\_idc\_present\_flag** equal to 1 specifies that the max\_mcts\_tier\_flag and max\_mcts\_level\_idc syntax elements are present. max\_mcts\_level\_idc\_present\_flag equal to 0 specifies that the max\_mcts\_tier\_flag and max\_mcts\_level\_idc syntax elements are not present.

**max\_mcts\_tier\_flag** specifies the tier context for the interpretation of max\_mcts\_level\_idc to which all motion constrained tile sets conform. max\_mcts\_tier\_flag equal to 0 indicates conformance to the Main tier, and max\_mcts\_tier\_flag equal to 1 indicates conformance to the High tier, according to the tier constraints specified in Table A‑1. max\_mcts\_tier\_flag shall be equal to 0 for levels below level 4 (corresponding to the entries in Table A‑1 marked with "-") indicated by max\_mcts\_level\_idc. When not present, the value of max\_mcts\_tier\_flag[ i ] is inferred to be equal to general\_tier\_flag.

**max\_mcts\_level\_idc** indicates a level to which all motion constrained tile sets conform. The value of max\_mcts\_level\_idc shall be less than or equal to the value of general\_level\_idc in the active SPS RBSP. When not present, the value of max\_mcts\_level\_idc is inferred to be equal to general\_level\_idc.

The following describes the tier and level restrictions on the bitstreams of each motion-constrained tile set:

– If mcts\_level\_idc\_present\_flag and max\_mcts\_tier\_level\_present\_flag are both equal to 0, the mctsLevelIdc of all motion-constrained tile sets are inferred to be equal to general\_level\_idc and the mctsTierFlag of all motion-constrained tile sets are inferred to be equal to general\_tier\_flag and the specifications of Annex A apply to all motion-constrained tile sets.

– Otherwise (mcts\_tier\_level\_idc\_present\_flag or max\_mcts\_tier\_level\_present\_flag is equal to 1), the variables mctsLevelIdc[ i ], mctsTierFlag[ i ], mctsSizeInSamplesY[ i ], mctsWidthInSamplesY[ i ], mctsHeightInSamplesY[ i ], NumTileColumnsInMCTS[ i ], NumTileRowsInMCTS[ i ], mctsMaxSliceSegments[ i ], mctsMaxLumaSr[ i ], mctsMaxBR[ i ], and mctsMinCr[ i ] are derived as follows:

– If each\_tile\_one\_tile\_set\_flag is equal to 0, for each tile set with index i in the range of 0 to num\_sets\_in\_message\_minus1, inclusive, the following applies:

– The variable mctsLevelIdc[ i ] for the i-th motion-constrained tile set is set equal to mcts\_level\_idc[ i ].

– The variable mctsTierFlag[ i ] for the i-th motion-constrained tile set is set equal to mcts\_tier\_flag[ i ].

– mctsWidthInSamplesY[ i ] is set equal to the width of the i-th motion-constrained tile set in units of luma samples.

– mctsHeightInSamplesY[ i ] is set equal to the height of the i-th motion-constrained tile set in units of luma samples.

– The variable mctsSizeInSamplesY[ i ] for the i-th motion-constrained tile set is derived as follows:

mctsSizeInSamplesY[ i ] = mctsWidthInSamplesY[ i ] \* mctsHeightInSamplesY[ i ] (D‑1)

– NumTileColumnsInMCTS[ i ] is set equal to the number of tile columns in the i-th motion-constrained tile set.

– NumTileRowsInMCTS[ i ] is set equal to the number of tile rows in the i-th motion-constrained tile set.

– mctsMaxSliceSegments[ i ] is set equal to the number of slice segments in the i-th motion-constrained tile set.

– mctsMaxLumaSr[ i ] is set equal to the number of samples per second in the i-th motion-constrained tile set.

– mctsMaxBR[ i ] is set equal to MaxBR[ i ] for level indicated by mctsLevelIdc with tier indicated by mctsTierFlag for VCL HRD parameters of the i-th motion-constrained tile set.

– mctsMinCr[ i ] is set equal to the minimum compression ratio of the i-th motion-constrained tile set.

– Otherwise (each\_tile\_one\_tile\_set\_flag is equal to 1), for each tile with TileId i, with i in the range of 0 to ( num\_tile\_columns\_minus1 + 1 ) \* ( num\_tile\_rows\_minus1 + 1 ) − 1, inclusive, the following applies:

– mctsTierFlag[ i ] for the i-th motion constrained tile set is set equal to max\_mcts\_tier\_flag.

– The variable mctsLevelIdc[ i ] for the i-th motion constrained tile set is set equal to max\_mcts\_level\_idc.

– The variable mctsSizeInSamplesY[ i ] for i-th tile is derived as follows:

mctsSizeInSamplesY[ i ] = colWidth[ i % ( num\_tile\_rows\_minus1 + 1 ) ] \* rowHeight[ i / ( num\_tile\_rows\_minus1 + 1 ) ] \* CtbSizeY \* CtbSizeY (D‑2)

– NumTileColumnsInMCTS[ i ] is set equal to 1.

– NumTileRowsInMCTS[ i ] is set equal to 1.

– mctsMaxSliceSegments[ i ] is set equal to the number of slice segments in the i-th tile. [Ed. What if there is more than one tile in a slice segment?] [Ed (JC): Does each\_tile\_one\_tile\_set\_flag equal to 1 imply that all tiles can be parsed and reconstructed independently, otherwise, the level pparameters, such mctsMaxBR and mctsMinCr are meaningless.]

– mctsMaxLumaSr[ i ] is set equal to the number of samples per second in the i-th tile.

– mctsMaxBR[ i ] is set equal to MaxBR[ i ] for level indicated by mctsLevelIdc[ i ] with tier indicated by mctsTierFlag[ i ] for VCL HRD parameters of the i-th motion-constrained tile set.

– mctsMinCr[ i ] is set equal to the Min Compression Ratio MinCr of the the i-th tile.

mctsLevelIdc[ i ] and mctsTierFlag[ i ] indicate the level and tier to which the i-th motion-constrained tile set conforms, as specified in Annex A with the following modifications and additions:

– The variable PicSizeInSamplesY is replaced with the variable mctsSizeInSamplesY[ i ].

– The variable MaxTileCols is replaced with the variable NumTileColumnsInMCTS[ i ].

– The variable MaxTileRows is replaced with the variable NumTileRowsInMCTS[ i ].

– The variable MaxSliceSegmentsPerPicture is replaced with the variable mctsMaxSliceSegments[ i ].

– The variable MaxLumaSr is replaced with the variable mctsMaxLumaSr[ i ].

– The variable MaxBR is replaced with the variable mctsMaxBR[ i ].

– The variable MinCr is replaced with the variable mctsMinCr[ i ].

– The nominal removal time of motion-constrained tile set from the CPB shall be the same as removal time of corresponding access unit.

– The difference between consecutive output times of motion-constrained tile sets from the DPB shall be same as removal time of corresponding access unit.

[Ed. (GJS): Are the tiles in an MCTS required to be in separate NAL units from the tiles that are not in the MCTS?]

D.3.26 Chroma resampling filter hint SEI message semantics

[Ed. (JB): This SEI message needs editorial improvements.]

The chroma resampling filter hint SEI message identifies the coefficient values of one or more chroma resampling filters that are suggested for post-processing use with the output decoded pictures. When the chroma resampling filter at both the encoder and the decoder side are designed based on the coefficients signalled in the chroma resampling filter hint SEI message, degradation of the colour difference between the original signals and the resampled signals is expected to be minimized.

The chroma resampling filter hint SEI message shall not be present in a CVS that has chroma\_format\_idc equal to 0.

**ver\_chroma\_filter\_idc** specifies indicated chroma resampling filter coefficients in the vertical direction, as specified in Table D‑2, for the output decoded pictures and possibly for input pictures at an encoder that takes the output decoded pictures as an input. The value of ver\_chroma\_filter\_idc shall be in the range of 0 to 2, inclusive. Other values of ver\_chroma\_filter\_idc are reserved for future use by ITU-T | ISO/IEC. When ver\_chroma\_filter\_idc is equal to 0, the chroma resampling filter in the vertical direction is unspecified.

When chroma\_format\_idc is equal to 1, ver\_chroma\_filter\_idc shall be equal to 1 or 2.

**hor\_chroma\_filter\_idc** specifies indicated chroma resampling filter coefficients in the horizontal direction, as specified in Table D‑4, for the output decoded pictures and possibly for input pictures at an encoder that takes the output decoded pictures as an input. The value of hor\_chroma\_filter\_idc shall be in the range of 0 to 2, inclusive. Other values of hor\_chroma\_filter\_idc are reserved for future use by ITU-T | ISO/IEC. When hor\_chroma\_filter\_idc is equal to 0, the chroma resampling filter in the horizontal direction is unspecified.

When chroma\_format\_idc is equal to 3, hor\_chroma\_filter\_idc shall be equal to 1 or 2.

**ver\_filtering\_field\_processing\_flag** indicates the described chroma resampling filter operation in the vertical direction. If ver\_filtering\_field\_processing\_flag is equal to 1, the chroma resampling filter is indicated to be applied on a field basis (i.e. only chroma samples of the same parity are used for filtering). Otherwise, the chroma resampling filter is indicated to be applied on a frame basis. [Ed. (GJS): It's not necessarily a recommendation, so I removed the suggested addition of the word "recommended".]

The variable chromaSampleLocType is derived as follows:

* If chroma\_format\_idc is equal to 1 and ver\_filering\_field\_processing\_flag is equal to 1 and the picture being resampled is a bottom field [Ed: (GJS): How do we know whether a bottom field is resampled or not? What is this trying to say?], chromaSampleLocType is set equal to chroma\_sample\_loc\_type\_bottom\_field.
* Otherwise, if chroma\_format\_idc is equal to 1, chromaSampleLocType is set equal to chroma\_sample\_loc\_type\_top\_field.
* Otherwise (chroma\_format\_idc is not equal to 1), chromaSampleLocType is set equal to 0.

When chromaSampleLocType is greater than 1, ver\_chroma\_filter\_idc shall not be equal to 2.

When chromaSampleLocType is equal to 1, 3 or 5, hor\_chroma\_filter\_idc shall not be equal to 2.

**target\_format\_idc** indicates the output format of chroma resampling relative to that of the luma sampling as specified for values of chroma\_format\_idc in subclause 6.2. [Ed. (GJS): I think "resampling" is correct here.] The value of target\_format\_idc shall be in the range of 1 to 3, inclusive. The value of target\_format\_idc shall not be equal to the value of chroma\_format\_idc.

When not present, the value of target\_format\_idc is inferred as follows:

* If chroma\_format\_idc is equal to 1 and ver\_chroma\_filter\_idc is equal to 2 and hor\_chroma\_filter\_idc is not equal to 2, the value of target\_format\_idc is inferred to be equal to 2.
* Otherwise, if chroma\_format\_idc is equal to 1 and ver\_chroma\_filter\_idc is equal to 2 and hor\_chroma\_filter\_idc is equal to 2, the value of target\_format\_idc is inferred to be equal to 3.
* Otherwise, if chroma\_format\_idc is equal to 2 and ver\_chroma\_filter\_idc is equal to 2, the value of target\_format\_idc is inferred to be equal to 1.
* Otherwise, if chroma\_format\_idc is equal to 2 and hor\_chroma\_filter\_idc is equal to 2, the value of target\_format\_idc is inferred to be equal to 3.
* Otherwise, if chroma\_format\_idc is equal to 3 and ver\_chroma\_filter\_idc is equal to 2 and hor\_chroma\_filter\_idc is equal to 2, the value of target\_format\_idc is inferred to be equal to 1.
* Otherwise (if chroma\_format\_idc is equal to 3 and ver\_chroma\_filter\_idc is not equal to 2 and hor\_chroma\_filter\_idc is equal to 2), the value of target\_format\_idc is inferred to be equal to 2.

When chroma\_format\_idc is equal to 2 or 3 and target\_format\_idc is equal to 2 or 3, ver\_chroma\_filter\_idc shall be equal to 0.

When chroma\_format\_idc is equal to 1 or 2 and target\_format\_idc is equal to 1 or 2, hor\_chroma\_filter\_idc shall be equal to 0.**minimum\_degradation\_flag** equal to 1 indicates that the chroma resampling filter coefficients completely or approximately satisfy the perfect reconstruction condition, which means that the upsampled results are completely or approximately the same when downsampling and upsampling are repeated using the indicated filters[Ed. (JB): What is meant by “certain margin”? As written, the usefulness of the flag is questionable, as its meaning is unclear. ] [Ed. (KK): By "approximately satisfy", it is meant that the signalled filter coefficients (in integer number) are the rounded version of the coefficients which mathematically satisfy the perfect reconstruction condition.] [Ed. (JB): No, this is still unclear. What does “approximately satisfy” mean? Why is this even useful?] [Ed. (GJS): I agree that this is entirely unclear.] minimum\_degradation\_flag equal to 0 indicates that the chroma resampling filter coefficients may not satisfy the perfect reconstruction condition. [Ed. Is integer rounding error being neglected when determining the value of this flag? Also, isn't perfect reconstruction an inherent property of the filter tap values that the receiver of the SEI message can identify from the indicated filter tap values themselves without this flag? If so, why indicate it? Even if not, why indicate it? Does the flag cause the decoder to do something different than it otherwise would have, or is it just a way of "advertising" the filter?] [Ed. (KK): Let consider the following case: A(4:2:2)->DS->B(4:2:0)->US->C(4:2:2)->DS->D(4:2:0)->US->E(4:2:2). If the downsampling filter and its corresponding upsampling filter satisfy the perfect reconstuction condition, then B = = D and C = = E. If the filters approximately satisfy the condition, then B ~= D and C ~= E. It was shown that the difference between B and D , C and E converge to zero after several iteration of DS/US operation). If we only use ver(hor)\_chroma\_filter\_idc equal to 2, then this flag is not necessary because we know the corresponding filters (completely or approximately) satisfy the condition. But when we allow arbitary filter coefficients, we have to explicitly signal the property of the filter since it is not straightforward to check the signalled filter coefficients satisfy the condition. I think it is useful to introduce this flag as a "metadata".]

NOTE 1 – minumum\_degradation\_flag is used only to aid a decoder in determining whether the signalled filter coefficients completely or approximately satisfy the perfect reconstruction condition or not. [Ed. That doesn't seem to be useful to the decoder, especially since it is unclear what it means to approximately satisfy the condition.]

NOTE 2 – The chroma resampling filter coefficients signalled by ver\_chroma\_filter\_idc equal to 2 approximately satisfy perfect reconstruction condition. The chroma resampling filter coefficients signalled byor hor\_chroma\_filter\_idc equal to 2 completely satisfy perfect reconstruction condition.

[Ed. (all): It’s asserted that the information delivered by the syntax element minimum\_degradation\_flag can be identified from filter coefficient values at the decoder. The editor group and original proponent suggest the JCTVC discuss the necessity of this flag at Sapporo meeting and consider removal of this flag from SHVC draft]

**num\_vertical\_filters** specifies the number of filters indicated for chroma resampling in the vertical direction. The value of num\_vertical\_filters shall correspond to a value specified in Table D‑3.

**ver\_tap\_length\_minus1**[ i ] plus 1 specifies the length of the i-th filter in the vertical direction. The value of ver\_tap\_length\_minus1[ i ] shall be in the range of 0 to 31, inclusive. When ver\_tap\_length\_minus[ i ] is not present and ver\_chroma\_filter\_idc is equal to 2, the value of ver\_tap\_length\_minus1[ i ] is inferred as specified in Table D‑2.

**ver\_filter\_coeff**[ i ][ j ] specifies the value of the j-th coefficient of the i-th filter in the vertical direction. The value of ver\_filter\_coeff[ i ][ j ] shall be in the range of −231 to 231 − 1, inclusive. When ver\_filter\_coeff[ i ][ j ] is not present and ver\_chroma\_filter\_idc is equal to 2, the value of ver\_filter\_coeff[ i ][ j ] is inferred as specified in Table D‑2.

**num\_horizontal\_filters** specifies the number of filters indicated for chroma resampling in the horizontal direction. The value of num\_horizontal\_filters shall correspond to a value specified in Table D‑5. [Ed. (JB): Should be clarified. Unclear what this means]

**hor\_tap\_length\_minus1**[ i ] plus 1 specifies the length of the i-th filter in the horizontal direction. The value of hor\_tap\_length\_minus1[ i ] shall be in the range of 0 to 31, inclusive. When hor\_tap\_length\_minus1[ i ] is not present and hor\_chroma\_filter\_idc is equal to 2, the value of hor\_tap\_length\_minus1[ i ] is inferred as specified in Table D-4.

**hor\_filter\_coeff**[ i ][ j ] specifies the value of the j-th coefficient of the i-th filter in the horizontal direction. The value of hor\_filter\_coeff[ i ][ j ] shall be in the range of −231 to 231 − 1, inclusive. When hor\_filter\_coeff[ i ][ j ] is not present and hor\_chroma\_filter\_idc is equal to 2, the value of hor\_filter\_coeff[ i ][ j ] is inferred as specified in Table D-4.

Table D‑2 specifies the coefficients of the chroma resampling filter in the vertical direction. When ver\_chroma\_filter\_idc is not equal to 0, the usage of coefficients of the chroma resampling filter in the vertical direction is specified in Table D‑3.

Table D‑4 specifies the coefficients of the chroma resampling filter in the horizontal direction. When hor\_chroma\_filter\_idc is not equal to 0, the usage of coefficients of the chroma resampling filter in the horizontal direction is specified in Table D‑5.

Table D‑2 – Chroma filter coefficients in the vertical direction

|  |  |  |  |
| --- | --- | --- | --- |
| ver\_chroma\_filter\_idc | ver\_filtering\_field\_processing\_flag | Vertical chroma resampling filter coefficients | Informative Remarks |
| 0 | – | Unspecified | Chroma filter coefficients are unknown or determined by the application. |
| 1 | – | ver\_filter\_coeff[ i ][ j ]  i = 0..num\_vertical\_filters − 1  j = 0..ver\_tap\_length\_minus1[ i ] | Filter coefficients are explicitly listed in the chroma resampling filter hint SEI message |
| 2 | 0 | ver\_filter\_coeff[ i ][ j ] inferred as follows:  ver\_filter\_coeff[ 0 ][ ] = { −3, −19, 34, 500, 500, 34, −19, −3 }  ver\_filter\_coeff[ 1 ][ ] = { 19, 103, 1037, −135 }  ver\_tap\_length\_minus1[ 0 ] = 7  ver\_tap\_length\_minus1[ 1 ] = 3 | Corresponds to the filter defined in SMPTE RP 2050-1:2012  See also SMPTE EG2050-2:2012 for information on a possible implementation of the filter defined in SMPTE RP2050-1:2012 |
| 1 | ver\_filter\_coeff[ 0 ][ ] = { −8, −26, 115, 586, 409, −48, −4, 0 }  ver\_filter\_coeff[ 1 ][ ] = { 24, −41, 1169, −128 }  ver\_filter\_coeff[ 2 ][ ] = { −76, 783, 330, −13 }  ver\_tap\_length\_minus1[ 0 ] = 7  ver\_tap\_length\_minus1[ 1 ] = 3  ver\_tap\_length\_minus1[ 2 ] = 3 |
| 3..255 |  | Reserved | For future use by ITU-T | ISO/IEC |

Table D‑3 – Usage of chroma filter in the vertical direction

|  |  |  |  |
| --- | --- | --- | --- |
| chromaSampleLocType | ver\_filtering\_process\_flag | num\_vertical\_filters (when applicable) | Usage |
| 0, 1 | 0 | 2 | The variables fDv[ 0 ][ ] and lenDv[ 0 ] are used for downsampling.  fDv[ 0 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 0 ] are set equal to ver\_filter\_coeff[ 0 ][ j ].  lenDv[ 0 ] is set equal to ver\_tap\_length\_minus1[ 0 ] + 1.  The variables fUv[ k ][ ] and lenUv[ k ] with k = 0..1 are used for upsampling.  fUv[ 0 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 1 ] are set equal to ver\_filter\_coeff[ 1 ][ j ] and fUv[ 1 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 1 ] are set equal to ver\_filter\_coeff[ 1 ][ ver\_tap\_length\_minus1[ 1 ] − j ].  lenUv[ 0 ] and lenUv[ 1 ] are set equal to ver\_tap\_length\_minus1[ 1 ] + 1. |
| 1 | 3 | The variables fDv[ 0 ][ ] and lenDv[ 0 ] are used for downsampling a top field.  fDv[ 0 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 0 ] are set equal to ver\_filter\_coeff[ 0 ][ j ].  lenDv[ 0 ] is set equal to ver\_tap\_length\_minus1[ 0 ] + 1.  The variables fDv[ 1 ][ ] and lenDv[ 1 ] are used for downsampling a bottom field.  fDv[ 1 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 0 ] are set equal to ver\_filter\_coeff[ 0 ][ ver\_tap\_length\_minus1[ 0 ] − j ].  lenDv[ 0 ] is set equal to ver\_tap\_length\_minus1[ 0 ] + 1.  The variables fUv[ k ][ ] and lenUv[ k ] with k = 0..1 are used for upsampling a top field.  fUv[ 0 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 1 ] are set equal to ver\_filter\_coeff[ 1 ][ j ] and fUv[ 1 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 2 ] are set equal to ver\_filter\_coeff[ 2 ][ j ].  lenUv[ 0 ] is set equal to ver\_tap\_length\_minus1[ 1 ] + 1 and lenUv[ 1 ] is set equal to ver\_tap\_length\_minus1[ 2 ] + 1.  The variables fUv[ k ][ ] and lenUv[ k ] with k = 2..3 are used for upsampling a top field.  fUv[ 2 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 1 ] are set equal to ver\_filter\_coeff[ 1 ][ ver\_tap\_length\_minus1[ 1 ] - j ] and fUv[ 3 ][ j ] with i = 0..ver\_tap\_length\_minus1[ 2 ] are set equal to ver\_filter\_coeff[ 2 ][ ver\_tap\_length\_minus1[ 2 ] − j ].  lenUv[ 2 ] is set equal to ver\_tap\_length\_minus1[ 1 ] + 1 and lenUv[ 3 ] is set equal to ver\_tap\_length\_minus1[ 2 ] + 1. |
| 2, 3 | 0 | 3 | The variables fDv[ 0 ][ ] and lenDv[ 0 ] are used for downsampling.  fDv[ 0 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 0 ] are set equal to ver\_filter\_coeff[ 0 ][ j ].  lenDv[ 0 ] is set equal to ver\_tap\_length\_minus1[ 0 ] + 1  The variables fUv[ k ][ ] and lenUv[ k ] with k = 0..1 are used for upsampling  fUv[ 0 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 1 ] are set equal to ver\_filter\_coeff[ 1 ][ j ] and fUv[ 1 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 2 ] are set equal to ver\_filter\_coeff[ 2 ][ j ].  lenUv[ 0 ] is set equal to ver\_tap\_length\_minus1[ 1 ] + 1 and lenUv[ 1 ] is set equal to ver\_tap\_length\_minus1[ 2 ] + 1. |
| 1 | 5 | The variables fDv[ 0 ][ ] and lenDv[ 0 ] are used for downsampling a top field.  fDv[ 0 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 0 ] are set equal to ver\_filter\_coeff[ 0 ][ j ].  lenDv[ 0 ] is set equal to ver\_tap\_length\_minus1[ 0 ] + 1.  The variables fDv[ 1 ][ ] and lenDv[ 1 ] are used for downsampling a bottom field.  fDv[ 1 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 1 ] are set equal to ver\_filter\_coeff[ 1 ][ j ].  lenDv[ 1 ] is set equal to ver\_tap\_length\_minus1[ 1 ] + 1.  The variables fUv[ k ][ ] and lenUv[ k ] with k = 0..1 are used for upsampling a top field.  fUv[ 0 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 2 ] are set equal to ver\_filter\_coeff[ 2 ][ j ] and fUv[ 1 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 3 ] are set equal to ver\_filter\_coeff[ 3 ][ j ].  lenUv[ 0 ] is set equal to ver\_tap\_length\_minus1[ 2 ] + 1 and lenUv[ 1 ] is set equal to ver\_tap\_length\_minus1[ 3 ] + 1.  The variables fUv[ k ][ ] and lenUv[ k ] with k = 2..3 are used for upsampling a bottom field.  fUv[ 2 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 4 ] are set equal to ver\_filter\_coeff[ 4 ][ j ] and fUv[ 3 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 4 ] are set equal to ver\_filter\_coeff[ 4 ][ ver\_tap\_length\_minus1[ 4 ] − i ].  lenUv[ 3 ] is set equal to ver\_tap\_length\_minus1[ 4 ] + 1 and lenUv[ 4 ] is set equal to ver\_tap\_length\_minus1[ 4 ] + 1. |
| 4, 5 | 0 | 3 | The variables fDv[ 0 ][ ] and lenDv[ 0 ] are used for downsampling.  fDv[ 0 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 0 ] are set equal to ver\_filter\_coeff[ 0 ][ j ].  lenDv[ 0 ] is set equal to ver\_tap\_length\_minus1[ 0 ] + 1.  The variables fUv[ k ][ ] and lenUv[ k ] with k = 0..1 are used for upsampling.  fUv[ 0 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 1 ] are set equal to ver\_filter\_coeff[ 1 ][ j ] and fUv[ 1 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 2 ] are set equal to ver\_filter\_coeff[ 2 ][ j ].  lenUv[ 0 ] is set equal to ver\_tap\_length\_minus1[ 1 ] + 1 and lenUv[ 1 ] is set equal to ver\_tap\_length\_minus1[ 2 ] + 1. |
| 1 | 5 | The variables fDv[ 0 ][ ] and lenDv[ 0 ] are used for downsampling a top field.  fDv[ 0 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 0 ] are set equal to ver\_filter\_coeff[ 0 ][ j ].  lenDv[ 0 ] is set equal to ver\_tap\_length\_minus1[ 0 ] + 1.  The variables fDv[ 1 ][ ] and lenDv[ 1 ] are used for downsampling a bottom field.  fDv[ 1 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 1 ] are set equal to ver\_filter\_coeff[ 1 ][ j ].  lenDv[ 1 ] is set equal to ver\_tap\_length\_minus1[ 1 ] + 1.  The variables fUv[ k ][ ] and lenUv[ k ] with k = 0..1 are used for upsampling a top field.  fUv[ 0 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 2 ] are set equal to ver\_filter\_coeff[ 2 ][ j ] and fUv[ 1 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 2 ] are set equal to ver\_filter\_coeff[ 2 ][ ver\_tap\_length\_minus1[ 2 ] − j ].  lenUv[ 0 ] is set equal to ver\_tap\_length\_minus1[ 2 ] + 1 and lenUv[ 1 ] is set equal to ver\_tap\_length\_minus1[ 2 ] + 1.  The variables fUv[ k ][ ] and lenUv[ k ] with k = 2..3 are used for upsampling a bottom field.  fUv[ 2 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 3 ] are set equal to ver\_filter\_coeff[ 3 ][ j ] and fUv[ 3 ][ j ] with j = 0..ver\_tap\_length\_minus1[ 4 ] are set equal to ver\_filter\_coeff[ 4 ][ j ].  lenUv[ 3 ] is set equal to ver\_tap\_length\_minus1[ 4 ] + 1 and lenUv[ 4 ] is set equal to ver\_tap\_length\_minus1[ 4 ] + 1. |

Table D‑4 – Chroma filter coefficients in the horizontal direction

|  |  |  |
| --- | --- | --- |
| hor\_chroma\_filter\_idc | Horizontal chroma resampling filter coefficients | Informative Remarks |
| 0 | Unspecified | Chroma filter is unknown or is determined by the application |
| 1 | hor\_filter\_coeff[ i ][ j ]  i = 0 .. num\_horizontal\_filters − 1  j = 0 .. hor\_tap\_length\_minus1[ i ] | Filter coefficients are explicitly listed in the chroma resampling filter hint SEI message |
| 2 | hor\_filter\_coeff[ i ][ j ] inferred as follows:  hor\_filter\_coeff[ 0 ][ ] = { −1, 2, 6, 2, −1 }  hor\_filter\_coeff[ 1 ][ ] = { 1 }  hor\_filter\_coeff[ 2 ][ ] = { 1, 1 }  hor\_tap\_length\_minus1[ 0 ] = 4  hor\_tap\_length\_minus1[ 1 ] = 0  hor\_tap\_length\_minus1[ 2 ] = 1 | Corresponds to the 5/3 filter specified in ITU-T Rec. T.800 | ISO/IEC15444-1 |
| 3..255 | Reserved | For future use by ITU-T | ISO/IEC |

Table D‑5 – Usage of chroma filter in the horizontal direction

|  |  |  |
| --- | --- | --- |
| chromaSampleLocType | num\_horizontal\_filters (when applicable) | Usage |
| 0, 2, 4 | 3 | The variables fDh[ 0 ][ ] and lenDh are used for downsampling.  fDh[ 0 ][ j ] with j = 0..hor\_tap\_length\_minus1[ 0 ] are set equal to hor\_filter\_coeff[ 0 ][ j ].  lenDh is set equal to hor\_tap\_length\_minus1[ 0 ] + 1.  The variables fUh[ k ][ ] and lenUh[ k ] with k = 0..1 are for upsampling.  fUh[ 0 ][ j ] with j = 0..hor\_tap\_length\_minus1[ 1 ] are set equal to hor\_filter\_coeff[ 1 ][ j ] and fUh[ 1 ][ j ] with j = 0..hor\_tap\_length\_minus1[ 2 ] are set equal to hor\_filter\_coeff[ 2 ][ j ].  lenUh[ 0 ] is set equal to hor\_tap\_length\_minus1[ 1 ] + 1 and lenUh[ 1 ] is set equal to hor\_tap\_length\_minus1[ 2 ] + 1. |
| 1, 3, 5 | 2 | The variables fDh[ 0 ][ ] and lenDh are used for downsampling.  fDh[ 0 ][ j ] with j = 0..hor\_tap\_length\_minus1[ 0 ] are set equal to hor\_filter\_coeff[ 0 ][ j ].  lenDh is set equal to hor\_tap\_length\_minus1[ 0 ] + 1.  The variables fUh[ k ][ ] and lenUh[ k ] with k = 0..1 are used for upsampling.  fUh[ 0 ][ j ] with j = 0..hor\_tap\_length\_minus1[ 1 ] are set equal to hor\_filter\_coeff[ 1 ][ j ] and fUh[ 1 ][ j ] with j = 0..hor\_tap\_length\_minus1[ 1 ] are set equal to hor\_filter\_coeff[ 1 ][ hor\_tap\_length\_minus1[ 1 ] − j ].  lenUh[ 0 ] is set equal to hor\_tap\_length\_minus1[ 1 ] + 1 and lenUh[ 1 ] is set equal to hor\_tap\_length\_minus1[ 1 ] + 1. |

The chroma resampling filtering process is modelled as follows:

* The variables phaseOffsetUp and phaseOffsetDown are derived as follows:
  + If a picture for resampling is a bottom field, phaseOffsetUp is set equal to 2 and phaseOffsetDown is set equal to 1.
  + Otherwise (a picture for resampling is a top field or a progressive frame), phaseOffsetUp is set equal to 0 and phaseOffsetDown is set equal to 0.
* When chroma\_format\_idc is equal to 1 and target\_format\_idc is equal to either 2 or 3, the chroma upsampling filtering process in the vertical direction is applied. It is modelled as follows:

divUv[ 0 ] = 0  
divUv[ 1 ] = 0  
for( j = 0; j < lenUv[ 0 + phaseOffsetUp ]; j++ )  
 divUv[ 0 ] += fUv[ 0 + phaseOffsetUp ][ j ]  
for( j = 0; j < lenUv[ 1 + phaseOffsetUp ]; j++ )  
 divUv[ 1 ] += fUv[ 1 + phaseOffsetUp ][ j ]  
w0 = ( pic\_width\_in\_luma\_samples / SubWidthC ) – ( conf\_win\_right\_offset + conf\_win\_left\_offset )  
h0 = ( pic\_height\_in\_luma\_samples >> 1 ) – ( conf\_win\_top\_offset + conf\_win\_bottom\_offset )  
for( u = 0; u < w0; u++ )   
 for( v = 0; v < ( h0 << 1 ); v++ ) {  
 sum = 0   
 for( j = − ( lenUv[ v % 2 + phaseOffsetUp ] − 1 ) / 2;  
 j <= lenUv[ v % 2 + phaseOffsetUp ] / 2; j++ )  
 sum += p0[ u ][ Clip3( 0, ( v  >>  1 ) + j, h0 − 1 ) ]  
 \* fUv[ v % 2 + phaseOffsetUp ][ j + ( lenUv[ v % 2 + phaseOffsetUp ] − 1 ) / 2 ]  
 p1[ u ][ v ] = ( sum + ( divUv[ v % 2 ]  >>  1 ) ) / divUv[ v % 2 ]  
 } (D‑3)

where p0[ ][ ] is the array of chroma samples in a cropped output picture before vertical chroma upsampling, and p1[ ][ ] is the array of chroma samples in a cropped output picture after vertical chroma upsampling.

When ver\_filtering\_process\_flag is equal to 1 and field\_seq\_flag is equal to 0, the chroma upsampling filtering process in the vertical direction is applied to each field of the cropped output frame picture p0. Firstly p0 is deinterleaved to two fields p0\_top and p0\_bottom whose heights are equal to h0 >> 1. Then the chroma upsampling filtering process in the vertical direction is applied to p0\_top and p0\_bottom. Finally the outputs of the filtering process p1\_top and p1\_bottom are interleaved to form the cropped output frame picture after vertical chroma upsampling.

* When chroma\_format\_idc is equal to either 1 or 2 and target\_format\_idc is equal to 3, the chroma upsampling filtering process in the horizontal direction applied. It is is modelled as follows:

divUh[ 0 ] = 0  
divUh[ 1 ] = 0  
for( j = 0; j < lenUh[ 0 ]; j++ )  
 divUh[ 0 ] += fUh[ 0 ][ j ]  
for( j = 0; j < lenUh[ 1 ]; j++ )  
 divUh[ 1 ] += fUh[ 1 ][ j ]  
h0 = ( pic\_height\_in\_luma\_samples / SubHeightC ) – ( conf\_win\_top\_offset + conf\_win\_bottom\_offset )  
w0 = ( pic\_width\_in\_luma\_samples >> 1 ) – ( conf\_win\_right\_offset + conf\_win\_left\_offset )  
for( v = 0; v < h0; v++ )  
 for( u = 0; u < ( w0 << 1 ); u++ ) {  
 sum = 0 for( i = − ( lenUh[ u % 2 ] − 1 ) / 2; i  <=  lenUh[ u % 2 ] / 2; i++ )  
 sum += p0[ Clip3( 0, ( u  >>  1 ) + i, h0 − 1 ) ][ v ] \* fUh[ u % 2 ][ i + ( lenUh[ u % 2 ] − 1 ) / 2 ]  
 p1[ u ][ v ] = ( sum + ( divUh[ u % 2 ] >> 1 ) ) / divUh[ u % 2 ]  
 } (D‑4)

where p0[ ][ ] is the array of chroma samples in a cropped output picture before horizontal chroma upsampling, and p1[ ][ ] is the array of chroma samples in a cropped output picture after horizontal chroma upsampling.

* When chroma\_format\_idc is equal to either 3 or 2 and target\_format\_idc is equal to 1, the chroma downsampling filtering process in the vertical direction is applied. It is is modelled as follows:

divDv = 0  
for( j = 0; j < lenDv[ phaseOffsetDown ]; j++ )  
 divDv += fDv[ phaseOffsetDown ][ j ]  
w0 = ( pic\_width\_in\_luma\_samples / SubWidthC ) – ( conf\_win\_right\_offset + conf\_win\_left\_offset )  
h0 = pic\_height\_in\_luma\_samples – ( conf\_win\_top\_offset + conf\_win\_bottom\_offset )  
for( u = 0; u < w0; u++ )  
 for( v = 0; v < ( h0 >> 1 ); v++ ) {  
 sum = 0  
 for( j = − ( lenDv[ phaseOffsetDown ] − 1 ) / 2; j  <=  lenDv[ phaseOffsetDown ] / 2; j++ )  
 sum += p0[ u ][ Clip3( 0, ( v  <<  1 ) + j, h0 − 1 ) ]  
 \* fDv[ phaseOffsetDown ][ j + ( lenDv[ phaseOffsetDown ] − 1 ) / 2 ]  
 p1[ u ][ v ] = ( sum + ( divDv >> 1 ) ) / divDv  
 } (D‑5)

where p0[ ][ ] is the array of chroma samplesin a cropped output picture before vertical chroma downsampling, and p1[ ][ ] is the array chroma samples in a cropped output picture after vertical chroma downsampling.

When ver\_filtering\_process\_flag is equal to 1 and field\_seq\_flag is equal to 0, the chroma downsampling filtering process in the vertical direction is applied to each field of the cropped output frame picture p0. First, p0 is deinterleaved to two fields p0\_top and p0\_bottom whose heights are equal to h0 >> 1. Then the chroma downsampling filtering process in the vertical direction is applied to p0\_top and p0\_bottom. Finally the outputs of the filtering process p1\_top and p1\_bottom are interleaved to form the cropped output frame picture after vertical chroma downsample.

* When chroma\_format\_idc is equal to 3 and target\_format\_idc is equal to either 1 or 2, the chroma downsampling filtering process in the horizontal direction is applied. It is modelled as follows:

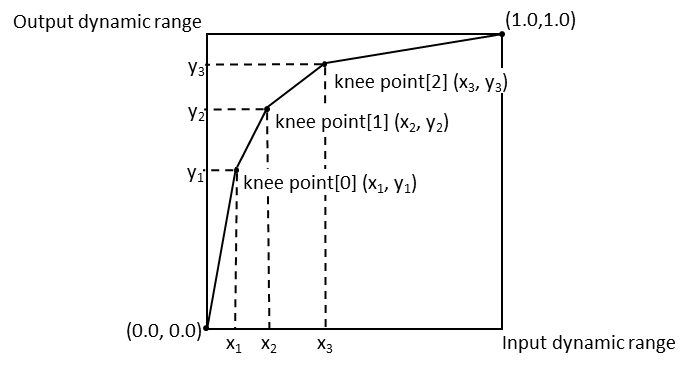
divUh = 0  
for( j = 0; j < lenDh; j++ )  
 divDh += fDh[ 0 ][ j ]   
h0 = ( pic\_height\_in\_luma\_samples / SubHeightC ) – ( conf\_win\_top\_offset + conf\_win\_bottom\_offset )  
w0 = pic\_width\_in\_luma\_samples – ( conf\_win\_right\_offset + conf\_win\_left\_offset )  
for( v = 0; v < h0; v++ )  
 for( u = 0; u < ( w0 >> 1 ); u++ ) {  
 sum = 0  
 for( i = − ( lenDh − 1 ) / 2; i  <=  lenDh / 2; i++ )  
 sum += p0[ Clip3( 0, ( u  <<  1 ) + i, w0 − 1 ) ][ v ] \* fDh[ 0 ][ i + ( lenDh − 1 ) / 2 ]  
 p1[ u ][ v ] = ( sum + ( divDh >> 1 ) ) / divDh  
 } (D‑6)

where p0[ ][ ] is the array of chroma samples in a cropped output picture before horizontal chroma downsampling, and p1[ ][ ] is the array of chroma samples in a cropped output picture after horizontal chroma downsampling.

D.3.27 Knee function information SEI message semantics

This SEI message provides information to enable decompression or compression of the colour samples of the output decoded pictures by using a knee function for customisation to particular display environments. The decompression or compression process maps the white level of sample values in the normalized linear RGB colour space. The compression or decompression should be applied to each RGB component produced by colour space conversion of the decoded image accordingly. [Ed. (JB): Consider defining what is meant here by “compression” or “decompression”]

A knee function is composed of several lines which starts with the point ( 0.0, 0.0 ), ends with the point ( 1.0, 1.0 ) and connects all knee points in ascending order of index i. The coordinate of the i-th knee point is specified by ( input\_knee\_point[ i ], output\_knee\_point[ i ] ). An example of knee function is shown in Figure D‑1.



**Figure D**‑**1** – **Knee Function**

**knee\_function\_id** contains an identifying number that may be used to identify the purpose of the knee functions. The value of knee\_function\_id shall be in the range of 0 to 232 − 2, inclusive.

Values of knee\_function\_id from 0 to 255 and from 512 to 231 − 1 may be used as determined by the application. Values of knee\_function\_id from 256 to 511, inclusive, and from 231 to 232 − 2, inclusive are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all knee function information SEI messages containing a value of knee\_function\_id in the range of 256 to 511, inclusive, or in the range of 231 to 232 − 2, inclusive, and bitstreams shall not contain such values.

NOTE 1 – The knee\_function\_id can be used to support knee function process that are suitable for different display scenarios. For example, different values of knee\_function\_id may correspond to different display bit depths.

**knee\_function\_cancel\_flag** equal to 1 indicates that the knee function information SEI message cancels the persistence of any previous knee function information SEI message in output order. knee\_function\_cancel\_flag equal to 0 indicates that knee function information follows.

**knee\_function\_persistence\_flag** specifies the persistence of the knee function information SEI message.

knee\_function\_persistence\_flag equal to 0 specifies that the knee function information applies to the current decoded picture only.

knee\_function\_persistence\_flag equal to 1 specifies that the knee function information persists in output order until any of the following conditions are true:

– A new CVS begins.

– A picture in an access unit containing a knee function information SEI message with the same value of knee\_function\_id is output having PicOrderCntVal greater than PicOrderCnt( CurrPic ).

**mapping\_flag** equals to 0 indicates that the knee function is a decompression function. mapping\_flag equals to 1 indicates that the knee function is a compression function.

**input\_d\_range** specifies the peak luminance level for the input picture of the knee function process relative to the nominal luminance level in units of 0.1%. When the value of input\_d\_range is 0, the peak luminance level of the input picture is unspecified.

**input\_disp\_luminance** specifies the expected display brightness of peak luminance level for the input picture of the knee function process. The value of input\_disp\_luminance is in units of candela per square metre. When the value of input\_disp\_luminance is 0, the expected display brightness of peak luminance level for the input picture is unspecified.

**output\_d\_range** specifies the peak luminance level for the output picture of the knee function process relative to the nominal luminance level in units of 0.1%. When the value of output\_d\_range is 0, the peak luminance level of the output picture is unspecified.

**output\_disp\_luminance** specifies the expected display brightness of peak luminance level for the output picture of the knee function process. The value of output\_disp\_luminance is in units of candela per square metre. When the value of output\_disp\_luminance is 0, the expected display brightness of peak luminance level for the output picture is unspecified.

**num\_knee\_points\_minus1** plus 1 specifies the number of knee points. num\_knee\_points\_minus1 shall be in the range of 0 to 998, inclusive.

**input\_knee\_point**[ i ] specifies the luminance level of the i-th knee point of the input picture. The luminance level of the knee point of the input picture is normalized to the range of 0 to 1.0 in units of 0.1%. The value of input\_knee\_point shall be in the range of 1 to 999, inclusive. The value of the i-th input\_knee\_point shall be greater than the value of the (i−1)-th input\_knee\_point.

**output\_knee\_point**[ i ] specifies the luminance level of the i-th knee of the output picture. The luminance level of the knee point of the output picture is normalized to the range of 0 to 1.0 in units of 0.1%. The value of output\_knee\_point shall be in the range of 0 to 1000, inclusive.

NOTE 2 – The input dynamic range (in percent) may be derived using input\_d\_range \* input\_knee\_point[ i ] ÷ 10000. The output dynamic range (in percent) may be derived using output\_d\_range \* output\_knee\_point[ i ] ÷ 10000.

D.3.28 Colour remapping information SEI message semantics

This SEI message provides information to enable remapping of the reconstructed colour samples of the output pictures. The colour remapping information may be applied to the reconstructed luma and chroma samples or the reconstructed RGB samples, and the remapped colour samples are expressed in either the luma and chroma domain or the RGB domain.

NOTE 1 – For example, an encoder may use the colour remapping information SEI message to map the original source video material from a first colour-grade to a second colour-grade. Colour remapping of the output pictures for the display is optional and does not affect the decoding process specified in this Specification.

**colour\_remap\_id** contains an identifying number that may be used to identify the purpose of the colour remapping information. The value of colour\_remap\_id shall be in the range of 0 to 232 − 2, inclusive.

Values of colour\_remap\_id from 0 to 255 and from 512 to 231 − 1 may be used as determined by the application. Values of colour\_remap\_id from 256 to 511, inclusive, and from 231 to 232 − 2, inclusive are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all colour remapping information SEI messages containing a value of colour\_remap\_id in the range of 256 to 511, inclusive, or in the range of 231 to 232 − 2, inclusive, and bitstreams shall not contain such values.

NOTE 2 – The colour\_remap\_id can be used to support different colour remapping processes that are suitable for different display scenarios. For example, different values of colour\_remap\_id may correspond to different remapped colour space supported by displays.

**colour\_remap\_cancel\_flag** equal to 1 indicates that the colour remapping information SEI message cancels the persistence of any previous colour remapping information SEI message in output order that applies to the current layer. colour\_remap\_cancel\_flag equal to 0 indicates that colour remapping information follows.

**colour\_remap\_persistence\_flag** specifies the persistence of the colour remapping information SEI message for the current layer. Let picA be the current picture. colour\_remap\_persistence\_flag equal to 0 specifies that the colour remapping information applies to picA only. colour\_remap\_persistence\_flag equal to 1 specifies that the colour remapping information persists for the current layer in output order until either of the following conditions is true:

– A new CVS begins.

– A picture picB in the current layer in an access unit containing a colour remapping information SEI message with the same value of colour\_remap\_id and applicable to the current layer is output for which PicOrderCnt( picB ) is greater than PicOrderCnt( picA ), where PicOrderCnt( picB ) and PicOrderCnt( picA ) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.[Ed. (PA): cf. notes on JCTVC-Q0183 and persistence notion]

**colour\_remap\_video\_signal\_type\_present\_flag** equal to 1 specifies that syntax elements colour\_remap\_video\_full\_range, colour\_remap\_primaries, colour\_remap\_transfer\_characteristics and colour\_remap\_matrix\_coeffs are present, colour\_remap\_video\_signal\_type\_present\_flag equal to 0 specifies that syntax elements colour\_remap\_video\_full\_range, colour\_remap\_primaries, colour\_remap\_transfer\_characteristics and colour\_remap\_matrix\_coeffs are not present.

**colour\_remap\_video\_full\_range\_flag** has the same semantics as specified in clause E.2.1 for the video\_full\_range\_flag syntax element, except that colour\_remap\_video\_full\_range\_flag specifies the colour space of the remapped reconstructed picture, rather than the colour space used for the CVS. When not present, the value of colour\_remap\_video\_full\_range\_flag is inferred to be equal to the value of video\_full\_range\_flag.

**colour\_remap\_primaries** has the same semantics as specified in clause E.2.1 for the colour\_primaries syntax element, except that colour\_remap\_primaries specifies the colour space of the remapped reconstructed picture, rather than the colour space used for the CVS. When not present, the value of colour\_remap\_primaries is inferred to be equal to the value of colour\_primaries.

**colour\_remap\_transfer\_characteristics** has the same semantics as specified in clause E.2.1 for the transfer\_characteristics syntax element, except that colour\_remap\_transfer\_characteristics specifies the colour space of the remapped reconstructed picture, rather than the colour space used for the CVS. When not present, the value of colour\_remap\_transfer\_characteristics is inferred to be equal to the value of transfer\_characteristics.

**colour\_remap\_matrix\_coeffs** has the same semantics as specified in clause E.2.1 for the matrix\_coeffs syntax element, except that colour\_remap\_matrix\_coeffs specifies the colour space of the remapped reconstructed picture, rather than the colour space used for the CVS. When not present, the value of colour\_remap\_matrix\_coeffs is inferred to be equal to the value of matrix\_coeffs.

**colour\_remap\_coded\_data\_bit\_depth** specifies the bit depth of the luma and chroma components or the RGB components of the associated pictures for purposes of interpretation of the colour remapping information SEI message. When colour remapping information SEI messages are present with the value of colour\_remap\_coded\_data\_bit\_depth not equal to the bit depth of the coded luma and chroma components or the coded RGB components, these refer to the hypothetical result of a transcoding operation performed to convert the coded video to a converted video with bit depth equal to colour\_remap\_coded\_data\_bit\_depth.

The value of colour\_remap\_coded\_data\_bit\_depth shall be in the range of 8 to 16, inclusive. Values of colour\_remap\_coded\_data\_bit\_depth from 0 to 7, inclusive, and from 17 to 255, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all colour remapping SEI messages that contain a colour\_remap\_coded\_data\_bit\_depth in the range of 0 to 7, inclusive, or in the range of 17 to 255, inclusive, and bitstreams shall not contain such values.

**colour\_remap\_target\_bit\_depth** specifies the bit depth of the output of the colour remapping function described by the colour remapping information SEI message.

The value of colour\_remap\_target\_bit\_depth shall be in the range of 8 to 16, inclusive. Values of colour\_remap\_target\_bit\_depth from 0 to 7, inclusive, and in the range of 17 to 255, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore all colour remapping SEI messages that contain a value of colour\_remap\_target\_bit\_depth from 0 to 7, inclusive, or in the range of 17 to 255, inclusive.

NOTE 3 –  colour\_remap\_target\_bit\_depth is representative of the intermediate computation accuracy.

**colour\_remap\_model\_id** specifies the colour remapping function used to remap the colour samples of the reconstructed output picture. A colour\_remap\_model\_id of 0 corresponds to a colour remapping model which applies to the three components of each sample a composition of a first piece-wise linear function applied to each component, a three by-three matrix applied to the three components, and a second piece-wise linear function applied to each component. Values greater than 0 are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore all colour remapping information SEI messages that contain a value of colour\_remap\_model\_id greater than 0.

**pre\_lut\_num\_pivots\_minus1**[ c ] plus 1specifies the number of pivot points in the piece-wise linear remapping function for the c-th component. When pre\_lut\_num\_pivots\_minus1[ c ] is equal to 0, the default end points of the input values are 0 and 2colour\_remap\_coded\_data\_bit\_depth – 1, and the corresponding default end points of the output values are 0 and 2colour\_remap\_target\_bit\_depth – 1, for the c-th component. In bitstreams conforming to this version of this Specification, the value of pre\_lut\_num\_pivots\_minus1[ c ] shall be in the range of 0 to 32, inclusive.

**pre\_lut\_coded\_pivot\_value**[ c ][ i ]specifies the value in the colour\_remap\_coded\_data\_bit\_depth corresponding to the i-th pivot point for the c-th component. The number of bits used to represent pre\_lut\_coded\_pivot\_value[ c ][ i ] is ( ( colour\_remap\_coded\_data\_bit\_depth + 7 ) >> 3 )  <<  3.

**pre\_lut\_target\_pivot\_value**[ c ][ i ] specifies the value in the reference colour\_remap\_target\_bit\_depth corresponding to the i-th pivot point for the c-th component. The number of bits used to represent pre\_lut\_target\_pivot\_value[ c ][ i ] is ( ( colour\_remap\_target\_bit\_depth + 7 ) >> 3 )  <<  3.

**colour\_remap\_matrix\_present\_flag** equal to 1 indicates that the syntax elements log2\_matrix\_denom and colour\_remap\_coeffs[ i ][ j ] are present. colour\_remap\_matrix\_present\_flag equal to 0 indicates that the syntax elements log2\_matrix\_denom and colour\_remap\_coeffs[ i ][ j ] are not present.

**log2\_matrix\_denom** specifies the base 2 logarithm of the denominator for all matrix coefficients. The value of log2\_matrix\_denom shall be in the range of 0 to 15, inclusive. When not present, the value of log2\_matrix\_denom is inferred to be equal to 0.

**colour\_remap\_coeffs**[ i ][ j ] specifies the value of the three-by-three matrix coefficients.

The variable matrixOutput[ c ] for c = 0, 1 and 2 is derived as follows:

roundingOffset = log2\_matrix\_denom = = 0 ? 0: 1<< ( log2\_matrix\_denom – 1 )  
matrixOutput[ c ] = Clip3( 0, ( 1 << colour\_remap\_target\_bit\_depth ) − 1,   
 ( colour\_remap\_coeffs[ c ][ 0 ] \* matrixInput[ 0 ] + colour\_remap\_coeffs[ c ][ 1 ] \* matrixInput[ 1 ]  
 + colour\_remap\_coeffs[ c ][ 2 ] \* matrixInput[ 2 ] + roundingOffset )  
 >> log2\_matrix\_denom ) (D‑7)

where matrixInput[ c ] represents the input sample value of the c-th colour component, where c equal to 0 refers to the luma or G component, c equal to 1 refers to the Cb or B component, and c equal to 2 refers to the Cr or R component. matrixOutput[ c ] represents the output sample value of the c-th colour component.

The value of colour\_remap\_coeffs[ i ][ j ] shall be in the range of −215 to 215 − 1, inclusive. When colour\_remap\_matrix\_present\_flag is equal to 0, the matrix consisting of corresponding values of colour\_remap\_coeffs[ i ][ j ] is inferred to be equal to the identity matrix I3.

**post\_lut\_num\_pivots\_minus1**[ c ] has the same semantics as pre\_lut\_num\_pivots\_minus1[ c ], with pre replaced by post, except that when post\_lut\_num\_pivots\_minus1[ c ] is equal to 0 and pre\_lut\_num\_pivots\_minus1[ c ] is not equal to 0 or post\_lut\_num\_pivots\_minus1[ c ] is equal to 0 and matrix\_present\_flag is equal to 1, the default end points of the input values are 0 and 2colour\_remap\_target\_bit\_depth − 1 for the c-th colour component. The value of post\_lut\_num\_pivots\_minus1[ c ] shall be in the range of 0 to 32, inclusive.

**post\_lut\_coded\_pivot\_value**[ c ][ i ] has the same semantics as pre\_lut\_coded\_pivot\_value[ c ][ i ], with pre replaced by post, except that when pre\_lut\_num\_pivots\_minus1[ c ] is not equal to 0 or matrix\_present\_flag is equal to 1, the number of bits used to represent post\_lut\_coded\_pivot\_value[ c ][ i ] is ( ( colour\_remap\_target\_bit\_depth + 7 ) >> 3 )  <<  3.

**post\_lut\_target\_pivot\_value**[ c ][ i ] has the same semantics as pre\_lut\_target\_pivot\_value[ c ][ i ], with pre replaced by post.

D.3.29 Deinterlaced picture information SEI message semantics

The deinterlaced picture information SEI message indicates that the current picture is a frame which had been interpolated via a deinterlacing process prior to encoding, and indicates the field parity of the associated source field picture prior to the deinterlacing process. When a progressive-to-interlace conversion process is applied to the decoded picture prior to display, it is recommended that the indicated field parity should be used.

When the deinterlaced picture information SEI message is present, field\_seq\_flag of the active SPS for the current picture shall be equal to 0.

**deinterlaced\_picture\_source\_parity\_flag** equal to 0 indicates that the current picture was deinterlaced using a top field picture as source. deinterlaced\_picture\_source\_parity\_flag equal to 1 indicates that the current picture was deinterlaced using a bottom field picture as source.

*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 – 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? ]

E.3.2 HRD parameters semantics

The specifications in clause E.3.2 apply with the following modifications and additions.

**initial\_cpb\_removal\_delay\_length\_minus1** plus 1 specifies the length, in bits, of the nal\_initial\_cpb\_removal\_delay[ i ], nal\_initial\_cpb\_removal\_offset[ i ], vcl\_initial\_cpb\_removal\_delay[ i ], and vcl\_initial\_cpb\_removal\_offset[ i ] syntax elements of the buffering period SEI message. Additionally, initial\_cpb\_removal\_delay\_length\_minus1 plus 1 within the j-th hrd\_parameters( ) syntax structure in the VPS specifies the length, in bits, of the nal\_initial\_arrival\_delay[ i ] and vcl\_initial\_arrival\_delay[ i ] syntax elements of the bitstream partition initial arrival time SEI message that is contained in a bitstream partition nesting SEI message within a scalable nesting SEI message with nesting\_op\_idx[ 0 ] equal to hrd\_layer\_set\_idx[ j ]. When the initial\_cpb\_removal\_delay\_length\_minus1 syntax element is not present, it is inferred to be equal to 23.

1. Annex F  
     
   Common specifications for multi-layer 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 with the following additions to subclause 2.4:

– ISO/IEC 10646:2003, *Information technology − Universal Multiple-Octet Coded Character Set (UCS)*.

* 1. Definitions

[Ed. (JO): Could be better to add this directly in clause 3. This can still be done with the new edition.]

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. additional layer set: a set of *layers* of a *bitstream* with a set of *layers* of one or more *non-base layer subtrees*.
2. alternative output layer: A *layer* that is a *direct reference layer* or an *indirect reference layer* of an *output layer* and which may include a *picture* that may be output when no picture of the *output layer* is present in the *access unit* containing the *picture*.
3. associated IRAP picture: The previous *IRAP picture* in *decoding order* within the same *layer* (if present).
4. auxiliary picture: A *picture* that has no normative effect on the *decoding process* of *primary pictures*, and with a nuh\_layer\_id value such that AuxId[ nuh\_layer\_id ] is greater than 0*.*
5. auxiliary picture layer: a *layer* with a nuh\_layer\_id value such that AuxId[ nuh\_layer\_id ] is greater than 0*.*
6. base layer: A *layer* in which all *VCL NAL units* have nuh\_layer\_id equal to 0.
7. 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.]
8. 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*.
9. collocated sample: A sample TBD. [Ed. (GT) Maybe it is easier to define a collocated position and require collocated samples to have it? ]
10. cross-layer random access skip (CL-RAS) picture: 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.
11. direct predicted layer: A *layer* for which another *layer* is a *direct reference layer*.
12. direct reference layer: A *layer* that may be used for inter-layer prediction of another *layer*.
13. independent layer: a *layer* that does not have *direct reference layers*.
14. indirect predicted layer: A *layer* for which another *layer* is an *indirect reference layer*.
15. indirect reference layer: A *layer* that is not a *direct reference layer* of a second *layer* but is a *direct reference layer* of a third *layer* that is a *direct reference layer* or indirect reference *layer* of a *direct reference layer* of the second *layer*.
16. 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.
17. 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.*
18. 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*.
19. layer subtree: a subset of the *layers* of a *layer tree* including all the *direct and indirect reference layers* of the *layers* within the subset.
20. layer tree: A set of *layers* such that each *layer* in the set of *layers* is a *direct or indirected predicted layer* or a *direct or indirect reference layer* of at least one other *layer* in the set of *layers* and no *layer* outside the set of *layers* is a *direct or indirected predicted layer* or a *direct or indirect reference layer* of any *layer* in the set of *layers*.
21. leading picture: A *picture* that is in the same *layer* as the *associated IRAP picture* and precedes the *associated IRAP picture* in *output order*.
22. non-base layer: A *layer* in which all *VCL NAL units* have the same nuh\_layer\_id value greater than 0.
23. non-base layer subtree: a *layer subtree* that does not include the *base layer*.
24. picture order count (POC): 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*.
25. picture order count (POC) resetting period: A sequence of access units in decoding order, starting with an access unit with poc\_reset\_idc equal to 1 or 2 and a particular value of poc\_reset\_period\_id and including all access units that either have the same value of poc\_reset\_period\_id or have poc\_reset\_idc equal to 0.
26. picture order count (POC) resetting picture: A picture that is the first picture, in decoding order, of a layer of a POC resetting period.
27. primary picture: a *picture* with nuh\_layer\_id value such that AuxId[ nuh\_layer\_id ] is equal to 0*.*
28. primary picture layer: a *layer* with a nuh\_layer\_id value such that AuxId[ nuh\_layer\_id ] is equal to 0*.*
29. reference layer picture: A *picture* in a *direct* *reference layer* which is used for inter-layer prediction of the current *picture* and is in the same access unit as the *current picture*.
30. reference picture list: A list of reference pictures that is used for inter prediction or inter-layer prediction of a P or B slice.
31. trailing picture: A *picture* that is in the same *layer* as the *associated IRAP picture* and follows the *associated IRAP picture* in *output order*.
32. 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.]
33. view: A sequence of pictures associated with the same value of ViewOrderIdx.

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

1. spatial scalability: A type of scalability, represented by a scalability dimension, that indicates that the coded pictures in different layers of a bitstream have different values of luma sample width and/or luma sample height.
2. quality scalability: A type of scalability, represented by a scalability dimension, that indicates that the coded pictures in all layers of a bitstream have the same values of luma sample width and luma sample height, but may have different fidelity level (e.g., in terms of SNR) or different values of luma or chroma sample bit depths.
   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, with the following additions:

– st(v): null-terminated string encoded in UTF-8 characters. The parsing process is specified as follows: st(v) reads and returns the next set of bytes from the bitstream until, exclusive, the next byte that is equal to 0x00, and advances the bitstream pointer by ( stringLength + 1 ) \* 8 bit positions, where stringLength is equal to the number of bytes returned..

more\_data\_in\_slice\_segment\_header\_extension( ) is specified as follows:

– If ( the current position in the slice\_segment\_header( ) syntax structure ) − ( the position immediately following slice\_segment\_header\_extension\_length ) is less than ( slice\_segment\_header\_extension\_length \* 8 ), the return value of more\_data\_in\_slice\_segment\_header\_extension( ) is equal to TRUE.

– Otherwise, the return value of more\_data\_in\_slice\_segment\_header\_extension( ) is equal to FALSE.

* + 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\_base\_layer\_internal\_flag** | u(1) |
| **vps\_reserved\_one\_bit** | u(1) |
| **vps\_max\_layers\_minus1** | u(6) |
| **vps\_max\_sub\_layers\_minus1** | u(3) |
| **vps\_temporal\_id\_nesting\_flag** | u(1) |
| **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 |
| **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\_num\_profile\_tier\_level\_minus1** | ue(v) |
| for( ptlIdx = 1; ptlIdx <= vps\_num\_profile\_tier\_level\_minus1; ptlIdx ++ ) { |  |
| **vps\_profile\_present\_flag**[ ptlIdx ] | u(1) |
| profile\_tier\_level( vps\_profile\_present\_flag[ ptlIdx ], vps\_max\_sub\_layers\_minus1 ) |  |
| } |  |
| if( NumIndependentLayers > 1 ) |  |
| **num\_add\_layer\_sets** | ue(v) |
| for( i = 0; i < num\_add\_layer\_sets; i++ ) |  |
| for( j = 1; j < NumIndependentLayers; j++ ) |  |
| **highest\_layer\_idx\_plus1**[ i ][ j ] | u(v) |
| if( NumLayerSets > 1 ) { |  |
| **num\_add\_olss** | ue(v) |
| **default\_output\_layer\_idc** | u(2) |
| } |  |
| NumOutputLayerSets = num\_add\_olss + NumLayerSets |  |
| for( i = 1; i < NumOutputLayerSets; i++ ) { |  |
| if( i >= NumLayerSets ) |  |
| **layer\_set\_idx\_for\_ols\_minus1**[ i ] | u(v) |
| if( i > vps\_num\_layer\_sets\_minus1 | | defaultOutputLayerIdc = = 2 ) |  |
| for( j = 0; j < NumLayersInIdList[ OlsIdxToLsIdx[ i ] ]; j++) |  |
| **output\_layer\_flag**[ i ][ j ] | u(1) |
| **profile\_level\_tier\_idx**[ i ] | u(v) |
| if( NumOutputLayersInOutputLayerSet[ i ] = = 1  && NumDirectRefLayers[ OlsHighestOutputLayerId[ i ] ] > 0 ) |  |
| **alt\_output\_layer\_flag**[ i ] | u(1) |
| } |  |
| **vps\_num\_rep\_formats\_minus1** | ue(v) |
| for( i = 0; i <= vps\_num\_rep\_formats\_minus1; i++ ) |  |
| rep\_format( ) |  |
| if( vps\_num\_rep\_formats\_minus1 > 0 ) |  |
| **rep\_format\_idx\_present\_flag** | u(1) |
| if( rep\_format\_idx\_present\_flag ) |  |
| for( i = vps\_base\_layer\_internal\_flag ? 1 : 0; i <= MaxLayersMinus1; i++ ) |  |
| **vps\_rep\_format\_idx**[ i ] | u(v) |
| **max\_one\_active\_ref\_layer\_flag** | u(1) |
| **vps\_poc\_lsb\_aligned\_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) |
| **default\_direct\_dependency\_flag** | u(1) |
| if( default\_direct\_dependency\_flag ) |  |
| **default\_direct\_dependency\_type** | u(v) |
| else { |  |
| for( i = vps\_base\_layer\_internal\_flag ? 1 : 2; i <= MaxLayersMinus1; i++ ) |  |
| for( j = vps\_base\_layer\_internal\_flag ? 0 : 1; j < i; j++ ) |  |
| if( direct\_dependency\_flag[ i ][ j ] ) |  |
| **direct\_dependency\_type**[ i ][ j ] | u(v) |
| } |  |
| **vps\_non\_vui\_extension\_length** | ue(v) |
| for( i = 1; i <= vps\_non\_vui\_extension\_length; i++ ) |  |
| **vps\_non\_vui\_extension\_data\_byte** | u(8) |
| **vps\_vui\_present\_flag** | u(1) |
| 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++ ) { |  |
| currLsIdx = OlsIdxToLsIdx[ i ] |  |
| **sub\_layer\_flag\_info\_present\_flag**[ i ] | u(1) |
| for( j = 0; j <= MaxSubLayersInLayerSetMinus1[ currLsIdx ]; 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 < NumLayersInIdList[ currLsIdx ]; 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) |
| if( cross\_layer\_irap\_aligned\_flag ) |  |
| **all\_layers\_idr\_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 = vps\_base\_layer\_internal\_flag ? 0 : 1; i <= vps\_num\_layer\_sets\_minus1; i++ ) |  |
| for( j = 0; j <= MaxSubLayersInLayerSetMinus1[ i ]; 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) |
| } |  |
| } |  |
| **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) |
| **tiles\_not\_in\_use\_flag** | u(1) |
| if( !tiles\_not\_in\_use\_flag ) { |  |
| for( i = vps\_base\_layer\_internal\_flag ? 0 : 1; i <= MaxLayersMinus1; i++ ) { |  |
| **tiles\_in\_use\_flag**[ i ] | u(1) |
| if( tiles\_in\_use\_flag[ i ] ) |  |
| **loop\_filter\_not\_across\_tiles\_flag**[ i ] | u(1) |
| } |  |
| for( i = vps\_base\_layer\_internal\_flag ? 1 : 2; 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** | u(1) |
| if( !wpp\_not\_in\_use\_flag ) |  |
| for( i = vps\_base\_layer\_internal\_flag ? 0 : 1; 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) |
| **vert\_phase\_position\_in\_use\_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++ ) |  |
| if( vps\_base\_layer\_internal\_flag | | RefLayerId[ layer\_id\_in\_nuh[ i ] ][ j ] > 0 ) { |  |
| **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) |
| } |  |
| } |  |
| **vps\_vui\_bsp\_hrd\_present\_flag** | u(1) |
| if( vps\_vui\_bsp\_hrd\_present\_flag ) |  |
| vps\_vui\_bsp\_hrd\_parameters( ) |  |
| for( i = 1; i <= MaxLayersMinus1; i++ ) |  |
| if( NumDirectRefLayers[ layer\_id\_in\_nuh[ i ] ] = = 0 ) |  |
| **base\_layer\_parameter\_set\_compatibility\_flag**[ i ] | u(1) |
| } |  |

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 < NumLayersInIdList[ h ]; j++ ) |  |
| **layer\_in\_bsp\_flag**[ h ][ i ][ j ] | u(1) |
| if( num\_bitstream\_partitions[ h ] > 0 ) { |  |
| **num\_bsp\_sched\_combinations\_minus1**[ h ] | ue(v) |
| for( i = 0; i <= num\_bsp\_sched\_combinations\_minus1[ h ]; i++ ) |  |
| for( j = 0; j < num\_bitstream\_partitions[ h ]; j++ ) { |  |
| **bsp\_comb\_hrd\_idx**[ h ][ i ][ j ] | u(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) |
| if( nuh\_layer\_id = = 0 ) { |  |
| **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\_present\_flag** | u(1) |
| if( sps\_extension\_present\_flag ) { |  |
| **sps\_range\_extensions\_flag** | u(1) |
| **sps\_multilayer\_extension\_flag** | u(1) |
| **sps\_extension\_6bits** | u(6) |
| } |  |
| if( sps\_range\_extensions\_flag ) |  |
| sps\_range\_extensions( ) |  |
| if( sps\_multilayer\_extension\_flag ) |  |
| sps\_multilayer\_extension( ) |  |
| if( sps\_extension\_6bits ) |  |
| 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) |
| **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) |
| **vert\_phase\_position\_enable\_flag**[ scaled\_ref\_layer\_id[ i ] ] | u(1) |
| **}** |  |
| } |  |

* + - * 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\_present\_flag** | u(1) |
| if( pps\_extension\_present\_flag ) { |  |
| **pps\_range\_extensions\_flag** | u(1) |
| **pps\_multilayer\_extension\_flag** | u(1) |
| **pps\_extension\_6bits** | u(6) |
| } |  |
| if( pps\_range\_extensions\_flag ) |  |
| pps\_range\_extensions( ) |  |
| if( pps\_multilayer\_extension\_flag ) { |  |
| **poc\_reset\_info\_present\_flag** | u(1) |
| **colour\_mapping\_enabled\_flag** | u(1) |
| if( colour\_mapping\_enabled\_flag ) |  |
| colour\_mapping\_table( ) |  |
| } |  |
| if( pps\_extension\_6bits ) |  |
| 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. Colour mapping table syntax

[Ed. (JC): The “color mapping table syntax” has been inserted as a new subclause as done in the same way as for “F.7.3.5 Scaling list data syntax”. This has the consequence that the remaining subclause indices of subcluase F.7.3 do not match to the subclause indices of subclause 7.3 in version 1, RExt and MV-HEVC specifications] [Ed. (MH): Table row height in F.7.3.5.1 and F.7.3.5.2 looks smaller than in other syntax tables.]

* + - * 1. General colour mapping table syntax

|  |  |
| --- | --- |
| colour\_mapping\_table( ) { | **Descriptor** |
| **num\_cm\_ref\_layers\_minus1** | ue(v) |
| for( i = 0; i < num\_cm\_ref\_layers\_minus1 + 1; i++) { |  |
| **cm\_ref\_layer\_id**[ i ] | u(6) |
| **}** |  |
| **cm\_octant\_depth** | u(2) |
| **cm\_y\_part\_num\_log2** | u(2) |
| **cm\_input\_luma\_bit\_depth\_minus8** | u(3) |
| **cm\_input\_chroma\_bit\_depth\_minus8** | u(3) |
| **cm\_output\_luma\_bit\_depth\_minus8** | u(3) |
| **cm\_output\_chroma\_bit\_depth\_minus8** | u(3) |
| **cm\_res\_quant\_bits** | u(2) |
| **cm\_delta\_flc\_bits\_minus1** | u(2) |
| if( cm\_octant\_depth = = 1 ) { |  |
| **cm\_adapt\_threshold\_u\_delta** | se(v) |
| **cm\_adapt\_threshold\_v\_delta** | se(v) |
| } |  |
| colour\_mapping\_octants( 0, 0, 0, 0, 1 << cm\_octant\_depth ) |  |
| } |  |

* + - * 1. Colour mapping octants syntax

|  |  |
| --- | --- |
| colour\_mapping\_octants( depth, yIdx, uIdx, vIdx, length ) { [Ed. (MH): Camel casing suggested for all input parameters, i.e. depth and length should be changed.] | **Descriptor** |
| if ( depth < cm\_octant\_depth ) |  |
| **split\_octant\_flag** | u(1) |
| if ( split\_octant\_flag ) |  |
| for( k = 0; k < 2; k++ ) |  |
| for( m = 0; m < 2 ; m++ ) |  |
| for( n = 0; n < 2; n++ ) |  |
| colour\_mapping\_octants( depth + 1, yIdx + YPartNum \* k \* length / 2, uIdx + m \* length / 2, vIdx + n \* length / 2, length / 2) |  |
| else |  |
| for( i = 0; i < YPartNum; i++ ) { |  |
| yIdxShift = yIdx + ( ( i << ( cm\_octant\_depth – depth ) ) |  |
| for( j = 0; j < 4; j++ ) { |  |
| **coded\_res\_flag**[ yIdxShift ][ uIdx ][ vIdx ][ j ] | u(1) |
| if( **coded\_res\_flag**[ yIdxShift ][ uIdx ][ vIdx ][ j ] ) |  |
| for( c = 0; c < 3; c++ ) { |  |
| **res\_coeff\_q**[ yIdxShift ][ uIdx ][ vIdx ][ j ][ c ] ) | ue(v) |
| **res\_coeff\_r**[ yIdxShift ][ uIdx ][ vIdx ][ j ][ c ] ) | u(v) |
| if( res\_coeff\_q[ yIdxShift ][ uIdx ][ vIdx ][ j ][ c ] | | res\_coeff\_r[ yIdxShift ][ uIdx ][ vIdx ][ j ][ c ]) |  |
| **res\_coeff\_s**[ yIdxShift ][ uIdx ][ vIdx ][ j ][ c ] ) | u(1) |
| **}** |  |
| } |  |
| } |  |
| } |  |

* + - 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) |
| } |  |
| 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) |
| } |  |
| } |  |
| for( i = 0; i < NumActiveRefLayerPics; i++ ) |  |
| if ( vert\_phase\_position\_enable\_flag[ RefPicLayerId[ i ] ] ) |  |
| **vert\_phase\_position\_flag[** RefPicLayerId[ i ]**]** | u(1) |
| 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) |
| if( poc\_reset\_info\_present\_flag ) |  |
| **poc\_reset\_idc** | u(2) |
| if( poc\_reset\_idc != 0 ) |  |
| **poc\_reset\_period\_id** | u(6) |
| if( poc\_reset\_idc = = 3 ) { |  |
| **full\_poc\_reset\_flag** | u(1) |
| **poc\_lsb\_val** | u(v) |
| } |  |
| if( !PocMsbValRequiredFlag && vps\_poc\_lsb\_aligned\_flag ) |  |
| **poc\_msb\_val\_present\_flag** | u(1) |
| if( poc\_msb\_val\_present\_flag ) |  |
| **poc\_msb\_val** | ue(v) |
| while( more\_data\_in\_slice\_segment\_header\_extension( ) ) |  |
| **slice\_segment\_header\_extension\_data\_bit** | u(1) |
| } |  |
| 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.

The variable CraOrBlaPicFlag is derived as follows:

CraOrBlaPicFlag = ( nal\_unit\_type = = BLA\_W\_LP | | nal\_unit\_type = = BLA\_N\_LP | |  
 nal\_unit\_type = = BLA\_W\_RADL | | nal\_unit\_type = = CRA\_NUT )

NOTE 1 – When a picture picA that is a CRA picture and belongs to a layer with nuh\_layer\_id equal to layerId is present in a bitstream and pictures belonging to the layer with nuh\_layer\_id equal to layerId and precede, in decoding order, the picture picA are dropped due to layer down-switching followed by layer up-switching, the RASL pictures associated with the picture picA, if any, may have some reference pictures that may not be available for reference unless one of the following conditions is true:

– The access unit auA containing the picture picA is an IRAP access unit, and the picture with nuh\_layer\_id equal to 0 in the access unit auA, if any, has NoClrasOutputFlag equal to 1.

– The value of HandleCraAsBlaFlag is equal to 1 for the CRA picture picA.

**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 2 – 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. Decoders shall ignore NAL units with nal\_unit\_type equal to VPS\_NUT and nuh\_layer\_id greater than 0. [Ed. (YK): Check the need of adding a wording like "Although the value of nuh\_layer\_id is required to be equal to 0 when nal\_unit\_type is equal to VPS\_NUT in this version of this Specification, decoders shall allow other values of nuh\_layer\_id in the range of 0 to 62, inclusive, to appear in the syntax when nal\_unit\_type is equal to VPS\_NUT."]

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.

When nal\_unit\_type is equal to EOS\_NUT, the value of nuh\_layer\_id shall be equal to 0. Decoders shall ignore (i.e. remove from the bitstream and discard) all NAL units with nal\_unit type equal to EOS\_NUT and nuh\_layer\_id greater than 0. [Ed. (YK): Check the need of adding the wording like "Although the value of nuh\_layer\_id is required to be equal to 0 when nal\_unit\_type is equal to EOB\_NUT in this version of this Specification, decoders shall allow other values of nuh\_layer\_id in the range of 0 to 62, inclusive, to appear in the syntax when nal\_unit\_type is equal to EOB\_NUT."]

When nal\_unit\_type is equal to EOB\_NUT, the value of nuh\_layer\_id shall be equal to 0. Decoders shall ignore (i.e. remove from the bitstream and discard) all NAL units with nal\_unit type equal to EOB\_NUT and nuh\_layer\_id greater than 0. [Ed. (YK): Check the need of adding the wording like "Although the value of nuh\_layer\_id is required to be equal to 0 when nal\_unit\_type is equal to EOB\_NUT in this version of this Specification, decoders shall allow other values of nuh\_layer\_id in the range of 0 to 62, inclusive, to appear in the syntax when nal\_unit\_type is equal to EOB\_NUT."]

**nuh\_temporal\_id\_plus1** minus 1 specifies a temporal identifier for the NAL unit. The value of nuh\_temporal\_id\_plus1 shall not be equal to 0.

The variable TemporalId is specified as follows:

TemporalId = nuh\_temporal\_id\_plus1 − 1 (F‑1)

If nal\_unit\_type is in the range of BLA\_W\_LP to RSV\_IRAP\_VCL23, inclusive, i.e. the coded slice segment belongs to an IRAP picture, TemporalId shall be equal to 0. Otherwise, if nal\_unit\_type is equal to TSA or TSA\_N, TemporalId shall not be equal to 0. Otherwise, when nuh\_layer\_id is equal to 0 and nal\_unit\_type is equal to STSA\_R or STSA\_N, TemporalId shall not be equal to 0.

The value of TemporalId shall be the same for all VCL NAL units of an access unit. The value of TemporalId of an access unit is the value of the TemporalId of the VCL NAL units of the access unit.

The value of TemporalId for non-VCL NAL units is constrained as follows:

– If nal\_unit\_type is equal to VPS\_NUT or SPS\_NUT, TemporalId shall be equal to 0 and the TemporalId of the access unit containing the NAL unit shall be equal to 0.

– Otherwise if nal\_unit\_type is equal to EOS\_NUT or EOB\_NUT, TemporalId shall be equal to 0.

– Otherwise, if nal\_unit\_type is equal to AUD\_NUT or FD\_NUT, TemporalId shall be equal to the TemporalId of the access unit containing the NAL unit.

– Otherwise, TemporalId shall be greater than or equal to the TemporalId of the access unit containing the NAL unit.

NOTE 3 – When the NAL unit is a non-VCL NAL unit, the value of TemporalId is equal to the minimum value of the TemporalId values of all access units to which the non-VCL NAL unit applies. When nal\_unit\_type is equal to PPS\_NUT, TemporalId may be greater than or equal to the TemporalId of the containing access unit, as all PPSs may be included in the beginning of a bitstream, wherein the first coded picture has TemporalId equal to 0. When nal\_unit\_type is equal to PREFIX\_SEI\_NUT or SUFFIX\_SEI\_NUT, TemporalId may be greater than or equal to the TemporalId of the containing access unit, as an SEI NAL unit may contain information, e.g. in a buffering period SEI message or a picture timing SEI message, that applies to a bitstream subset that includes access units for which the TemporalId values are greater than the TemporalId of the access unit containing the SEI NAL unit.

* + - * 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.

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 with the following modification:

*Replace*

It is a requirement of bitstream conformance that, when present, the next access unit after an access unit that contains an end of sequence NAL unit or an end of bitstream NAL unit shall be an IRAP access unit, which may be an IDR access unit, a BLA access unit, or a CRA access unit.

*with the following (removing ", which may be an IDR access unit, a BLA access unit, or a CRA access unit"):*

It is a requirement of bitstream conformance that, when present, the next access unit after an access unit that contains an end of sequence NAL unit or an end of bitstream NAL unit shall be an IRAP access unit.

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

This clause specifies the order of NAL units and coded pictures and their association to access unit for CVSs that contain NAL units with nuh\_layer\_id greater than 0 that are decoded using the decoding processes specified in Annex F, Annex G and Annex H.

An access unit consists of one or more coded pictures with different values of nuh\_layer\_id and zero or more non-VCL NAL units. The association of VCL NAL units to coded pictures is described in subclause 7.4.2.4.5.

The first access unit in the bitstream starts with the first NAL unit of the bitstream.

A VCL NAL unit is the first VCL NAL unit of an access unit, when all of the following conditions are true: [Ed. (YK): These conditions seem sufficient but more than necessary for a VCL NAL unit to be the first VCL NAL unit of an AU. For example, it seems possible to have back-to-back AUs in the same POC resetting period and having the same PicOrderCntVal.]

– first\_slice\_segment\_in\_pic\_flag is equal to 1.

– At least one of the following conditions is true:

– The previous picture in decoding order belongs to a different POC resetting period than the picture containing the VCL NAL unit.

– PicOrderCntVal derived for the VCL NAL unit differs from the PicOrderCntVal of the previous picture in decoding order.

NOTE 1 – Additionally, the following conditions could but need not be used:

– The nuh\_layer\_id value of the VCL NAL unit is equal to 0.

– vps\_poc\_lsb\_aligned\_flag is equal to 1 and the slice\_pic\_order\_cnt\_lsb value of the VCL NAL unit differs from the slice\_pic\_order\_cnt\_lsb value of the previous VCL NAL unit in decoding order.

The first of any of the following NAL units specifies the start of a new access unit:

– Access unit delimiter NAL unit (when present).

– The first NAL unit (when present), in decoding order, of a contiguous sequence of one or more of any of the following NAL units in any order, when the sequence of NAL units immediately precedes the first VCL NAL unit of an access unit:

– VPS NAL unit (when present)

– SPS NAL unit (when present)

– PPS NAL unit (when present)

– Prefix SEI NAL unit (when present)

– NAL units with nal\_unit\_type in the range of RSV\_NVCL41..RSV\_NVCL44 (when present)

– NAL units with nal\_unit\_type in the range of UNSPEC48..UNSPEC55 (when present)

– The first VCL NAL unit of an access unit (always present).

The order of the coded pictures and non-VCL NAL units within an access unit shall obey the following constraints:

– When an access unit delimiter NAL unit is present, it shall be the first NAL unit. There shall be at most one access unit delimiter NAL unit in any access unit.

– When any prefix SEI NAL units are present, they shall not follow the last VCL NAL unit of the access unit.

– NAL units having nal\_unit\_type equal to FD\_NUT or SUFFIX\_SEI\_NUT, or in the range of RSV\_NVCL45..RSV\_NVCL47 or UNSPEC56..UNSPEC63 shall not precede the first VCL NAL unit of the access unit.

– When an end of sequence NAL unit is present, it shall be the last NAL unit in the access unit other than an end of bitstream NAL unit (when present).

– When an end of bitstream NAL unit is present, it shall be the last NAL unit in the access unit.

NOTE 2 – VPS NAL units, SPS NAL units, PPS NAL units, prefix SEI NAL units, or NAL units with nal\_unit\_type in the range of RSV\_NVCL41..RSV\_NVCL44 or UNSPEC48..UNSPEC55, may be present in an access unit, but cannot follow the last VCL NAL unit of the access unit, as this condition would specify the start of a new access unit.

Order of VCL NAL units and association to coded pictures

The specifications in subclause 7.4.2.4.5 apply.

Order of VCL NAL units and association to picture units

[Ed. (MH): The exact specification of a picture unit is missing. There is a decision to add such text, as response to Proposal 2.2-4 of JCTVC-Q0183. The decision is documented in the BoG notes JCTVC-Q0223 as follows: "agreed in spirit. Additional text is also needed to define which NAL units are associated with a picture unit."]

* + - 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.
* *Remove the semantics of* vps\_reserved\_three\_2bits.
* *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 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. Each output operation point is associated with an operation point, a list of nuh\_layer\_id values of the output layers, in increasing order of nuh\_layer\_id values, denoted as OptLayerIdList, and the OpTid of the associated operation point. The OpLayerIdList of the operation point associated with an output operation point is also referred to as the OpLayerIdList of the output operation point."

**vps\_base\_layer\_internal\_flag** equal to 0 specifies that the base layer is provided by an external means not specified in this Specification. vps\_base\_layer\_internal\_flag equal to 1 specifies that the base layer is provided in the bitstream.

When vps\_base\_layer\_internal\_flag is equal to 0, the following applies:

* The value of vps\_sub\_layer\_ordering\_info\_present\_flag shall be equal to 0.
* The values of vps\_max\_dec\_pic\_buffering\_minus1[ i ], vps\_max\_num\_reorder\_pics[ i ], and vps\_max\_latency\_increase\_plus1[ i ] shall all be equal to 0 for all possible values of i.
* Decoders shall ignore the values of vps\_sub\_layer\_ordering\_info\_present\_flag, vps\_max\_dec\_pic\_buffering\_minus1[ i ], vps\_max\_num\_reorder\_pics[ i ], and vps\_max\_latency\_increase\_plus1[ i ].
* The value of hrd\_layer\_set\_idx[ i ] shall be greater than 0.

**vps\_reserved\_one\_bit** shall be equal to 1 in bitstreams conforming to this version of this Specification. The value 0 for vps\_reserved\_one\_bit is reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore the value of vps\_reserved\_one\_bit.

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

**vps\_num\_hrd\_parameters** specifies the number of hrd\_parameters( ) syntax structures present in the VPS RBSP. The value of vps\_num\_hrd\_parameters shall be in the range of 0 to vps\_num\_layer\_sets\_minus1 + 1, inclusive.

**hrd\_layer\_set\_idx**[ i ] specifies the index, into the list of layer sets specified by the VPS, of the layer set to which the i‑th hrd\_parameters( ) syntax structure in the VPS applies. The value of hrd\_layer\_set\_idx[ i ] shall be in the range of ( vps\_base\_layer\_internal\_flag ? 0 : 1 ) to vps\_num\_layer\_sets\_minus1, inclusive.

It is a requirement of bitstream conformance that the value of hrd\_layer\_set\_idx[ i ] shall not be equal to the value of hrd\_layer\_set\_idx[ j ] for any value of j not equal to i.

**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

**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/quality 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‑2)

– 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, the variable DependencyId[ layer\_id\_in\_nuh[ i ] ] specifying the spatial/quality 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 ) {

newViewFlag = 1

for( j = 0; j < i; j++ )

if( ViewOrderIdx[ lId ]  = =  ViewOrderIdx[ layer\_id\_in\_nuh[ j ] ] )  
 newViewFlag = 0

NumViews += newViewFlag

}

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** | **SEI message describing interpretation of auxiliary pictures** |
| 1 | AUX\_ALPHA | Alpha plane | Alpha channel information |
| 2 | AUX\_DEPTH | Depth picture | Depth representation information |
| 3..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.

SEI messages may describe the interpretation of auxiliary pictures, including their possible association with one or more primary pictures.

NOTE 4 – Unless constrained by the semantics of the SEI messages specifying the interpretation of auxiliary pictures, it is allowed to have two layers with nuh\_layer\_id values layerIdA and layerIdB such that AuxId[ layerIdA ] is equal to AuxId[ layerIdB ], both being greater than 0, and to have all values of ScalabilityId[ LayerIdxInVps[ layerIdA ] ][ i ] equal to ScalabilityId[ LayerIdxInVps[ layerIdB ] ][ i ] for each value of i in the range of 0 to 15, inclusive. SEI messages specifying the interpretation of auxiliary pictures may specify that a picture with nuh\_layer\_id equal to layerIdA and a picture with nuh\_layer\_id equal to layerIdB in the same access unit may both be associated with the same primary picture.

**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 62, inclusive.

– For each layer with nuh\_layer\_id equal to currLayerId, and for all values of j in the range of 0 to 62, 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 ]  
}

The variables NumPredictedLayers[ i ] and PredictedLayerId[ i ][ j ] are derived as follows:

for( i = 0; i < MaxLayersMinus1; i++ ) {  
 iNuhLId = layer\_id\_in\_nuh[ i ]  
 for( j = iNuhLId + 1, predIdx = 0; j < 63; j++ )  
 if( recursiveRefLayerFlag[ j ][ iNuhLId ] )  
 PredictedLayerId[ iNuhLId ][ predIdx++ ] = j  
 NumPredictedLayers[ iNuhLId ] = predIdx  
}

The variables NumIndependentLayers, NumLayersInTreePartition[ i ], and TreePartitionLayerIdList[ i ][ j ] for i in the range of 0 to NumIndependentLayers − 1, inclusive, and j in the range of 0 to NumLayersInTreePartition[ i ] − 1, inclusive, are derived as follows:

for( i = 0; i <= MaxLayersMinus1; i++ )  
 countedLayerIdxFlag[ i ] = 0  
for( i = 0, k = 0; i <= MaxLayersMinus1; i++ ) {  
 iNuhLId = layer\_id\_in\_nuh[ i ]  
 if( NumDirectRefLayers[ iNuhLId ] = = 0 ) {  
 TreePartitionLayerIdList[ k ][ 0 ] = iNuhLId  
 NumLayersInTreePartition[ k ] = 1  
 for( j = 0; j < NumPredictedLayers[ iNuhLId ]; j++ )  
 if( !countedLayerIdxFlag[ LayerIdxInVps[ PredictedLayerId[ iNuhLId ][ j ] ] ] ) {  
 TreePartitionLayerIdList[ k ][ NumLayersInTreePartition[ k ] ] = PredictedLayerId[ iNuhLId ][ j ]  
 NumLayersInTreePartition[ k ]++  
 countedLayerIdxFlag[ LayerIdxInVps[ PredictedLayerId[ iNuhLId ][ j ] ] ] = 1  
 }  
 k++  
 }  
 NumIndependentLayers = k  
}

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 layerId equal to layer\_id\_in\_nuh[ i ] such that layerId is greater than or equal to ( vps\_base\_layer\_internal\_flag ? 0 : 1 ). When vps\_base\_layer\_internal\_flag is equal to 0, sub\_layers\_vps\_max\_minus1[ 0 ] constrains the access units for which a decoded picture with nuh\_layer\_id equal to 0 may be provided by external means as follows: a decoded picture with nuh\_layer\_id equal to 0 cannot be provided by external means for decoding of an access unit with TemporalId greater than sub\_layers\_vps\_max\_minus1[ 0 ]. 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.

The variable MaxSubLayersInLayerSetMinus1[ i ] is derived as follows:

for( i = 0; i < NumLayerSets; i++ ) {  
 maxSlMinus1 = 0  
 for( k = 0; k < NumLayersInIdList[ i ]; k++ ) {  
 lId = LayerSetLayerIdList[ i ][ k ] (F‑3)  
 maxSlMinus1 = Max( maxSLMinus1, sub\_layers\_vps\_max\_minus1[ LayerIdxInVps[ lId ] ] )  
 }  
 MaxSubLayersInLayerSetMinus1[ i ] = maxSlMinus1  
}

**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 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 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\_num\_profile\_tier\_level\_minus1** plus 1 specifies the number of profile\_tier\_level( ) syntax structures in the VPS. The value of vps\_num\_profile\_tier\_level\_minus1 shall be in the range of 0 to 63, inclusive.

**vps\_profile\_present\_flag**[ i ] equal to 1 specifies that profile and tier information 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.

**num\_add\_layer\_sets** specifies the number of additional layer sets. When not present, num\_add\_layer\_sets is inferred to be equal to 0. num\_add\_layer\_sets shall be in the range of 0 to 1023, inclusive.

The variable NumLayerSets is derived as follows:

NumLayerSets = vps\_num\_layer\_sets\_minus1 + 1 + num\_add\_layer\_sets

When num\_add\_layer\_sets is greater than 0, the variables FirstAddLayerSetIdx and LastAddLayerSetIdx are derived as follows:

FirstAddLayerSetIdx = vps\_num\_layer\_sets\_minus1 + 1   
LastAddLayerSetIdx = FirstAddLayerSetIdx + num\_add\_layer\_sets − 1

When num\_add\_layer\_sets is greater than 0, it is a requirement of bitstream conformance that the following applies:

– When the non-base layer subtree extraction process of subclause F.10.2 is applied with the input variable lsIdx equal to vps\_num\_layer\_sets\_minus1 + 1 + i for any value of i in the range of 0 to num\_add\_layer\_sets − 1, inclusive, and NumLayersInIdList[ lsIdx ] is equal to 1, the output of the process of subclause F.10.2 shall be a conforming bitstream except that the output bitstream is not required to contain any VPS NAL units.

– When the non-base layer subtree extraction process of subclause F.10.2 is applied with the input variable lsIdx equal to vps\_num\_layer\_sets\_minus1 + 1 + i for any value of i in the range of 0 to num\_add\_layer\_sets − 1, inclusive, and NumLayersInIdList[ lsIdx ] is greater than 1, the output of the process of subclause F.10.2 shall be a conforming bitstream.

**highest\_layer\_idx\_plus1**[ i ][ j ] specifies the values of NumLayersInIdList[ vps\_num\_layer\_sets\_minus1 + 1 + i ] and LayerSetLayerIdList[ vps\_num\_layer\_sets\_minus1 + 1 + i ][ layerNum ] and is used to infer layer\_id\_included\_flag[ vps\_num\_layer\_sets\_minus1 + 1 + i ][ layerId ] as follows:

layerNum = 0  
lsIdx = vps\_num\_layer\_sets\_minus1 + 1 + i  
for( layerId = 0; layerId <= 62; layerId++ )  
 layer\_id\_included\_flag[ lsIdx ][ layerId ] = 0 [Ed. (GT), Assignment to syntax element should be changed. ]  
for( treeIdx = 1; treeIdx < NumIndependentLayers; treeIdx++ )  
 for( layerCnt = 0; layerCnt < highest\_layer\_idx\_plus1[ i ][ j ]; layerCnt++ ) {  
 LayerSetLayerIdList[ lsIdx ][ layerNum ] = TreePartitionLayerIdList[ treeIdx ][ layerCnt ]  
 layer\_id\_included\_flag[ lsIdx ][ TreePartitionLayerIdList[ treeIdx ][ layerCnt ] ] = 1  
 layerNum++  
 }  
NumLayersInIdList[ lsIdx ] = layerNum

The value of highest\_layer\_idx\_plus1[ i ][ j ] shall be in the range of 0 to NumLayersInTreePartition[ j ], inclusive.

The length of highest\_layer\_idx\_plus1[ i ][ j ] is equal to Ceil( Log2( NumLayersInTreePartition[ j ] + 1 ) ).

It is a requirement of bitstream conformance that NumLayersInIdList[ vps\_num\_layer\_sets\_minus1 + 1 + i ] shall be greater than 0.

AssignedBaseLayerId[ vps\_num\_layer\_sets\_minus1 + 1 + i ] is set equal to the smallest nuh\_layer\_id value in NumLayersInIdList[ vps\_num\_layer\_sets\_minus1 + 1 + i ].

It is a requirement of bitstream conformance that each SPS or PPS that is active for the layer with nuh\_layer\_id equal to AssignedBaseLayerId[ vps\_num\_layer\_sets\_minus1 + 1 + i ] shall have nuh\_layer\_id equal to 0.

**num\_add\_olss** specifies the number of OLSs in addition to the first NumLayerSets OLSs specified by the VPS. The value of num\_add\_olss shall be in the range of 0 to 1023, inclusive. When not present, the value of num\_add\_olss is inferred to be equal to 0.

**default\_output\_layer\_idc** specifies the derivation of the output layers for the OLSs with index in the range of 1 to vps\_num\_layer\_sets\_minus1, inclusive. default\_output\_layer\_idc equal to 0 specifies that all layers in each of the OLSs with index in the range of 1 to vps\_num\_layer\_sets\_minus1, inclusive, are output layers of their respective OLSs. default\_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 OLSs with index in the range of 1 to vps\_num\_layer\_sets\_minus1, inclusive, is an output layer of its OLS. default\_output\_layer\_idc equal to 2 specifies that the output layers for the OLSs with index in the range of 1 to vps\_num\_layer\_sets\_minus1, inclusive, are specified with the syntax elements output\_layer\_flag[ i ][ j ]. The value of 3 for default\_output\_layer\_idc is reserved for future use by ITU-T | ISO/IEC. Although the value of default\_output\_layer\_idc is required to be less than 3 in this version of this Specification, decoders shall allow a value of default\_output\_layer\_idc equal to 3 to appear in the syntax.

The variable defaultOutputLayerIdc is set equal to Min( default\_output\_layer\_idc, 2 ).

**layer\_set\_idx\_for\_ols\_minus1**[ i ] plus 1specifies the index of the layer set for the i-th OLS. The value of layer\_set\_idx\_for\_ols\_minus1[ i ] shall be in the range of 0 to NumLayerSets – 2, inclusive. The length of the layer\_set\_idx\_for\_ols\_minus1[ i ] syntax element is Ceil( Log2( NumLayerSets – 1 ) ) bits.

For i in the range of 0 to NumOutputLayerSets ‑ 1, inclusive, the variable OlsIdxToLsIdx[ i ] is derived as specified in the following:

OlsIdxToLsIdx[ i ] = ( i < NumLayerSets ) ? i : layer\_set\_idx\_for\_ols\_minus1[ i ] + 1 (F‑4)

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

The value of output\_layer\_flag[ 0 ][ 0 ] is inferred to be equal to 1.

When defaultOutputLayerIdc is equal to 0 or 1, for i in the range of 0 to vps\_num\_layer\_sets\_minus1, inclusive, and j in the range of 0 to NumLayersInIdList[ OlsIdxToLsIdx[ i ] ] − 1, inclusive, the variable OutputLayerFlag[ i ][ j ] is derived as follows:

– If defaultOutputLayerIdc is equal to 0 or LayerSetLayerIdList[ OlsIdxToLsIdx[ i ] ][ j ] is equal to nuhLayerIdA, with nuhLayerIdA being the highest value in LayerSetLayerIdList[ OlsIdxToLsIdx[ i ] ] with AuxId[ nuhLayerIdA ] equal to 0, OutputLayerFlag[ i ][ j ] is set equal to 1.

– Otherwise, OutputLayerFlag[ i ][ j ] is set equal to 0.

For i in the range of ( defaultOutputLayerIdc  = =  2 ) ? 0 : ( vps\_num\_layer\_sets\_minus1 + 1 ) to NumOutputLayerSets − 1, inclusive, and j in the range of 0 to NumLayersInIdList[ OlsIdxToLsIdx[ i ] ] − 1, inclusive, the variable OutputLayerFlag[ i ][ j ] is set equal to output\_layer\_flag[ i ][ j ].

The variable NumOutputLayersInOutputLayerSet[ i ] is derived as follows:

NumOutputLayersInOutputLayerSet[ i ] = 0  
for( j = 0 ; j < NumLayersInIdList[ OlsIdxToLsIdx[ i ] ]; j++) {   
 NumOutputLayersInOutputLayerSet[ i ] += OutputLayerFlag[ i ][ j ]  
 if( OutputLayerFlag[ i ][ j ] )  
 OlsHighestOutputLayerId[ i ] = LayerSetLayerIdList[ OlsIdxToLsIdx[ i ] ][ j ]

It is a requirement of bitstream conformance that NumOutputLayersInOutputLayerSet[ i ] shall be greater than 0 for i in the range of 0 to NumOutputLayers − 1, inclusive.

**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 OLS. When num\_add\_layer\_sets is greater than 0 and OlsIdxToLsIdx[ i ] is in the range of FirstAddLayerSetIdx to LastAddLayerSetIdx, inclusive, the profile\_tier\_level( ) syntax structure applies to the output of the non-base layer subtree extraction process of subclause F.10.2 with the input variable lsIdx set equal to OlsIdxToLsIdx[ i ], where the active VPSs of the output bitstream outBitstream, if any, shall contain an OLS specifying the output of the same layers as the i-th OLS of the current VPS. [Ed. (MH): The sentence could be editorially improved to more specifically state that the new base layer had nuh\_layer\_id equal to AssignedBaseLayerId in the inBitstream.] 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 ] for i in the range of 1 to NumOutputLayerSet − 1, inclusive, shall be in the range of ( vps\_base\_layer\_internal\_flag ? 0 : 1 ) to vps\_num\_profile\_tier\_level\_minus1, inclusive.

**alt\_output\_layer\_flag**[ i ] equal to 0 specifies that an alternative output layer is not used for any output layer in the i-th OLS. alt\_output\_layer\_flag[ i ] equal to 1 specifies that an alternative output layer may be used for the output layer in the i-th OLS.

– If NumOutputLayersInOutputLayerSet[ i ] is equal to 1 and NumDirectRefLayers[ OlsHighestOutputLayerId[ i ] ] is greater than 0, the variable AltOptLayerFlag[ i ] is set equal to alt\_output\_layer\_flag[ i ].

– Otherwise, the variable AltOptLayerFlag[ i ] is set equal to 0.

AltOptLayerFlag[ 0 ] is set equal to 0.

NOTE 6 – When AltOptLayerFlag[ olsIdx ] is equal to 0, pictures that are not at the output layers of the OLS with index olsIdx are not output. When AltOptLayerFlag[ olsIdx ] is equal to 1 and a picture at the output layer of the OLS with index olsIdx is not present in an access unit or has PicOutputFlag equal to 0, a picture with highest nuh\_layer\_id among those pictures of the access unit for which PicOutputFlag is equal to 1 and which has nuh\_layer\_id value among the nuh\_layer\_id values of the direct and indirect reference layers of the output layer is output.

For each value of olsIdx in the range of 0 to NumOutputLayerSets − 1, inclusive, the following applies:

– When AltOptLayerFlag[ olsIdx ] is equal to 1, the value of pic\_output\_flag shall be the same in the slice headers of an access unit that have nuh\_layer\_id value equal to OlsHighestOutputLayerId[ olsIdx ] or equal to the nuh\_layer\_id value of any direct or indirect reference layer of the layer with nuh\_layer\_id equal to OlsHighestOutputLayerId[ olsIdx ].

– Let olsBitstream be the output of the sub-bitstream extraction process with inputs of the current bitstream, TemporalId equal to 7 and layerIdListTarget equal to LayerSetLayerIdList[ OlsIdxToLsIdx[ olsIdx ] ]. Let truncatedOlsBitstream be olsBitstream or be formed from the olsBitstream by removing access units preceding, in decoding order, any access unit with an IRAP picture having nuh\_layer\_id equal to 0. It is a requirement of bitstream conformance that when AltOptLayerFlag[ olsIdx ] is equal to 1, a bitstream that is formed by removing, from the truncatedOlsBitstream, any coded picture that is not used as a reference for prediction for any other picture and is not the only coded picture of an access unit is a conforming bitstream.

NOTE 7 – When AltOptLayerFlag[ olsIdx ] is equal to 1, encoders are required to set the values of max\_vps\_dec\_pic\_buffering\_minus1[ i ][ k ][ j ] such that these values suffice also when pictures of an alternative output layer are marked as "needed for output" in the HRD.

**vps\_num\_rep\_formats\_minus1** plus 1 specifies the number of the following rep\_format( ) syntax structures in the VPS. The value of vps\_num\_rep\_formats\_minus1 shall be in the range of 0 to 255, inclusive.

**rep\_format\_idx\_present\_flag** equal to 1 specifies that the syntax elements vps\_rep\_format\_idx[ i ] are present. rep\_format\_idx\_present\_flag equal to 0 specifies that the syntax elements vps\_rep\_format\_idx[ i ] are not present. When not present, the value of rep\_format\_idx\_present\_flag is inferred to be equal to 0.

**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 Min( i, vps\_num\_rep\_formats\_minus1 ). The value of vps\_rep\_format\_idx[ i ] shall be in the range of 0 to vps\_num\_rep\_formats\_minus1, inclusive. The number of bits used for the representation of vps\_rep\_format\_idx[ i ] is Ceil( Log2( vps\_num\_rep\_formats\_minus1 + 1 ) ).

**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.

**vps\_poc\_lsb\_aligned\_flag** equal to 0 specifies that the value of slice\_pic\_order\_cnt\_lsb may or may not be the same in different pictures of an access unit. vps\_poc\_lsb\_aligned\_flag equal to 1 specifies that the value of slice\_pic\_order\_cnt\_lsb is the same in all pictures of an access unit. Additionally, the value of vps\_poc\_lsb\_aligned\_flag affects the decoding process for picture order count in subclause F.8.3.1. When not present, vps\_poc\_lsb\_aligned\_flag is inferred to be equal to 0.

**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.

It is a requirement of bitstream conformance that when poc\_lsb\_not\_present\_flag[ i ] is equal to 1, for any picture picA that has nuh\_layer\_id equal to layer\_id\_in\_nuh[ i ] and refers to the VPS, the following applies:

– When slice\_pic\_order\_cnt\_lsb is greater than 0, poc\_reset\_idc shall not be equal to 2.

– When full\_poc\_reset\_flag is equal to 1, poc\_lsb\_val shall be equal to 0.

[Ed. (GT) Consider moving above constraints to semantics of poc\_reset\_idc and poc\_lsb\_val]

**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 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 1 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 ]. 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 both inter-layer motion prediction and 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 vps\_base\_layer\_internal\_flag is equal to 1 and direct\_dependency\_type[ i ][ j ] is not present, the value of direct\_dependency\_type[ i ][ j ] is inferred to be equal to default\_direct\_dependency\_type.

When vps\_base\_layer\_internal\_flag is equal to 0, the value of direct\_dependency\_type[ i ][ 0 ] for i in the range of 1 to MaxLayersMinus1, inclusive, is inferred to be equal to 0.

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

VpsInterLayerSamplePredictionEnabled[ i ][ j ] = ( direct\_dependency\_type[ i ][ j ] + 1 ) & 0x1 (F‑5)

VpsInterLayerMotionPredictionEnabled[ i ][ j ] = ( direct\_dependency\_type[ i ][ j ] + 1 ) & 0x2 (F‑6)

**vps\_non\_vui\_extension\_length** specifies the length of the non-VUI VPS extension data following this syntax element and before vps\_vui\_present\_flag, in bytes. The value of vps\_non\_vui\_extension\_length shall be in the range of 0 to 4096, inclusive.

**vps\_non\_vui\_extension\_data\_byte** may have any value. Decoders shall ignore the value of vps\_non\_vui\_extension\_data\_byte. Its value does not affect decoder conformance to profiles specified in this version of this Specification.

**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\_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 are inferred. 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 the syntax elements chroma\_format\_vps\_idc, separate\_colour\_plane\_vps\_flag, bit\_depth\_vps\_luma\_minus8, and bit\_depth\_vps\_chroma\_minus8 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

For the lsIdx-th layer set, the number of sub-DPBs is NumLayersInIdList[ lsIdx ], and for each layer with a particular value of nuh\_layer\_id in the layer set, the sub-DPB with index layerIdx is assigned, where LayerSetLayerIdList[ lsIdx ][ layerIdx ] is equal to nuh\_layer\_id.

**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 MaxSubLayersInLayerSetMinus1[ OlsIdxToLsIdx[ i ] ], 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 NumLayersInIdList[ OlsIdxToLsIdx[ 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 NumLayersInIdList[ OlsIdxToLsIdx[ 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. When not present, the value of sub\_layer\_dpb\_info\_present\_flag[ i ][ j ] for j greater than 0 and any possible value of i, is inferred to be equal to be equal to 0.

**max\_vps\_dec\_pic\_buffering\_minus1**[ i ][ k ][ j ] plus 1 specifies the maximum number of decoded pictures, of the k-th layer for the CVS in the i-th OLS, that need to be stored in the DPB 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 MaxSubLayersInLayerSetMinus1[ OlsIdxToLsIdx[ i ] ], inclusive, it is inferred to be equal to max\_vps\_dec\_pic\_buffering\_minus1[ i ][ k ][ j − 1 ]. The value of max\_vps\_dec\_pic\_buffering\_minus1[ 0 ][ 0 ][ j ] is inferred to be equal to sps\_max\_dec\_pic\_buffering\_minus1[ j ] of the active SPS of the base layer. [Ed. (YK): Consider adding a note to say that, within the scope of this VPS, the inference needs to be performed again when a new SPS is activated for the base layer that has a different value of sps\_max\_dec\_pic\_buffering\_minus1[ j ], or other editorial changes if such inference is not appropriate. Similarly for the inferences of max\_vps\_num\_reorder\_pics[ 0 ][ j ] and max\_vps\_latency\_increase\_plus1[ 0 ][ j ] in below.]

**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 OLS 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 MaxSubLayersInLayerSetMinus1[ OlsIdxToLsIdx[ i ] ], 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]. The value of max\_vps\_num\_reorder\_pics[ 0 ][ j ] is inferred to be equal to sps\_max\_num\_reorder\_pics[ j ] of the active SPS of the base layer.

**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 OLS 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 MaxSubLayersInLayerSetMinus1[ OlsIdxToLsIdx[ i ] ], 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 ]. The value of max\_vps\_latency\_increase\_plus1[ 0 ][ j ] is inferred to be equal to sps\_max\_latency\_increase\_plus1[ j ] of the active SPS of the base layer.

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‑7)

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 vps\_vui\_present\_flag.

**all\_layers\_idr\_aligned\_flag** equal to 1 indicates that within each access unit for which the VCL NAL units refer to the VPS, when one picture is an IRAP picture, all the pictures in the same access unit are IDR pictures and have the same value of nal\_unit\_type. all\_layers\_idr\_aligned\_flag equal to 0 specifies that the above restriction may or may not apply. When not present, the value of all\_layers\_idr\_aligned\_flag is inferred to be equal to 0.

**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‑8)

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‑9)

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

( x & ( 214 − 1 ) ) = = 0 (F‑10)

**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‑11)

**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‑12)

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‑13)

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

avg\_pic\_rate[ i ] = = 0 (F‑14)

**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. When not present, the value of tiles\_not\_in\_use\_flag is inferred to be equal to 0.

**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. When not present, the value of tiles\_in\_use\_flag[ i ] is inferred to be equal to 0.

**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. When not present, the value of loop\_filter\_not\_across\_tiles\_flag[ i ] is inferred to be equal to 0.

**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. When not present, the value of wpp\_not\_in\_use\_flag is inferred to be equal to 0.

**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. When not present, the value of wpp\_in\_use\_flag[ i ] is inferred to be equal to 0.

[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 not present, the value of single\_layer\_for\_non\_irap\_flag is inferred to be equal to 0.

**higher\_layer\_irap\_skip\_flag** equal to 1 indicates that for every IRAP picture that refers 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 – 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 value of nuh\_layer\_id is an IRAP picture for which the decoded samples can be derived by applying the inter layer reference picture derivation process specified in subclause H.8.1.4 with the other picture with lower value of nuh\_layer\_id as input.

**vert\_phase\_position\_in\_use\_flag** equal to 0 indicates that the value of vert\_phase\_position\_enable\_flag is equal to 0 for each SPS referring to the VPS. vert\_phase\_position\_in\_use\_flag equal to 1 indicates that such a restriction may or may not apply.

**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‑15)  
 ( 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 refCroppedPicWidthInSamplesL[ i ][ j ] and refCroppedPicHeightInSamplesL[ i ][ j ] are set equal to the cropped picture width and cropped picture height in unit of luma sample, 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[ layer\_id\_in\_nuh[ j ] ]<<1, scaled\_ref\_layer\_top\_offset[ layer\_id\_in\_nuh[ j ] ]<<1, scaled\_ref\_layer\_right\_offset[ layer\_id\_in\_nuh[ j ] ]<<1, scaled\_ref\_layer\_bottom\_offset[ layer\_id\_in\_nuh[ 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‑16)

yP = ( ctbAddr / curPicWidthInCtbsY )  <<  curCtbLog2SizeY (F‑17)

– 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‑18)

curScaledRefLayerPicHeightInSamplesL[ i ][ j ] = curPicHeightInSamplesL[ i ] −   
 curScaledRefLayerTopOffset[ i ][ j ] − curScaledRefLayerBottomOffset[ i ][ j ] (F‑19)

scaleFactorX[ i ][ j ] = ( ( refCroppedPicWidthInSamplesL[ i ][ j ]  << 16 ) +   
( curScaledRefLayerPicWidthInSamplesL[ i ][ j ]>> 1 ) ) / curScaledRefLayerPicWidthInSamplesL[ i ][ j ] (F‑20)

scaleFactorY[ i ][ j ] = ( ( refCroppedPicHeightInSamplesL[ i ][ j ] << 16 ) +   
( curScaledRefLayerPicHeightInSamplesL >> 1 ) ) / curScaledRefLayerPicHeightInSamplesL[ i ][ j ] (F‑21)

[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[ 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‑22)

yCol[ i ][ j ] = Clip3( 0 , ( refPicHeightInSamplesL[ i ][ j ] − 1 ), ( ( yP − curScaledRefLayerTopOffset[ i ][ j ]) \* scaleFactorY[ i ][ j ] + ( 1 << 15 ) ) >> 16) ) (F‑23)

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

xColCtb[ i ][ j ] = xCol[ i ][ j ]  >>  refCtbLog2SizeY[ i ][ j ] (F‑24)

yColCtb[ i ][ j ] = yCol[ i ][ j ]  >>  refCtbLog2SizeY[ i ][ j ] (F‑25)

colCtbAddr[ i ][ j ] = xColCtb[ i ][ j ] + ( yColCtb[ i ][ j ] \* refPicWidthInCtbsY[ i ][ j ] ) (F‑26)

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‑27)

yOffset[ i ][ j ] = ( min\_spatial\_segment\_offset\_plus1[ i ][ j ] − 1 ) \* refPicWidthInCtbsY[ i ][ j ] (F‑28)

refCtbAddr[ i ][ j ] = colCtbAddr[ i ][ j ] + xOffset[ i ][ j ] + yOffset[ i ][ j ] (F‑29)

* 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.

**base\_layer\_parameter\_set\_compatibility\_flag**[ i ] equal to 1 specifies that the following constraints apply to the layer with nuh\_layer\_id equal to layer\_id\_in\_nuh[ i ]. base\_layer\_parameter\_set\_compatibility\_flag[ i ] equal to 0 specifies that the following constraints may or may not apply to the layer with nuh\_layer\_id equal to layer\_id\_in\_nuh[ i ].

* Each coded slice segment NAL unit with nuh\_layer\_id value equal to layer\_id\_in\_nuh[ i ] referring to the VPS shall refer to a PPS with nuh\_layer\_id value equal to 0.
* Each coded slice segment NAL unit with nuh\_layer\_id value equal to layer\_id\_in\_nuh[ i ] referring to the VPS shall refer to a SPS with nuh\_layer\_id value 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, respectively, of the active SPS for the layer with nuh\_layer\_id equal to layer\_id\_in\_nuh[ i ] shall be the same as 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, respectively, of the vps\_rep\_format\_idx[ i ]-th rep\_format( ) syntax structure in the active VPS.

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. [Ed. (MH): Add the allowed value range for this syntax element.]

**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. [Ed. (MH): Add the allowed value range for this syntax element.]

**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 the following constraints apply:

* The 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.
* When vps\_base\_layer\_internal\_flag is equal to 0 and layer\_in\_bsp\_flag[ h ][ i ][ 0 ] is equal to 1 for any value of h in the range of 1 to vps\_num\_layer\_sets\_minus1, inclusive, and any value of i in the range of 0 to num\_bitstream\_partitions[ h ] − 1, inclusive, the value of layer\_in\_bsp\_flag[ h ][ i ][ j ] for at least one value of j in the range of 1 to NumLayersInIdList[ h ] − 1, inclusive, shall be equal to 1.

[Ed. (GT): The following item corresponds to items 5/6 in Q0101 and might, according to meeting notes, require further alignment for the case that the base layer is externally specified.]

* When num\_bitstream\_partitions[ h ] is equal to 1 for any value of h in the range 1 to vps\_num\_layer\_set\_minus1, inclusive, the value of layer\_in\_bsp\_flag[ h ][ 0 ][ j ] should be equal to 0 for at least one value of j in the range 0 to NumLayersInIdList[ h ] − 1, inclusive. [Ed. (MH): Supposedly "should" in this sentence is meant to be "shall", as otherwise the sentence would not specify a constraint.] [Ed. (MH): It should be considered whether a constraint that "num\_bitstream\_partitions[ h ] shall not be equal to 1" would be better.]
* For any value of h in the range 1 to vps\_num\_layer\_set\_minus1, inclusive, the value of layer\_in\_bsp\_flag[ h ][ i ][ j ] shall be equal to 1 for at most one value of i in the range of 0 to num\_bitstream\_partitions[ h ] − 1, inclusive. [Ed. (MH): I think "at most" should be replaced by "exactly", because a partitioning specifies the mapping of each layer to a bitstream partition.]

**num\_bsp\_sched\_combinations\_minus1**[ h ] plus 1 specifies the number of combinations of delivery schedules and hrd\_parameters( ) specified for bitstream partitions for the layer set with index h. [Ed. (MH): Add the allowed value range for this syntax element.]

The variable SchedCombCnt[ h ] is set equal to num\_bsp\_sched\_combinations\_minus1[ h ] + 1.

**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. The length of the bsp\_comb\_hrd\_idx[ h ][ i ][ j ] syntax element is Ceil( Log2( vps\_num\_bsp\_hrd\_parameters\_minus1 + 1 ) ) bits. The value of bsp\_comb\_hrd\_idx[ h ][ i ][ j ] shall be in the range of 0 to vps\_num\_bsp\_hrd\_parameters\_minus1, inclusive.

**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. The value of bsp\_comb\_sched\_idx[ h ][ i ][ j ] 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 from the hrd\_parameters( ) syntax structure corresponding to the index bsp\_comb\_hrd\_idx[ h ][ i ][ j ]. [Ed. (YK): Both forms of "sub\_layer\_hrd\_parameters( HighestTid )" and "sub\_layer\_hrd\_parameters( )" are used in the document for referencing of the syntax structure. Check whether it would be better to consistently use just one of them.]

* + - * 1. Sequence parameter set RBSP semantics

The specifications in subclause 7.4.3.2 apply, with following additions and modifications.

**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.

**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, the value of sps\_temporal\_id\_nesting\_flag is inferred as follows:

– If sps\_max\_sub\_layers\_minus1 is greater than 0, the value of sps\_temporal\_id\_nesting\_flag is inferred to be equal to vps\_temporal\_id\_nesting\_flag.

– Otherwise, the value of sps\_temporal\_id\_nesting\_flag is inferred to be equal to 1.

NOTE 1 – 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.

**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 the value of vps\_num\_rep\_formats\_minus1 in the active VPS is equal to 0, it is a requirement of bitstream conformance that the value of update\_rep\_format\_flag shall 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‑30)

QpBdOffsetY = 6 \* bit\_depth\_luma\_minus8 (F‑31)

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‑32)

QpBdOffsetC = 6 \* bit\_depth\_chroma\_minus8 (F‑33)

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\_max\_dec\_pic\_buffering\_minus1**[ i ] plus 1 specifies the maximum required size of the decoded picture buffer for the CVS in units of picture storage buffers when HighestTid is equal to i. The value of sps\_max\_dec\_pic\_buffering\_minus1[ i ] shall be in the range of 0 to MaxDpbSize − 1 (as specified in subclause A.4), inclusive. When i is greater than 0, sps\_max\_dec\_pic\_buffering\_minus1[ i ] shall be greater than or equal to sps\_max\_dec\_pic\_buffering\_minus1[ i − 1 ]. The value of sps\_max\_dec\_pic\_buffering\_minus1[ i ] shall be less than or equal to vps\_max\_dec\_pic\_buffering\_minus1[ i ] for each value of i. When sps\_max\_dec\_pic\_buffering\_minus1[ i ] is not present for i in the range of 0 to sps\_max\_sub\_layers\_minus1 − 1, inclusive, due to sps\_sub\_layer\_ordering\_info\_present\_flag being equal to 0, it is inferred to be equal to sps\_max\_dec\_pic\_buffering\_minus1[ sps\_max\_sub\_layers\_minus1 ].

When sps\_max\_dec\_pic\_buffering\_minus1[ i ] is not present for i in the range of 0 to sps\_max\_sub\_layers\_minus1, inclusive, due to nuh\_layer\_id being greater than 0, for a layer that refers to the SPS and has nuh\_layer\_id equal to currLayerId, the value of sps\_max\_dec\_pic\_buffering\_minus1[ i ] is inferred to be equal to max\_vps\_dec\_pic\_buffering\_minus1[ TargetOlsIdx ][ layerIdx ][ i ] of the active VPS, where layerIdx is equal to the value such that LayerSetLayerIdList[ TargetDecLayerSetIdx ][ layerIdx ] is equal to currLayerId.

**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 vps\_base\_layer\_internal\_flag is equal to 0, it is a requirement of bitstream conformance that the value of sps\_scaling\_list\_ref\_layer\_id, when present, shall be greater than 0. [Ed. (JB): Should an inference value be added when not present?]

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\_extension\_present\_flag** equal to 1 specifies that the syntax elements sps\_range\_extensions\_flag, sps\_multilayer\_extension\_flag, and sps\_extension\_6bits are present in the SPS RBSP syntax structure. sps\_extension\_present\_flag equal to 0 specifies that these syntax elements are not present.

**sps\_range\_extensions\_flag** equal to 1 specifies that the sps\_range\_extensions( ) syntax structure is present in the SPS RBSP syntax structure. sps\_range\_extensions\_flag equal to 0 specifies that the sps\_range\_extensions( ) syntax structure is not present. When not present, the value of sps\_range\_extensions\_flag is inferred to be equal to 0.

**sps\_multilayer\_extension\_flag** equal to 1 specifies that the sps\_multilayer\_extension( ) syntax structure is present in the SPS RBSP syntax structure. sps\_multilayer\_extension\_flag equal to 0 specifies that the sps\_multilayer\_extension( ) syntax structure is not present. When not present, the value of sps\_multilayer\_extension\_flag is inferred to be equal to 0.

**sps\_extension\_6bits** equal to 0 specifies that no sps\_extension\_data\_flag syntax elements are present in the SPS RBSP syntax structure. When present, sps\_extension\_6bits shall be equal to 0 in bitstreams conforming to this version of this Specification. Values of sps\_extension\_6bits not equal to 0 are reserved for future use by ITU-T | ISO/IEC. Decoders shall allow the value of sps\_extension\_6bits to be not equal to 0 and shall ignore all sps\_extension\_data\_flag syntax elements in an SPS NAL unit. When not present, the value of sps\_extension\_6bits is inferred to be equal to 0.

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.

**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.

**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.

**vert\_phase\_position\_enable\_flag**[ scaled\_ref\_layer\_id[ i ] ] equal to 1 specifies that the syntax vert\_phase\_position\_flag for the associated inter-layer picture with nuh\_layer\_id equal to scaled\_ref\_layer\_id[ i ] is present in each slice segment header referring to the SPS. vert\_phase\_position\_enable\_flag[ scaled\_ref\_layer\_id[ i ] ] equal to 0 specifies that the syntax ver\_phase\_position\_flag for the associated inter-layer picture with nuh\_layer\_id equal to scaled\_ref\_layer\_id[ i ] is not present in each slice segment header referring to the SPS. When not present, the value of vert\_phase\_position\_enable\_flag[ 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:

**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 vps\_base\_layer\_internal\_flag is equal to 0, it is a requirement of bitstream conformance that pps\_scaling\_list\_ref\_layer\_id, when present, shall be greater than 0. [Ed. (JB): Should an inference value be added when not present?]

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 specified in subclause 7.4.5.

**pps\_extension\_present\_flag** equal to 1 specifies that the syntax elements pps\_range\_extensions\_flag, pps\_multilayer\_extension\_flag, and pps\_extension\_6bits, are present in the PPS RBSP syntax structure. pps\_extension\_present\_flag equal to 0 specifies that these syntax elements are not present.

**pps\_range\_extensions\_flag** equal to 1 specifies that the pps\_range\_extensions( ) syntax structure is present in the PPS RBSP syntax structure. pps\_range\_extensions\_flag equal to 0 specifies that the pps\_range\_extensions( ) syntax structure is not present. When not present, the value of pps\_range\_extensions\_flag is inferred to be equal to 0.

**pps\_multilayer\_extension\_flag** equal to 1 specifies that the poc\_reset\_info\_present\_flag and pps\_extension\_reserved\_zero\_flag syntax elements are present in the PPS RBSP syntax structure. pps\_multilayer\_extension\_flag equal to 0 specifies that the poc\_reset\_info\_present\_flag and pps\_extension\_reserved\_zero\_flag syntax elements are not present. When not present, the value of pps\_multilayer\_extension\_flag is inferred to be equal to 0.

**pps\_extension\_6bits** equal to 0 specifies that no pps\_extension\_data\_flag syntax elements are present in the PPS RBSP syntax structure. When present, pps\_extension\_6bits shall be equal to 0 in bitstreams conforming to this version of this Specification. Values of pps\_extension\_6bits not equal to 0 are reserved for future use by ITU-T | ISO/IEC. Decoders shall allow the value of pps\_extension\_6bits to be not equal to 0 and shall ignore all pps\_extension\_data\_flag syntax elements in a PPS NAL unit. When not present, the value of pps\_extension\_6bits is inferred to be equal to 0.

**poc\_reset\_info\_present\_flag** equal to 0 specifies that the syntax element poc\_reset\_idc is not present in the slice segment headers of the slices referring to the PPS. poc\_reset\_info\_present\_flag equal to 1 specifies that the syntax element poc\_reset\_idc is present in the slice segment headers of the slices referring to the PPS. When not present, the value of poc\_reset\_info\_present\_flag is inferred to be equal to 0.

**colour\_mapping\_enabled\_flag** equal to 1 specifies the colour mapping process may be applied to the inter-layer reference pictures for the coded pictures referring to the PPS. colour\_mapping\_enabled\_flag equal to 0 specifies that the colour mapping process is not applied to the inter-layer reference pictures for the coded pictures referring to the PPS. When not present, colour\_mapping\_enabled\_flag is inferred to be equal to 0.

It is a requirement of bitstream conformance that, when pps\_pic\_parameter\_set\_id is greater than or equal to 8, colour\_mapping\_enabled\_flag shall be equal to 0.

* + - * 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

If vps\_base\_layer\_internal\_flag is equal to 0 and the profile\_tier\_level( ) syntax structure is the first profile\_tier\_level( ) syntax structure in the VPS, all bits in the syntax structure shall be equal to 0 and decoders shall ignore the syntax structure. Otherwise, the semantics of the profile\_tier\_level( ) syntax structure are specified by the remaining part of the current subclause.

The profile\_tier\_level( ) syntax structure provides profile, tier and level to which an OLS conforms. When the profile\_tier\_level( ) syntax structure is included in a vps\_extension( ) syntax structure, the profile\_tier\_level\_idx[ i ] syntax element of the vps\_extension( ) syntax structure specifies which profile\_tier\_level( ) syntax structure applies to the i-th OLS. When num\_add\_layer\_sets is greater than 0 and i is in the range of FirstAddLayerSetIdx to LastAddLayerSetIdx, inclusive, the profile\_tier\_level( ) syntax structure identified by profile\_tier\_level\_idx[ i ] applies to the output of the non-base layer subtree extraction process of subclause F.10.2 with the input variable lsIdx set equal to OlsIdxToLsIdx[ i ]. When the profile\_tier\_level( ) syntax structure is included in a VPS, but not in a vps\_extension( ) syntax structure, it applies to the 0-th OLS. When the profile\_tier\_level( ) syntax structure is included in an active SPS for the base layer, it applies to the 0-th OLS. When the profile\_tier\_level( ) syntax structure is included in an active SPS for an independent non-base layer with nuh\_layer\_id equal to layerId, it applies to the 0-th OLS in an output bitstream of the non-base layer subtree extraction process of subclause F.10.2 with an input parameter lsIdx such that AssignedBaseLayerId[ lsIdx ] is equal to layerId.

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, they are inferred to be equal to the corresponding values of the (ptlIdx − 1 )-th profile\_tier\_level( ) syntax structure in the VPS extension.

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, they are inferred to be equal to the corresponding values of (ptlIdx − 1 )-th profile\_tier\_level( ) syntax structure in the VPS extension.

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 subclause G.11 or 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 subclause G.11 or 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 subclause G.11 or subclause H.11.

A sequence of pictures picSeq is derived as follows:

– If the profile\_tier\_level( ) syntax structure is included in an SPS, picSeq consists of the pictures in the CVS for which the SPS is the active SPS.

– Otherwise, if the profile\_tier\_level( ) syntax structure is included in a VPS, but not in the vps\_extension( ) syntax structure, picSeq consists of the pictures with nuh\_layer\_id equal to 0 in the CVS.

– Otherwise, if the profile\_tier\_level( ) syntax structure is associated with an OLS with alt\_output\_layer\_flag[ i ] equal to 1, picSeq consists of the pictures of the output layer and its direct and indirect reference layers within the CVS.

– Otherwise, picSeq consists of the pictures of the output layers of the associated OLS within the CVS.

**general\_progressive\_source\_flag** and **general\_interlaced\_source\_flag** are interpreted as follows:

– If general\_progressive\_source\_flag is equal to 1 and general interlaced\_source\_flag is equal to 0, the source scan type of the pictures in the picSeq should be interpreted as progressive only.

– Otherwise, if general\_progressive\_source\_flag is equal to 0 and general\_interlaced\_source\_flag is equal to 1, the source scan type of the pictures in the picSeq should be interpreted as interlaced only.

– Otherwise, if general\_progressive\_source\_flag is equal to 0 and general\_interlaced\_source\_flag is equal to 0, the source scan type of the pictures in the picSeq should be interpreted as unknown or unspecified.

– Otherwise, general\_progressive\_source\_flag is equal to 1 and general\_interlaced\_source\_flag is equal to 1, the source scan type of each picture in the picSeq is indicated at the picture level using the syntax element source\_scan\_type in a picture timing SEI message or the syntax element ffinfo\_source\_scan\_type in a frame-field information SEI message.

NOTE 3 – Decoders may ignore the values of general\_progressive\_source\_flag and general\_interlaced\_source\_flag for purposes other than determining the value to be inferred for frame\_field\_info\_present\_flag when vui\_parameters\_present\_flag is equal to 0, as there are no other decoding process requirements associated with the values of these flags. Moreover, the actual source scan type of the pictures is outside the scope of this Specification, and the method by which the encoder selects the values of general\_progressive\_source\_flag and general\_interlaced\_source\_flag is unspecified.

**general\_non\_packed\_constraint\_flag** equal to 1 specifies that there are no frame packing arrangement SEI messages present for the pictures of picSeq. general non\_packed\_constraint\_flag equal to 0 indicates that there may or may not be one or more frame packing arrangement SEI messages present for the pictures of picSeq.

NOTE 4 – Decoders may ignore the value of general\_non\_packed\_constraint\_flag, as there are no decoding process requirements associated with the presence or interpretation of frame packing arrangement SEI messages.

**general\_frame\_only\_constraint\_flag** equal to 1 specifies that field\_seq\_flag in the active SPSs for the pictures of picSeq is equal to 0. general\_frame\_only\_constraint\_flag equal to 0 indicates that field\_seq\_flag in the active SPSs for the pictures of picSeq may or may not be equal to 0.

NOTE 5 – Decoders may ignore the value of general\_frame\_only\_constraint\_flag, as there are no decoding process requirements associated with the value of field\_seq\_flag.

NOTE 6 – When general\_progressive\_source\_flag is equal to 1, general\_frame\_only\_constraint\_flag may or may not be equal to 1.

**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 representation 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. Colour mapping table semantics
         1. General colour mapping table semantics

**num\_cm\_ref\_layers\_minus1** specifies the number of the syntax element cm\_ref\_layer\_id that are present in the PPS. When not present, the value of num\_cm\_ref\_layers\_minus1 is inferred to be equal to −1. The value of num\_cm\_ref\_layers\_minus1 shall be in the range of −1 to 61, inclusive.

**cm\_ref\_layer\_id**[ i ] specifies the nuh\_layer\_id value of the associated reference layer picture for which the colour mapping table is specified.

**cm\_octant\_depth** specifies the maximal split depth of the colour mapping table. In bitstreams conforming to this version of this Specification, the value of cm\_octant\_depth shall be in the range of 0 to 1, inclusive. Other values for cm\_octant\_depth are reserved for future use by ITU-T | ISO/IEC.

The variable COctantNum is derived as follows.

COctantNum = 1 << cm\_octant\_depth (F‑34)

**cm\_y\_part\_num\_log2** specifies the number of partitions of the smallest colour mapping table octant for the luma component. In bitstreams conforming to this version of this Specification, the value of ( cm\_y\_part\_num\_log2 + cm\_octant\_depth ) shall be in the range of 0 to 3, inclusive. Other value for ( cm\_y\_part\_num\_log2 + cm\_octant\_depth ) is reserved for future use by ITU-T | ISO/IEC.

The variables YOctantNum and YPartNum are derived as follows.

YOctantNum = 1 << ( cm\_octant\_depth + cm\_y\_part\_num\_log2 ) (F‑35)

YPartNum = 1 << cm\_y\_part\_num\_log2 (F‑36)

**cm\_input\_luma\_bit\_depth\_minus8** specifies the sample bit depth of the input luma component of the colour mapping process. The variable CMInputBitDepthY is derived as follows:

CMInputBitDepthY = 8 + cm\_input\_luma\_bit\_depth\_minus8 (F‑37)

It is a requirement of bitstream conformance that the value of CMInputBitDepthY shall be equal to the bit depth of the luma component of any layer with nuh\_layer\_id equal to one of the cm\_ref\_layer\_id[ i ], for all i in the range of 0 to num\_cm\_ref\_layers\_minus1, inclusive.

**cm\_input\_chroma\_bit\_depth\_minus8** specifies the sample bit depth of the input chroma components of the colour mapping process. The variable CMInputBitDepthC is derived as follows:

CMInputBitDepthC = 8 + cm\_input\_chroma\_bit\_depth\_minus8 (F‑38)

It is a requirement of bitstream conformance that the value of CMInputBitDepthC shall be equal to the bit depth of the chroma component of any layer with nuh\_layer\_id equal to one of the cm\_ref\_layer\_id[ i ], for all i in the range of 0 to num\_cm\_ref\_layers\_minus1, inclusive.

**cm\_output\_luma\_bit\_depth\_minus8** specifies the sample bit depth of the output luma component of the colour mapping process. The variable CMOutputBitDepthY is derived as follows:

CMOutputBitDepthY = 8 + cm\_output\_luma\_bit\_depth\_minus8 (F‑39)

It is a requirement of bitstream conformance that the value of CMOutputBitDepthY shall be greater than or equal to CMInputBitDepthY and less than or equal to the bit depth of the luma component of any picture that refers to this PPS,.

**cm\_output\_chroma\_bit\_depth\_minus8** specifies the sample bit depth of the output chroma components of the colour mapping process. The variable CMOutputBitDepthC is derived as follows:

CMOutputBitDepthC = 8 + cm\_output\_chroma\_bit\_depth\_minus8 (F‑40)

It is a requirement of bitstream conformance that the value of CMOutputBitDepthC shall be greater than or equal to CMInputBitDepthC and less than or equal to the bit depth of the chroma component of any picture that refers to this PPS.

**cm\_res\_quant\_bits** specifies the number of least significant bits to be added to the the colour mapping coefficient residual values res\_coeff\_q and res\_coeff\_r.

**cm\_delta\_flc\_bits\_minus1** specifies the delta value used to derive the number of bits used to code the syntax element res\_coeff\_r. The variable CMResLSBits is set equal to max( 0, ( 10 + CMInputBitDepthY − CMOutputBitDepthY − cm\_res\_quant\_bits − ( cm\_delta\_flc\_bits\_minus1 + 1 ) ) ).

**cm\_adapt\_threshold\_u\_delta** specifies the partitioning threshold of the Cb component used in colour mapping process. When not present, the value of cm\_adapt\_threshold\_u\_delta is inferred to be equal to 0. The value of cm\_adapt\_threshold\_u\_delta shall be in the range of −2CMInputBitDepthC − 1 to 2CMInputBitDepthC − 1 − 1, inclusive.

The variable CMThreshU is derived as follows:

CMThreshU = cm\_adapt\_threshold\_u\_delta + (1 << (CMInputBitDepthC − 1) )

**cm\_adapt\_threshold\_v\_delta** specifies the partitioning threshold of the Cr component used in colour mapping process. When not present, the value of cm\_adapt\_threshold\_v\_delta is inferred to be equal to 0. The value of cm\_adapt\_threshold\_v\_delta shall be in the range of −2CMInputBitDepthC − 1 to 2CMInputBitDepthC − 1 − 1, inclusive.

The variable CMThreshV is derived as follows:

CMThreshV = cm\_adapt\_threshold\_v\_delta + (1 << (CMInputBitDepthC − 1) )

* + - * 1. Colour mapping octants semantics

**split\_octant\_flag** equal to 1 specifies that the current colour mapping octant is further split into eight octants with half length in each of the three dimensions. split\_octant\_flag equal to 0 specifies that the current colour mapping octant is not further split into eight octants. When not present, the value of split\_octant\_flag is inferred to be equal to 0.

**coded\_res\_flag**[ yIdxShift ][ uIdx ][ vIdx ][ j ]equal to 1 specifies that the residuals for the colour mapping coefficients with index [ yIdxShift ][ uIdx ][ vIdx ][ j ] are present. coded\_res\_flag[ yIdxShift ][ uIdx ][ vIdx ][ j ] equal to 0 specifies that the residuals for the colour mapping coefficients with index [ yIdxShift ][ uIdx ][ vIdx ][ j ] are not present. When not present, the value of coded\_res\_flag[ yIdxShift ][ uIdx ][ vIdx ][ j ] is inferred to be equal to 0.

**res\_coeff\_q**[ yIdxShift ][ uIdx ][ vIdx ][ j ][ c ]specifies the quotient of the residual for the colour mapping coefficient with index[ yIdxShift ][ uIdx ][ vIdx ][ j ][ c ]. When not present, the value of res\_coeff\_q[ yIdxShift ][ uIdx ][ vIdx ][ j ][ c ] is inferred to be equal to 0.

**res\_coeff\_r**[ yIdxShift ][ uIdx ][ vIdx ][ j ][ c ]specifies the remainder of the residual for the colour mapping coefficient with index[ yIdxShift ][ uIdx ][ vIdx ][ j ][ c ]. The number of bits used to code res\_coeff\_r is equal to CMResLSBits. If CMResLSBits is equal to 0, res\_coeff\_r is not present. When not present, the value of res\_coeff\_r[ yIdxShift ][ uIdx ][ vIdx ][ j ][ c ] is inferred to be equal to 0.

**res\_coeff\_s**[ yIdxShift ][ uIdx ][ vIdx ][ j ][ c ] specifies the sign of the residual for the colour mapping coefficient with index[ yIdxShift ][ uIdx ][ vIdx ][ j ][ c ]. When not present, the value of res\_coeff\_s[ yIdxShift ][ uIdx ][ vIdx ][ j ][ c ] is inferred to be equal to 0.

The variables cmResCoeff[ yIdxShift ][ uIdx ][ vIdx ][ j ][ c ] is derived as follows:

cmResCoeff[ yIdxShift ][ uIdx ][ vIdx ][ j ][ c ] =   
 ( 1 − 2 \* res\_coeff\_s[ yIdxShift ][ uIdx ][ vIdx ][ j ][ c ] ) \*   
 ( ( ( res\_coeff\_q[ yIdxShift ][ uIdx ][ vIdx ][ j ][ c ] << CMResLSBits ) +  
 res\_coeff\_r[ yIdxShift ][ uIdx ][ vIdx ][ j ][ c ] ) << cm\_res\_quant\_bits ) (F‑XX)

The colour mapping coeffients LutY [ yIdx ][ uIdx ][ vIdx ][ i ], LutU [ yIdx ][ uIdx ][ vIdx ][ i ], LutV [ yIdx ][ uIdx ][ vIdx ][ i ] with yIdx in the range of 0 and YOctantNum − 1, inclusive, uIdx in the range of 0 and COctantNum − 1, inclusive, vIdx in the range of 0 and COctantNum − 1, inclusive, and i in the range of 0 and 3, inclusive, are derived as follows:

if( yIdx = = 0 ) {  
 predY[ yIdx ][ uIdx ][ vIdx ][ i ] = ( i = = 0 ) ? 1024 : 0  
 predU[ yIdx ][ uIdx ][ vIdx ][ i ] = ( i = = 1 ) ? 1024 : 0  
 predV[ yIdx ][ uIdx ][ vIdx ][ i ] = ( i = = 2 ) ? 1024 : 0  
}  
else {  
 predY[ yIdx ][ uIdx ][ vIdx ][ i ] = LutY[ yIdx − 1 ][ uIdx ][ vIdx ][ i ]  
 predU[ yIdx ][ uIdx ][ vIdx ][ i ] = LutU[ yIdx − 1 ][ uIdx ][ vIdx ][ i ]  
 predV[ yIdx ][ uIdx ][ vIdx ][ i ] = LutV[ yIdx − 1 ][ uIdx ][ vIdx ][ i ] (F‑XX)  
}  
LutY[ yIdx ][ uIdx ][ vIdx ][ i ] = ( cmResCoeff [ yIdx ][ uIdx ][ vIdx ][ i ][ 0 ] )   
+ predY[ yIdx ][ uIdx ][ vIdx ][ i ]   
LutU[ yIdx ][ uIdx ][ vIdx ][ i ] = ( cmResCoeff [ yIdx ][ uIdx ][ vIdx ][ i ][ 1 ] )   
+ predU[ yIdx ][ uIdx ][ vIdx ][ i ]   
LutV[ yIdx ][ uIdx ][ vIdx ][ i ] = ( cmResCoeff [ yIdx ][ uIdx ][ vIdx ][ i ][ 2 ] )   
+ predV[ yIdx ][ uIdx ][ vIdx ][ i ]

It is a requirement of bitstream conformance that the values of LutY[ yIdx ][ uIdx ][ vIdx ][ i ], LutU[ yIdx ][ uIdx ][ vIdx ][ i ], and LutV[ yIdx ][ uIdx ][ vIdx ][ i ] shall be in the range of −211 to 211 − 1, inclusively.

* + - 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, inter\_layer\_pred\_enabled\_flag, num\_inter\_layer\_ref\_pics\_minus1, poc\_reset\_idc, poc\_reset\_period\_id, full\_poc\_reset\_flag, poc\_lsb\_val, and poc\_msb\_val 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 ], and vert\_phase\_position\_flag[ i ] shall be the same in all slice segment headers of a coded picture for each possible value of i.

When vps\_poc\_lsb\_aligned\_flag is equal to 1, slice\_pic\_order\_cnt\_lsb shall be the same in all slice segment headers of all coded pictures of the same access unit.

– "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.

When nal\_unit\_type is equal to TRAIL\_R, TSA\_R, STSA\_R, RASL\_R or RADL\_R, the value of discardable\_flag shall be equal to 0.

**cross\_layer\_bla\_flag** equal to 1 affects the derivation of NoClrasOutputFlag as specified in subclause 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.

**num\_long\_term\_sps** specifies the number of entries in the long-term RPS of the current picture that are derived based on the candidate long-term reference pictures specified in the active SPS. The value of num\_long\_term\_sps shall be in the range of 0 to num\_long\_term\_ref\_pics\_sps, inclusive. When not present, the value of num\_long\_term\_sps is inferred to be equal to 0.

**num\_long\_term\_pics** specifies the number of entries in the long-term RPS of the current picture that are directly signalled in the slice header. When not present, the value of num\_long\_term\_pics is inferred to be equal to 0.

The variable maxNumPics is derived as follows:

maxNumPics = MaxDpbSize ‑ 1  
for( olsIdx = 0; olsIdx < NumOutputLayerSets; olsIdx++) {  
 lsIdx = OlsIdxToLsIdx[ olsIdx ]  
 for( j = 0; j < NumLayersInIdList[ lsIdx ]; j++ )  
 if( LayerSetLayerIdList[ lsIdx ][ j ] = = nuh\_layer\_id ) {  
 maxSL = MaxSubLayersInLayerSetMinus1[ lsIdx ]  
 maxNumPics = Min( maxNumPics, max\_vps\_dec\_pic\_buffering\_minus1[ olsIdx ][ j ][ maxSL ] )  
 }  
}

When nuh\_layer\_id is equal to 0, the sum of NumNegativePics[ CurrRpsIdx ], NumPositivePics[ CurrRpsIdx ], num\_long\_term\_sps, and num\_long\_term\_pics shall be less than or equal to sps\_max\_dec\_pic\_buffering\_minus1[ sps\_max\_sub\_layers\_minus1 ]. When vps\_extension\_flag is equal to 1, the sum of NumNegativePics[ CurrRpsIdx ], NumPositivePics[ CurrRpsIdx ], num\_long\_term\_sps, and num\_long\_term\_pics shall be less than or equal to maxNumPics.

**slice\_temporal\_mvp\_enabled\_flag** specifies whether temporal motion vector predictors can be used for inter prediction. If slice\_temporal\_mvp\_enabled\_flag is equal to 0, the syntax elements of the current picture shall be constrained such that no temporal motion vector predictor is used in decoding of the current picture. Otherwise (slice\_temporal\_mvp\_enabled\_flag is equal to 1), temporal motion vector predictors may be used in decoding of the current picture. When not present, the value of slice\_temporal\_mvp\_enabled\_flag is inferred to be equal to 0.

Let currLayerId be equal to nuh\_layer\_id of the current NAL unit. When NumDirectRefLayers[ currLayerId ] is equal to 0, slice\_temporal\_mvp\_enabled\_flag is equal to 0, and TemporalId is equal to 0, the syntax elements for all coded pictures with nuh\_layer\_id equal to currLayerId that follow the current picture in decoding order shall be constrained such that no temporal motion vector from any picture that precedes the current picture in decoding order is used in decoding of any coded picture with nuh\_layer\_id equal to currLayerId that follows the current picture in decoding order.

NOTE 7 – When NumDirectRefLayers[ currLayerId ] is equal to 0 and slice\_temporal\_mvp\_enabled\_flag is equal to 0 in an I slice, it has no impact on the normative decoding process of the picture but merely expresses a bitstream constraint.

NOTE 8 – When NumDirectRefLayers[ currLayerId ] is equal to 0 and slice\_temporal\_mvp\_enabled\_flag is equal to 0 in a slice with TemporalId equal to 0, decoders may empty "motion vector storage" for all reference pictures with nuh\_layer\_id equal to currLayerId in the decoded picture buffer.

**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 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 refLayerPicIdc[ j ] are derived as follows:

for( i = 0, j = 0; i < NumDirectRefLayers[ nuh\_layer\_id ]; i++ ) {  
 refLayerIdx = LayerIdxInVps[ RefLayerId[ nuh\_layer\_id ][ i ] ]  
 if( sub\_layers\_vps\_max\_minus1[ refLayerIdx ] >= TemporalId && ( TemporalId = = 0 | |   
 max\_tid\_il\_ref\_pics\_plus1[ refLayerIdx ][ LayerIdxInVps[ nuh\_layer\_id ] ] > TemporalId ) )  
 refLayerPicIdc[ j++ ] = i  
}

numRefLayerPics = j

The variable NumActiveRefLayerPics is derived as follows:

if( nuh\_layer\_id = = 0 | | numRefLayerPics = = 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 = 1  
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.

**vert\_phase\_position\_flag[** RefPicLayerId[ i ] **]** specifies the phase position in the vertical direction used to derive reference layer sample location when the reference layer picture with nuh\_layer\_id equal to RefPicLayerId[ i ] is resampled. When not present, the value of phase\_position\_flag[ RefPicLayerId[ i ] ] is inferred to be equal to 0.

**poc\_reset\_idc** equal to 0 specifies that neither the most significant bits nor the least significant bits of the picture order count value for the current picture are reset. poc\_reset\_idc equal to 1 specifies that only the most significant bits of the picture order count value for the current picture may be reset. poc\_reset\_idc equal to 2 specifies that both the most significant bits and the least significant bits of the picture order count value for the current picture may be reset. poc\_reset\_idc equal to 3 specifies that either only the most significant bits or both the most significant bits and the least significant bits of the picture order count value for the current picture may be reset and additional picture order count information is signalled. When not present, the value of poc\_reset\_idc is inferred to be equal to 0.

It is a requirement of bitstream conformance that the following constraints apply:

– The value of poc\_reset\_idc shall not be equal to 1 or 2 for a RASL picture, a RADL picture, a sub-layer non-reference picture, or a picture that has TemporalId greater than 0, or a picture that has discardable\_flag equal to 1.

– The value of poc\_reset\_idc of all pictures in an access unit shall be the same.

– When the picture in an access unit with nuh\_layer\_id equal to 0 is an IRAP picture with a particular value of nal\_unit\_type and there is at least one other picture in the same access unit with a different value of nal\_unit\_type, the value of poc\_reset\_idc shall be equal to 1 or 2 for all pictures in the access unit.

– When there is at least one picture that has nuh\_layer\_id greater than 0 and that is an IDR picture with a particular value of nal\_unit\_type in an access unit and there is at least one other picture in the same access unit with a different value of nal\_unit\_type, the value of poc\_reset\_idc shall be equal to 1 or 2 for all pictures in the access unit.

– The value of poc\_reset\_idc of a CRA or BLA picture shall less than 3.

– When the picture with nuh\_layer\_id equal to 0 in an access unit is an IDR picture and there is at least one non-IDR picture in the same access unit, the value of poc\_reset\_idc shall be equal to 2 for all pictures in the access unit.

– When the picture with nuh\_layer\_id equal to 0 in an access unit is not an IDR picture, the value of poc\_reset\_idc shall not be equal to 2 for any picture in the access unit.

The value of poc\_reset\_idc of an access unit is the value of poc\_reset\_idc of the pictures in the access unit.

**poc\_reset\_period\_id** identifies a POC resetting period. There shall be no two pictures consecutive in decoding order in the same layer that have the same value of poc\_reset\_period\_id and poc\_reset\_idc equal to 1 or 2. When not present, the value of poc\_reset\_period\_id is inferred as follows:

– If the previous picture picA that has poc\_reset\_period\_id present in the slice segment header in present in the same layer of the bitstream as the current picture, the value of poc\_reset\_period\_id is inferred to be equal to the value of the poc\_reset\_period\_id of picA.

– Otherwise, the value of poc\_reset\_period\_id is inferred to be equal to 0.

NOTE – It is not prohibited for multiple pictures in a layer to have the same value of poc\_reset\_period\_id and to have poc\_reset\_idc equal to 1 or 2 unless such pictures occur in two consecutive access units in decoding order. To minimize the likelihood of such two pictures appearing in the bitstream due to picture losses, bitstream extraction, seeking, or splicing operations, encoders should set the value of poc\_reset\_period\_id to be a random value for each POC resetting period (subject to the constraints specified above).

It is a requirement of bitstream conformance that the following constraints apply:

– One POC resetting period shall not include more than one access unit with poc\_reset\_idc equal to 1 or 2.

– An access unit with poc\_reset\_idc equal to 1 or 2 shall be the first access unit in a POC resetting period.

– A picture that follows, in decoding order, the first POC resetting picture among all layers of a POC resetting period in decoding order shall not precede, in output order, another picture in any layer that precedes the first POC resetting picture in decoding order.

**full\_poc\_reset\_flag** equal to 1 specifies that both the most significant bits and the least significant bits of the picture order count value for the current picture are reset when the previous picture in decoding order in the same layer does not belong to the same POC resetting period. full\_poc\_reset\_flag equal to 0 specifies that only the most significant bits of the picture order count value for the current picture are reset when the previous picture in decoding order in the same layer does not belong to the same POC resetting period.

**poc\_lsb\_val** specifies a value that may be used to derive the picture order count of the current picture. The length of the poc\_lsb\_val syntax element is log2\_max\_pic\_order\_cnt\_lsb\_minus4 + 4 bits.

It is a requirement of bitstream conformance that, when poc\_reset\_idc is equal to 3, and the previous picture picA in decoding order that is in the same layer as the current picture, that has poc\_reset\_idc equal to 1 or 2, and that belongs to the same POC resetting period is present in the bitstream, picA shall be the same picture as the previous picture in decoding order that is in the same layer as the current picture, that is not a RASL picture, a RADL picture or a sub-layer non-reference picture, and that has TemporalId equal to 0 and discardable\_flag equal to 0, and the value of poc\_lsb\_val of the current picture shall be equal to the value of slice\_pic\_order\_cnt\_lsb of picA.

The variable PocMsbValRequiredFlag is derived as follows:

PocMsbValRequiredFlag = CraOrBlaPicFlag && ( !vps\_poc\_lsb\_aligned\_flag | | (F‑41)  
 ( vps\_poc\_lsb\_aligned\_flag && NumDirectRefLayers[ nuh\_layer\_id ] = = 0 ) )

**poc\_msb\_val\_present\_flag** equal to 1 specifies that poc\_msb\_val is present. When poc\_msb\_val\_present\_flag is equal to 0, poc\_msb\_val is not present. When not present, the value of poc\_msb\_val\_present\_flag is inferred as follows:

– If slice\_segment\_header\_extension\_length is equal to 0, the value of poc\_msb\_val\_present\_flag is inferred to be equal to 0.

– Otherwise, if PocMsbValRequiredFlag is equal to 1, the value of poc\_msb\_val\_present\_flag is inferred to be equal to 1.

– Otherwise, the value of poc\_msb\_val\_present\_flag is inferred to be equal to 0.

**poc\_msb\_val** specifies the value of the most significant bits of the picture order count value of the current picture. The value of poc\_msb\_val may also be used to derive the value used to decrement the picture order count values of previously decoded pictures in the same layer as the current picture. The value of poc\_msb\_val shall be in the range of 0 to 232 − log2\_max\_pic\_order\_cnt\_lsb\_minus4 − 4, inclusive. The value of poc\_msb\_val shall be equal to the difference between the values of the most significant bits of the picture order counts of the current picture and the previous POC resetting picture in the same layer or the previous IDR picture in the same layer, whichever is closer, in decoding order, to the current picture. If neither picture is present, the value of poc\_msb\_val can be any value in the allowed range.

**slice\_segment\_header\_extension\_data\_bit** may have any value. Decoders shall ignore the value of slice\_segment\_header\_extension\_data\_bit. Its value does not affect decoder conformance to profiles specified in this version of this Specification.

* + - * 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‑42)  
 if(UsedByCurrPicS1[ CurrRpsIdx ][ i ] )  
 NumPicTotalCurr++  
for( i = 0; i < num\_long\_term\_sps + num\_long\_term\_pics; i++ )  
 if( UsedByCurrPicLt[ i ] )  
 NumPicTotalCurr++  
NumPicTotalCurr += NumActiveRefLayerPics

* + - * 1. Weighted prediction parameters semantics

The specifications in subclause 7.4.7.3 apply.

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

The specifications in subclause 7.4.8 apply, with the following modifications:

The variable maxNumPics is derived as follows:

maxNumPics = MaxDpbSize ‑ 1  
for( olsIdx = 0; olsIdx < NumOutputLayerSets; olsIdx++) {  
 lsIdx = OlsIdxToLsIdx[ olsIdx ]  
 for( j = 0; j < NumLayersInIdList[ lsIdx ]; j++ )  
 if( LayerSetLayerIdList[ lsIdx ][ j ] = = nuh\_layer\_id ) {  
 maxSL = MaxSubLayersInLayerSetMinus1[ lsIdx ]  
 maxNumPics = Min( maxNumPics, max\_vps\_dec\_pic\_buffering\_minus1[ olsIdx ][ j ][ maxSL ] )  
 }  
}

**num\_negative\_pics** specifies the number of entries in the stRpsIdx-th candidate short-term RPS that have picture order count values less than the picture order count value of the current picture. When nuh\_layer\_id is equal to 0, the value of num\_negative\_pics shall be in the range of 0 to sps\_max\_dec\_pic\_buffering\_minus1[ sps\_max\_sub\_layers\_minus1 ], inclusive. When vps\_extension\_flag is equal to 1, the value of num\_negative\_pics shall be in the range of 0 to maxNumPics, inclusive.

**num\_positive\_pics** specifies the number of entries in the stRpsIdx-th candidate short-term RPS that have picture order count values greater than the picture order count value of the current picture. When nuh\_layer\_id is equal to 0, the value of num\_positive\_pics shall be in the range of 0 to sps\_max\_dec\_pic\_buffering\_minus1[ sps\_max\_sub\_layers\_minus1 ] − num\_negative\_pics, inclusive. When vps\_extension\_flag is equal to 1, the value of num\_positive\_pics shall be in the range of 0 to maxNumPics − num\_negative\_pics, inclusive.

[Ed. (AR): Currently derivation of maxNumPics repeated here and in slice segment header semantics.]

* + - 1. 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, and subclause 8.1.1 with subclauses F.7, and F.8.1.1, respectively.*

*– At the end of the subclause, add the following in this subclause:*

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.

– When vps\_base\_layer\_internal\_flag is equal to 0, the following applies: [Ed. (YK): Check other places to ensure correct handling of the base layer when this flag is equal to 0.] [Ed. (MH): It might be better to move this to subclause 8.1 to specify clearly that it applies before processing the first (internal) picture of an access unit.]

– There is no coded picture with nuh\_layer\_id equal to 0 in the bitstream.

– The size of the sub-DPB for the layer with nuh\_layer\_id equal to 0 is set equal to 1.

– The values of 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 for decoded pictures with nuh\_layer\_id equal to 0 are set equal to the values of 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, respectively, of the vps\_rep\_format\_idx[ 0 ]-th rep\_format( ) syntax structure in the active VPS.

– In addition to a list of decoded pictures, this process also outputs a flag BaseLayerOutputFlag, and, when BaseLayerOutputFlag is equal to 0 and AltOptLayerFlag[ TargetOptLayerSetIdx ] is equal to 1, a flag BaseLayerPicOutputFlag for each access unit.

NOTE – The BaseLayerOutputFlag and, when present, BaseLayerPicOutputFlag for each access unit, are to be sent by an external means to the base layer decoder for controlling the output of base layer decoded pictures.

The variables BaseLayerOutputFlag and BaseLayerPicOutputFlag are derived as specified in the following :

– The variable BaseLayerOutputFlag is set equal to ( TargetOptLayerIdList[ 0 ]  = =  0 ).

BaseLayerOutputFlag equal to 1 specifies that the base layer is an output layer. BaseLayerOutputFlag equal to 0 specifies that the base layer is not an output layer.

– When BaseLayerOutputFlag is equal to 0 and AltOptLayerFlag[ TargetOptLayerSetIdx ] is equal to 1, for each access unit, BaseLayerPicOutputFlag is derived as follows:

– If the base layer is a direct or indirect reference layer of the output layer, the access unit does not contain a picture at the output layer or contains a picture at the output layer that has PicOutputFlag equal to 0, and does not contain a picture at any other direct or indirect reference layer of the output layer, BaseLayerPicOutputFlag is set equal to 1.

– Otherwise, BaseLayerPicOutputFlag is set equal to 0.

BaseLayerPicOutputFlag equal to 1 for an access unit specifies that the base layer picture of the access unit is output. BaseLayerPicOutputFlag equal to 0 for an access unit specifies that the base layer picture of the access unit is not output.

– The variable LayerInitializedFlag[ 0 ] is set equal to 1 and the variable FirstPicInLayerDecodedFlag[ 0 ] is set equal to 1.

– For each access unit with TemporalId less than or equal to sub\_layers\_vps\_max\_minus1[ 0 ], a decoded picture with nuh\_layer\_id equal to 0 may be provided by external means. When not provided, no picture with nuh\_layer\_id equal to 0 is used for inter-layer prediction for the current access unit. When provided, the following applies:

– The following information of the picture with nuh\_layer\_id equal to 0 for the access unit is provided by external means:

– The decoded sample values (1 sample array SL if chroma\_format\_idc is equal to 0 or 3 sample arrays SL, SCb, and SCr otherwise)

– The value of the variable BlIrapPicFlag, and when BlIrapPicFlag is equal to 1, the value of nal\_unit\_type of the decoded picture

– BlIrapPicFlag equal to 1 specifies that the decoded picture is an IRAP picture. BlIrapPicFlag equal to 0 specifies that the decoded picture is a non-IRAP picture.

– The provided value of nal\_unit\_type of the decoded picture shall be equal to IDR\_W\_RADL, CRA\_NUT, or BLA\_W\_LP.

– nal\_unit\_type equal to IDR\_W\_RADL specifies that the decoded picture is an IDR picture.

– nal\_unit\_type equal to CRA\_NUT specifies that the decoded picture is a CRA picture.

– nal\_unit\_type equal to BLA\_W\_LP specifies that the decoded picture is a BLA picture.

– When BlIrapPicFlag of the picture with nuh\_layer\_id equal to 0 is equal to 1, the following applies for the decoded picture with nuh\_layer\_id equal to 0 for the access unit:

– The variable NoRaslOutputFlag is specified as follows:

– If nal\_unit\_type is IDR\_W\_RADL or BLA\_W\_LP, the variable NoRaslOutputFlag is set equal to 1.

– Otherwise, if the current access unit is the first access unit in the bitstream in decoding order, the variable NoRaslOutputFlag is set equal to 1.

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

– The variable NoClrasOutputFlag is specified as follows:

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

– Otherwise, if nal\_unit\_type is equal to CRA\_NUT and NoRaslOutputFlag is equal to 1, NoClrasOutputFlag is set equal to 1.

– Otherwise, if nal\_unit\_type is equal to BLA\_W\_LP, NoClrasOutputFlag is set equal to1.

– 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 1 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 following applies for the decoded picture with nuh\_layer\_id equal to 0 for the access unit:

– If the access unit has at least one picture with nuh\_layer\_id greater than 0, the TemporalId and PicOrderCntVal of the decoded picture with nuh\_layer\_id equal to 0 are set equal to the TemporalId and PicOrderCntVal, respectively, of any picture with nuh\_layer\_id greater than 0 in the access unit.

– Otherwise, the decoded picture with nuh\_layer\_id equal to 0 is discarded and the sub-DPB for the layer with nuh\_layer\_id equal to 0 is set to be empty. [Ed. (YK): Make sure that there is a restriction, indirect or direct, that requires the TemporalId to be equal to 0 when BlIrapPicFlag is equal to 1.]

– The decoded picture with nuh\_layer\_id equal to 0 is stored in the sub-DPB for the layer with nuh\_layer\_id equal to 0 and is marked as "used for long-term reference".

– When the access unit has at least one picture with nuh\_layer\_id greater than 0, after all pictures in the access unit are decoded, the sub-DPB for the layer with nuh\_layer\_id equal to 0 is set to be empty.

* + - 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.

– At the end of the subclause, add item 5 as follows:

5. When FirstPicInLayerDecodedFlag[ 0 ] is equal to 0, FirstPicInLayerDecodedFlag[ 0 ] is set equal to 1.

* + - 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.

– ~~When~~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. (MH & JC): The condition on ViewScalExtLayerFlag[ nuh\_layer\_id ] or DependencyId[ nuh\_layer\_id ] being equal to 1 has the consequence that no decoding process is invoked for an auxiliary picture layer with ViewOrderIdx equal to 0. In general, it should be clarified which decoding process is invoked for coded pictures of any layers, including coded pictures that have nuh\_layer\_id greater than 0 but ViewScalExtLayerFlag[ nuh\_layer\_id ] and DependencyId[ nuh\_layer\_id ] equal to 0.]

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

When FirstPicInLayerDecodedFlag[ nuh\_layer\_id ] is equal to 0, FirstPicInLayerDecodedFlag[ nuh\_layer\_id ] is set equal to 1.

* + - 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.

When the current picture is the first picture among all layers of a POC resetting period, the variable PocDecrementedInDPBFlag[ i ] is set equal to 0 for each value of i in the range of 0 to 62, inclusive.

The variable pocResettingFlag is derived as follows:

* If the current picture is a POC resetting picture, the following applies:
  + If vps\_poc\_lsb\_aligned\_flag is equal to 0, pocResettingFlag is set equal to 1.
  + Otherwise, if PocDecrementedInDPBFlag[ nuh\_layer\_id ] is equal to 1, pocResettingFlag is set equal to 0.
  + Otherwise, pocResettingFlag is set equal to 1.
* Otherwise, pocResettingFlag is set equal to 0.

The list affectedLayerList is derived as follows:

* If vps\_poc\_lsb\_aligned\_flag is equal to 0, affectedLayerList consists of the nuh\_layer\_id of the current picture.
* Otherwise, affectedLayerList consists of the nuh\_layer\_id of the current picture and the nuh\_layer\_id values equal to PredictedLayerId[ currNuhLayerId ][ j ] for all values of j in the range of 0 to NumPredictedLayers[ currNuhLayerId ] − 1, inclusive, where currNuhLayerId is the nuh\_layer\_id value of the current picture.

If pocResettingFlag is equal to 1, the following applies:

* When FirstPicInLayerDecodedFlag[ nuh\_layer\_id ] is equal to 1, the following applies:
  + The variables pocMsbDelta, pocLsbDelta and DeltaPocVal are derived as follows:

if( poc\_reset\_idc = = 3 )  
 pocLsbVal = poc\_lsb\_val  
 else  
 pocLsbVal = slice\_pic\_order\_cnt\_lsb  
 if( poc\_msb\_val\_present\_flag )  
 pocMsbDelta = poc\_msb\_val \* MaxPicOrderCntLsb  
 else {  
 prevPicOrderCntLsb = PrevPicOrderCnt[ nuh\_layer\_id ] & ( MaxPicOrderCntLsb − 1 )  
 prevPicOrderCntMsb = PrevPicOrderCnt[ nuh\_layer\_id ] − prevPicOrderCntLsb  
 pocMsbDelta = getCurrMsb( pocLsbVal, prevPicOrderCntLsb, prevPicOrderCntMsb,  
 MaxPicOrderCntLsb )  
 }  
 if( poc\_reset\_idc = = 2 | | ( poc\_reset\_idc = = 3 && full\_poc\_reset\_flag ) )  
 pocLsbDelta = pocLsbVal  
 else  
 pocLsbDelta = 0  
 DeltaPocVal = pocMsbDelta + pocLsbDelta

* + The PicOrderCntVal of each picture that is in the DPB and has nuh\_layer\_id value nuhLayerId for which PocDecrementedInDPBFlag[ nuhLayerId ] is equal to 0 and that is equal to any value in affectedLayerList is decremented by DeltaPocVal.
  + PocDecrementedInDPBFlag[ nuhLayerId ] is set equal to 1 for each value of nuhLayerId included in affectedLayerList.
* The PicOrderCntVal of the current picture is derived as follows:

if( poc\_reset\_idc = = 1 )  
 PicOrderCntVal = slice\_pic\_order\_cnt\_lsb  
else if( poc\_reset\_idc = = 2 )  
 PicOrderCntVal = 0  
else { // poc\_reset\_idc = = 3  
 PicOrderCntMsb = getCurrMsb( slice\_pic\_order\_cnt\_lsb, full\_poc\_reset\_flag ? 0 : poc\_lsb\_val,  
 0, MaxPicOrderCntLsb )  
 PicOrderCntVal = PicOrderCntMsb + slice\_pic\_order\_cnt\_lsb  
}

Otherwise, the following applies:

* The PicOrderCntVal of the current picture is derived as follows:

if( poc\_msb\_val\_present\_flag )  
 PicOrderCntMsb = poc\_msb\_val \* MaxPicOrderCntLsb   
else if(!FirstPicInLayerDecodedFlag[ nuh\_layer\_id ] | |  
 nal\_unit\_type = = IDR\_N\_LP | | nal\_unit\_type = = IDR\_W\_RADL )  
 PicOrderCntMsb = 0  
else {  
 prevPicOrderCntLsb = PrevPicOrderCnt[ nuh\_layer\_id ] & ( MaxPicOrderCntLsb − 1 ).  
 prevPicOrderCntMsb = PrevPicOrderCnt[ nuh\_layer\_id ] − prevPicOrderCntLsb  
 PicOrderCntMsb = getCurrMsb( slice\_pic\_order\_cnt\_lsb, prevPicOrderCntLsb, prevPicOrderCntMsb,  
 MaxPicOrderCntLsb )  
}  
PicOrderCntVal = PicOrderCntMsb + slice\_pic\_order\_cnt\_lsb

The value of PrevPicOrderCnt[ lId ] for each of the lId values included in affectedLayerList is derived as follows:

* + If 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 and discardable\_flag equal to 0, PrevPicOrderCnt[ lId ] is set equal to PicOrderCntVal.
  + Otherwise, when poc\_reset\_idc is equal to 3 and one of the following conditions is true, PrevPicOrderCnt[ lId ] is set equal to ( full\_poc\_reset\_flag ? 0 : poc\_lsb\_val ):
  + FirstPicInLayerDecodedFlag[ nuh\_layer\_id ] is equal to 0.
  + FirstPicInLayerDecodedFlag[ nuh\_layer\_id ] is equal to 1 and the current picture is a POC resetting picture.

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‑43)

The function DiffPicOrderCnt( picA, picB ) is specified as follows:

DiffPicOrderCnt( picA, picB ) = PicOrderCnt( picA ) − PicOrderCnt( picB ) (F‑44)

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

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 = 0; i < NumPocStCurrAfter && rIdx < NumRpsCurrTempList0; rIdx++, i++ ) (F‑45)   
 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 ]  
}

The list RefPicList0 is constructed as follows:

for( rIdx = 0; rIdx <= num\_ref\_idx\_l0\_active\_minus1; rIdx++) (F‑46)  
 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 = 0; i < NumPocStCurrBefore && rIdx < NumRpsCurrTempList1; rIdx++, i++ ) (F‑47)  
 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 ]  
}

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++) (F‑48)  
 RefPicList1[ rIdx ] = ref\_pic\_list\_modification\_flag\_l1 ? RefPicListTemp1[ list\_entry\_l1[ rIdx ] ] :  
 RefPicListTemp1[ rIdx ]

* + 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.

* + 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
     1. Sub-bitstream extraction process

Inputs to this process are a bitstream, a target highest TemporalId value tIdTarget, and a target layer identifier list layerIdListTarget.

Output of this process is a sub-bitstream.

If vps\_base\_layer\_internal\_flag is equal to 1, it is a requirement of bitstream conformance for the input bitstream that any output sub-bitstream of the process specified in this clause with tIdTarget equal to any value in the range of 0 to 6, inclusive, and layerIdListTarget equal to the layer identifier list associated with a layer set specified in the active VPS shall be a conforming bitstream.

NOTE 1 – When vps\_base\_layer\_internal\_flag is equal to 1, a conforming bitstream contains one or more coded slice segment NAL units with nuh\_layer\_id equal to 0 and TemporalId equal to 0.

Otherwise (vps\_base\_layer\_internal\_flag is equal to 0), it is a requirement of bitstream conformance for the input bitstream that any output sub-bitstream of the process specified in this clause with tIdTarget equal to any value in the range of 0 to 6, inclusive, and layerIdListTarget equal to the layer identifier list associated with a layer set specified in the active VPS shall be a conforming bitstream according to at least one profile in which vps\_base\_layer\_internal\_flag may be equal to 0.

The output sub-bitstream is derived as follows:

– When one or more of the following two conditions are true, remove all SEI NAL units that have nuh\_layer\_id equal to 0 and that contain a non-nested buffering period SEI message, a non-nested picture timing SEI message, or a non-nested decoding unit information SEI message:

– layerIdListTarget does not include all the values of nuh\_layer\_id in all NAL units in the bitstream.

– tIdTarget is less than the greatest TemporalId in all NAL units in the bitstream.

NOTE 2 – A "smart" bitstream extractor may include appropriate non-nested buffering picture SEI messages, non-nested picture timing SEI messages, and non-nested decoding unit information SEI messages in the extracted sub-bitstream, provided that the SEI messages applicable to the sub-bitstream were present as nested SEI messages in the original bitstream.

– Remove all NAL units with TemporalId greater than tIdTarget or nuh\_layer\_id not among the values included in layerIdListTarget.

* + 1. Non-base layer subtree extraction process

This process may be applied when num\_add\_layer\_sets is greater than 0 in the active VPSs of the input bitstream.

Inputs to this process are a bitstream inBitstream and for each CVS of the input bitstream a layer set index lsIdx in the range of FirstAddLayerSetIdx to LastAddLayerSetIdx, inclusive.

Output of this process is a bitstream outBitstream.

It is a requirement of bitstream conformance for the input bitstream that any output bitstream of the process specified in this clause with lsIdx in the range of FirstAddLayerSetIdx to LastAddLayerSetIdx, inclusive, shall otherwise be a conforming bitstream except that the output bitstream is not required to contain any VPS NAL units, when all the VCL NAL units of the output bitstream have nuh\_layer\_id equal to 0.

NOTE 1 – When an additional layer set contains only one layer, a VPS rewriting SEI message for that additional layer set is not required to be present.

The bitstream outBitstream is derived from the bitstream inBitstream by applying the following ordered steps:

– The bitstream outBitstream is set to be identical to the bitstream inBitstream.

– NAL units with nal\_unit\_type not equal to SPS\_NUT, PPS\_NUT, EOS\_NUT, and EOB\_NUT and with nuh\_layer\_id not among the nuh\_layer\_id values of the layer set with index lsIdx are removed from outBitstream.

– NAL units with nal\_unit\_type equal to SPS\_NUT, PPS\_NUT, EOS\_NUT, or EOB\_NUT and with nuh\_layer\_id not equal to 0 and not among the nuh\_layer\_id values of the layer set with index lsIdx are removed from outBitstream.

– SEI NAL units including a scalable nesting SEI message for which at least one of the following conditions is true are removed from outBitstream:

– bitstream\_subset\_flag is equal to 1.

– nesting\_op\_flag is equal to 1.

– nesting\_op\_flag is equal to 0 and all\_layers\_flag is equal to 1.

– nesting\_op\_flag is equal to 0, all\_layers\_flag is equal to 0, and nesting\_layer\_id[ i ] for any value of i in the range of 0 to nesting\_num\_layers\_minus1, inclusive, is not among the layer set with index lsIdx.

– For each NAL unit, the following applies:

– When nuh\_layer\_id is equal to AssignedBaseLayerId[ lsIdx ], nuh\_layer\_id is set equal to 0.

– For each SEI NAL unit containing an OLS nesting SEI message for which all the following conditions are true:

– The OLS nesting SEI message contains a VPS rewriting SEI message.

– ols\_idx[ i ] is equal to lsIdx for a value of i in the range of 0 to num\_ols\_indices\_minus1, inclusive.

the following applies:

– If an access unit delimiter NAL unit is present in the same access unit that contains the SEI NAL unit, the nal\_unit( ) in the payload of the VPS rewriting SEI message is inserted into outBitstream as the first NAL unit following the access unit delimiter NAL unit in decoding order.

– Otherwise, the nal\_unit( ) in the payload of the VPS rewriting SEI message is inserted into outBitstream as the first NAL unit, in decoding order, of the access unit.

– The SEI NAL unit is removed.

– For each SEI NAL unit oldSeiNalUnit containing an OLS nesting SEI message for which all the following conditions are true:

– The OLS nesting SEI message contains a scalable nesting SEI message that contains one or more of the following: buffering period SEI message, picture timing SEI message, decoding unit information SEI message.

– ols\_idx[ i ] is equal to lsIdx for a value of i in the range of 0 to num\_ols\_indices\_minus1, inclusive.

the following applies:

– A new prefix SEI NAL unit is added in the same access unit that contains the SEI NAL unit oldSeiNalUnit before the first VCL NAL unit of the access unit, where the NAL unit payload of the new prefix SEI NAL unit consists of the scalable nesting SEI message. nuh\_layer\_id of the new prefix SEI NAL unit is equal to 0 and nuh\_temporal\_id\_plus1 of the new prefix SEI NAL unit is equal to 1.

– The SEI NAL unit oldSeiNalUnit is removed.

– All SEI NAL units containing an OLS nesting SEI message or a bitstream partition nesting SEI message are removed.

– For each SEI NAL unit containing a scalable nesting SEI message, the following applies:

– For each value of i in the range of 0 to nesting\_num\_layers\_minus1, inclusive, the following applies:

– When nesting\_layer\_id[ i ] in a scalable nesting SEI message is equal to AssignedBaseLayerId[ lsIdx ], nesting\_layer\_id[ i ] is set equal to 0.

* 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. (JO): Could be better to put all about the SEI messages directly to annex D, and VUI related stuff directly to annex E. This can however still be done when the new edition is produced.]

*The semantics of the structure of pictures information SEI message specified in subclause D.3.18 are replaced with the following (changed parts are highlighted in turquois):*

The structure of pictures information SEI message provides information for a list of entries, some of which correspond to the target picture set consists of a series of pictures starting from the current picture until the last picture in decoding order in the CVS or the last picture in decoding order in the current POC resetting period, whichever is earlier.

The first entry in the structure of pictures information SEI message corresponds to the current picture. When there is a picture in the target picture set that has PicOrderCntVal equal to the variable entryPicOrderCnt[ i ] as specified below, the entry i corresponds to a picture in the target picture set. The decoding order of the pictures in the target picture set that correspond to entries in the structure of pictures information SEI message corresponds to increasing values of i in the list of entries.

Any picture picB in the target picture set that has PicOrderCntVal equal to entryPicOrderCnt[ i ] for any i in the range of 0 to num\_entries\_in\_sop\_minus1, inclusive, where PicOrderCntVal is the value of PicOrderCntVal of picB immediately after the invocation of the decoding process for picture order count for picB, shall correspond to an entry in the list of entries.

The structure of pictures information SEI message shall not be present in a CVS for which the active SPS has long\_term\_ref\_pics\_present\_flag equal to 1 or num\_short\_term\_ref\_pic\_sets equal to 0.

The structure of pictures information SEI message shall not be present in any access unit that has TemporalId greater than 0 or contains a RASL, RADL or sub-layer non-reference picture. Any picture in the target picture set that corresponds to an entry other than the first entry described in the structure of pictures information SEI message shall not be an IRAP picture.

**sop\_seq\_parameter\_set\_id** indicates and shall be equal to the sps\_seq\_parameter\_set\_id value of the active SPS. The value of sop\_seq\_parameter\_set\_id shall be in the range of 0 to 15, inclusive.

**num\_entries\_in\_sop\_minus1** plus 1 specifies the number of entries in the structure of pictures information SEI message. num\_entries\_in\_sop\_minus1 shall be in the range of 0 to 1023, inclusive.

**sop\_vcl\_nut**[ i ], when the i-th entry corresponds to a picture in the target picture set, indicates and shall be equal to the nal\_unit\_type value of the picture corresponding to the i-th entry.

**sop\_temporal\_id**[ i ], when the i-th entry corresponds to a picture in the target picture set, indicates and shall be equal to the TemporalId value of the picture corresponding to the i-th entry. The value of 7 for sop\_temporal\_id[ i ] is reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore structure of pictures information SEI messages that contain the value 7 for sop\_temporal\_id[ i ].

**sop\_short\_term\_rps\_idx**[ i ], when the i-th entry corresponds to a picture in the target picture set, indicates and shall be equal to the index, into the list of candidate short-term RPSs included in the active SPS, of the candidate short-term RPS used by the picture corresponding to the i-th entry for derivation of the short-term reference picture set. sop\_short\_term\_rps\_idx[ i ] shall be in the range of 0 to num\_short\_term\_ref\_pic\_sets − 1, inclusive.

**sop\_poc\_delta**[ i ] is used to specify the value of the variable entryPicOrderCnt[ i ] for the i-th entry described in the structure of pictures information SEI message. sop\_poc\_delta[ i ] shall be in the range of ( −MaxPicOrderCntLsb ) / 2 + 1 to MaxPicOrderCntLsb / 2 − 1, inclusive.

The variable entryPicOrderCnt[ i ] is derived as follows:

entryPicOrderCnt[ 0 ] = PicOrderCnt( currPic )  
for( i = 1; i <= num\_entries\_in\_sop\_minus1; i++ )  
 entryPicOrderCnt[ i ] = entryPicOrderCnt[ i − 1 ] + sop\_poc\_delta[ i ] (F‑49)

where currPic is the current picture.

[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** |
| **lnp\_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[ nesting\_op\_idx[ 0 ] ]; i++ ) |  |
| **nal\_initial\_arrival\_delay**[ i ] | u(v) |
| else |  |
| for( i = 0; i < SchedCombCnt[ nesting\_op\_idx[ 0 ] ]; 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 ] |  |
| **num\_sei\_bitstream\_partitions\_minus1**[ lsIdx ] | ue(v) |
| for( i = 0; i <= num\_sei\_bitstream\_partitions\_minus1[ lsIdx ]; i++ ) |  |
| for( j = 0; j < NumLayersInIdList[ 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 <= num\_sei\_bitstream\_partitions\_minus1[ lsIdx ]; j++ ) { |  |
| **sei\_bsp\_comb\_hrd\_idx**[ lsIdx ][ i ][ j ] | u(v) |
| **sei\_bsp\_comb\_sched\_idx**[ lsIdx ][ i ][ j ] | ue(v) |
| } |  |
| } |  |
| } |  |

* + - 1. Sub-bitstream property SEI message syntax

|  |  |
| --- | --- |
| sub\_bitstream\_property( payloadSize ) { | **Descriptor** |
| **sb\_property\_active\_vps\_id** | u(4) |
| **num\_additional\_sub\_streams\_minus1** | ue(v) |
| for( i = 0; i <= num\_additional\_sub\_streams\_minus1; i++) { |  |
| **sub\_bitstream\_mode[** i **]** | u(2) |
| **ols\_idx\_to\_vps**[ i ] | ue(v) |
| **highest\_sublayer\_id**[ i ] | u(3) |
| **avg\_sb\_property\_bit\_rate**[ i ] | u(16) |
| **max\_ sb\_property\_bit\_rate**[ i ] | u(16) |
| } |  |
| } |  |

* + - 1. Alpha channel information SEI message syntax

|  |  |
| --- | --- |
| alpha\_channel\_info( payloadSize ) { | **Descriptor** |
| **alpha\_channel\_cancel\_flag** | u(1) |
| if( !alpha\_channel\_cancel\_flag ) { |  |
| **alpha\_channel\_use\_idc** | u(3) |
| **alpha\_channel\_bit\_depth\_minus8** | u(3) |
| **alpha\_transparent\_value** | u(v) |
| **alpha\_opaque\_value** | u(v) |
| **alpha\_channel\_incr\_flag** | u(1) |
| **alpha\_channel\_clip\_flag** | u(1) |
| if( alpha\_channel\_clip\_flag ) |  |
| **alpha\_channel\_clip\_type\_flag** | u(1) |
| } |  |
| } |  |

* + - 1. Overlay information SEI message syntax

[Ed. (GT): All this "information" and "info" in SEI message names is redundant. However, this is already in base spec.]

|  |  |
| --- | --- |
| overlay\_info( payloadSize ) { | **Descriptor** |
| **overlay\_info\_cancel\_flag** | u(1) |
| if( !overlay\_info\_cancel\_flag ) { |  |
| **overlay\_content\_aux\_id\_minus128** | ue(v) |
| **overlay\_label\_aux\_id\_minus128** | ue(v) |
| **overlay\_alpha\_aux\_id\_minus128** | ue(v) |
| **num\_overlays\_minus1** | ue(v) |
| for( i = 0; i <= num\_overlays\_minus1; i++ ) { |  |
| **overlay\_idx**[ i ] | ue(v) |
| **overlay\_name**[ i ] | st(v) |
| **overlay\_content\_layer\_id**[ i ] | u(6) |
| **overlay\_label\_present\_flag**[ i ] | u(1) |
| if( overlay\_label\_present\_flag[ i ] ) |  |
| **overlay\_label\_layer\_id**[ i ] | u(6) |
| **overlay\_alpha\_present\_flag**[ i ] | u(1) |
| if( overlay\_alpha\_present\_flag[ i ] ) |  |
| **overlay\_alpha\_layer\_id**[ i ] | u(6) |
| if( overlay\_label\_present\_flag[ i ] ) { |  |
| **num\_overlay\_elements\_minus1**[ i ] | ue(v) |
| for( j = 0; j <= num\_overlay\_elements\_minus1[ i ]; j++ ) { |  |
| **overlay\_element\_name**[ i ][ j ] | st(v) |
| **overlay\_element\_label\_min**[ i ][ j ] | u(v) |
| **overlay\_element\_label\_max**[ i ][ j ] | u(v) |
| } |  |
| } |  |
| } |  |
| **overlay\_info\_persistence\_flag** | u(1) |
| } |  |
| } |  |

* + - 1. Temporal motion vector prediction constraints SEI message syntax

|  |  |
| --- | --- |
| temporal\_motion\_vector\_prediction\_constraints( payloadSize ) { | **Descriptor** |
| **prev\_pics\_not\_used\_flag** | u(1) |
| **no\_intra\_layer\_col\_pic\_flag** | u(1) |
| } |  |

* + - 1. Frame-field information SEI message syntax

|  |  |
| --- | --- |
| frame\_field\_info( payloadSize ) { | **Descriptor** |
| **ffinfo\_pic\_struct** | u(4) |
| **ffinfo\_source\_scan\_type** | u(2) |
| **ffinfo\_duplicate\_flag** | u(1) |
| } |  |

* + - 1. OLS nesting SEI message syntax

|  |  |
| --- | --- |
| ols\_nesting( payloadSize ) { | **Descriptor** |
| **ols\_flag** | u(1) |
| **num\_ols\_indices\_minus1** | ue(v) |
| for( i = 0; i <= num\_ols\_indices\_minus1; i++ ) |  |
| **ols\_idx**[ i ] | ue(v) |
| while( !byte\_aligned( ) ) |  |
| **ols\_nesting\_zero\_bit** /\* equal to 0 \*/ | u(1) |
| do |  |
| sei\_message( ) |  |
| while( more\_data\_in\_payload( ) ) |  |
| } |  |

* + - 1. VPS rewriting SEI message syntax

|  |  |
| --- | --- |
| vps\_rewriting( payloadSize ) { | **Descriptor** |
| nal\_unit( payloadSize ) |  |
| } |  |

* + 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 [Ed. (AR): Would be better to move this to semantics of SEI, as done for other SEI messages.] |
| 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 |
| Sub-bitstream property | The CVS containing the SEI message |
| Alpha channel information | Specified by the syntax of the SEI message |
| Overlay information | Specified by the syntax of the SEI message |
| Temporal motion vector prediction constraints | Specified by the semantics of the SEI message in subclause F.14.2.9 |
| Frame-field information | One or more pictures of the access unit containing the SEI message [Ed. (YK): To be aligned with version 1, this should be "The access unit containing the SEI message", same as picture timing SEI message. This table column should be interpreted as the persistence scope in the temporal direction or on AU level. We should clarify this a bit, and check this for all the new SEI messages, as well as all the old ones taking into account the multi-layer context. (MH): Agreed.] |
| OLS nesting | Depending on the nested SEI messages. Each nested SEI message has the same persistence scope as if the SEI message was not nested |
| VPS rewriting | Specified in subclause F.10.2 |

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.

**lnp\_sei\_active\_vps\_id** identifies the active VPS of the CVS containing the layers not present SEI message. The value of lnp\_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 1 and i is less than MaxLayersMinus1, layer\_not\_present\_flag[ LayerIdxInVps[ PredictedLayerId[ layer\_id\_in\_nuh[ i ] ][ j ] ] ] shall be equal to 1 for all values of j in the range of 0 to NumPredictedLayers[ 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 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-layer prediction of any sample within the identified tile set. [Ed. (AR). Should tile set be defined here?]

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 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 SEI 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. [Ed. (AR). All occurrences of "tile set having nuh\_layer\_id equal to ictsNuhLayerId" may have to be modified based on the definition of tile set.]

**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.

**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 ] 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.

**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 i-th 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 i-th 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.

**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 SEI 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. [Ed (AR). Default value of all\_tiles\_ilc\_idc should be zero?]

* + - 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 shall be equal to 1, default\_op\_flag shall be equal to 0, nesting\_num\_ops\_minus1 shall be equal to 0, and nesting\_op\_idx[ 0 ] shall not 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** is used to specify the bitstream partition to which the contained SEI message applies 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, inclusive, 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 a bitstream partition nesting SEI message is present, it shall have an associated bitstream partition HRD parameters SEI 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.

**bsp\_nesting\_zero\_bit** shall be equal to 0.

* + - 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 a bitstream partition nesting SEI message that is contained in a scalable nesting SEI message. The same bitstream partition nesting SEI message shall also contain a buffering period SEI message.

Let hrdParamIdx be equal to the index of the hrd\_parameters( ) syntax structure in the active VPS for which hrd\_layer\_set\_idx[ hrdParamIdx ] is equal to nesting\_op\_idx[ 0 ] of the scalable nesting SEI message that contains the bitstream partition SEI message containing this bitstream partition initial arrival time SEI message. Let initialCpbRemovalDelayLength be equal to initial\_cpb\_removal\_delay\_length\_minus1 + 1, where initial\_cpb\_removal\_delay\_length\_minus1 is found in the hrdParamIdx-th hrd\_parameters( ) syntax structure in the active VPS.

**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. The length, in bits, of the nal\_initial\_arrival\_delay[ i ] syntax element is equal to initialCpbRemovalDelayLength.

**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. The length, in bits, of the vcl\_initial\_arrival\_delay[ i ] syntax element is equal to initialCpbRemovalDelayLength.

* + - 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.

When both this SEI message and the vps\_vui\_bsp\_hrd\_parameters( ) syntax structure in the active VPS are present, the contents of this SEI message shall be semantically identical to the contents of the vps\_vui\_bsp\_hrd\_parameters( ) syntax structure of the active VPS.

**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 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 ]. It is a requirement of bitstream conformance that when a scalable nesting SEI message includes a bitstream partition HRD parameters SEI message, nesting\_op\_idx[ h ] shall not be equal to 0 for any value of h in the range of 0 to nesting\_num\_ops\_minus1, inclusive.

**num\_sei\_bitstream\_partitions\_minus1**[ lsIdx ] plus 1 specifies the number of bitstream partitions for which HRD parameters are specified for the layer set with index lsIdx.

**sei\_layer\_in\_bsp\_flag**[ lsIdx ][ i ][ j ] specifies that the layer with index LayerSetLayerIdList[ lsIdx ][ 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 the following constraints apply:

– The bitstream partition with index j shall not include direct or indirect reference layers of any layers in the bitstream partition with index i for any values of i and j in the range of 0 to num\_sei\_bitstream\_partitions\_minus1[ lsIdx ], inclusive, such that i is less than j.

– When vps\_base\_layer\_internal\_flag is equal to 0 and sei\_layer\_in\_bsp\_flag[ lsIdx ][ i ][ 0 ] is equal to 1 for any value of h in the range of 0 to nesting\_num\_ops\_minus1, inclusive, and any value of i in the range of 0 to num\_sei\_bitstream\_partitions\_minus1[ lsIdx ], inclusive, the value of sei\_layer\_in\_bsp\_flag[ lsIdx ][ i ][ j ] for at least one value of j in the range of 1 to NumLayersInIdList[ lsIdx ] − 1, inclusive, shall be equal to 1.

[Ed. (GT): The following item corresponds to items 5/6 in Q0101 and might, according to meeting notes, require further alignment for the case that the base layer is externally specified.]

– When num\_sei\_bitstream\_partitions\_minus1[ lsIdx ] is equal to 0 for any value of h in the range 0 to nesting\_num\_ops\_minus1, inclusive, the value of sei\_layer\_in\_bsp\_flag[ lsIdx ][ 0 ][ j ] should be equal to 0 for at least one value of j in the range 0 to NumLayersInIdList[ lsIdx ] − 1, inclusive.

– For any value of h in the range 1 to nesting\_num\_ops\_minus1, inclusive, the value of sei\_layer\_in\_bsp\_flag[ h ][ i ][ j ] shall be equal to 1 for at most one value of i in the range 0 to num\_sei\_bitstream\_partitions\_minus1, inclusive.

**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.

The variable SchedCombCnt[ lsIdx ] is set equal to sei\_num\_bsp\_sched\_combinations\_minus1[ lsIdx ] + 1.

**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. The value of sei\_bsp\_comb\_hrd\_idx[ lsIdx ][ i ][ j ] shall be in the range of 0 to sei\_num\_bsp\_hrd\_parameters\_minus1, inclusive. The length of the sei\_bsp\_comb\_hrd\_idx[ lsIdx ][ i ][ j ] syntax element is Ceil( Log2( sei\_num\_bsp\_hrd\_parameters\_minus1 + 1 ) ) bits.

**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. The value of sei\_bsp\_comb\_sched\_idx[ lsIdx ][ i ][ j ] 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 from the hrd\_parameters( ) syntax structure corresponding to the index sei\_bsp\_comb\_hrd\_idx[ lsIdx ][ i ][ j ].

* + - 1. Sub-bitstream property SEI message semantics

The sub-bitstream property SEI message, when present, provides the bit rate information for a sub-bitstream created by discarding those pictures in the layers that do not belong to the output layers of the OLSs specified by the active VPS and that do not affect the decoding of the output layers.

When present, the sub-bitstream property SEI message shall be associated with an initial IRAP access unit, and the information provided by the SEI messages applies to the bitstream corresponding to the CVS containing the associated initial IRAP access unit.

**sb\_property\_active\_vps\_id** identifies the active VPS. The value of sb\_property\_active\_vps\_id shall be equal to the value of vps\_video\_parameter\_set\_id of the active VPS referred to by the VCL NAL units of the associated access unit.

**num\_additional\_sub\_streams\_minus1** plus 1 specifies the number of the sub-bitstreams for which the bit rate information may be provided by this SEI message. The value of num\_additional\_sub\_streams\_minus1 shall be in the range of 0 to 210− 1, inclusive.

**sub\_bitstream\_mode[** i **]** specifies how the i-th sub-bitstream is generated. The value of sub\_bitstream\_mode[ i ] shall be equal to 0 or 1, inclusive. The values 2 and 3 are reserved for future use by ITU-T and ISO/IEC. When sub\_bitstream\_mode[ i ] is the greater than 1, decoders shall ignore the syntax elements ols\_idx\_to\_vps[ i ], highest\_sublayer\_id[ i ], avg\_sb\_property\_bit\_rate[ i ], and max\_sb\_property\_bit\_rate[ i ].

When sub\_bitstream\_mode[ i ] is equal to 0, the i-th sub-bitstream is generated as specified by the following steps:

– The sub-bitstream extraction process as specified in clause 10 is invoked with the bitstream corresponding to the CVS containing the sub-bitstream property SEI message, highest\_sublayer\_id[ i ], and LayerSetLayerIdList[ OlsIdxToLsIdx[ ols\_idx\_to\_vps[ i ] ] ] as inputs.

– Remove all NAL units for which the nuh\_layer\_id is not included in TargetOptLayerIdList and either of the following conditions is true:

– The value of nal\_unit\_type is not in the range of BLA\_W\_LP to RSV\_IRAP\_VCL23, inclusive, and max\_tid\_il\_ref\_pics\_plus1[ LayerIdxInVps[ nuh\_layer\_id ] ][LayerIdxInVps[ layerId ] ] is equal to 0 for layerId values included in TargetOptLayerIdList.

– TemporalId is greater than the maximum value of max\_tid\_il\_ref\_pics\_plus1[ LayerIdxInVps[ nuh\_layer\_id ] ][LayerIdxInVps[ layerId ] ] − 1 for all layerId values included in TargetOptLayerIdList.

When sub\_bitstream\_mode[ i ] is equal to 1, the i-th sub-bitstream is generated as specified by the above steps followed by:

– Remove all NAL units with nuh\_layer\_id not among the values included in TargetOptLayerIdList and with discardable\_flag equal to 1.

**ols\_idx\_to\_vps[** i **]** specifies the index of the OLS corresponding to the i-th sub-bitstream.

**highest\_sublayer\_id**[ i ] specifies the highest TemporalId of access units in the i-th sub-bitstream.

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

BitRateBPS( x ) = ( x & ( 214 − 1 ) ) \* 10( 2 + ( x >> 14 ) ) (F‑50)

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 i-th sub-bitstream, 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\_sb\_property\_bit\_rate[ i ], 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‑51)

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

( x & ( 214 − 1 ) ) = = 0 (F‑52)

**max\_sb\_property\_bit\_rate**[ i ] indicates an upper bound for the bit rate of the i-th sub-bitstream 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\_sb\_property\_bit\_rate[ i ] ). 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\_sb\_property\_bit\_rate[ i ], the following condition shall be obeyed for all values of t1:

( x & ( 214 − 1 ) ) >= bTotal ÷ ( ( t2 − t1 ) \* 10( 2 + ( x >> 14 ) ) ) (F‑53)

* + - 1. Alpha channel information SEI message semantics

The alpha channel information SEI message provides information about alpha channel sample values and post-processing applied to the decoded alpha planes coded in auxiliary pictures of type AUX\_ALPHA, and one or more associated primary pictures.

For an auxiliary picture with nuh\_layer\_id equal to nuhLayerIdA and AuxId[ nuhLayerIdA ] equal to AUX\_ALPHA, an associated primary picture, if any, is a 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.

**alpha\_channel\_cancel\_flag** equal to 1 indicates that the alpha channel information SEI message cancels the persistence of any previous alpha channel information SEI message in output order. alpha\_channel\_cancel\_flag equal to 0 indicates that alpha channel information follows. [Ed. (YK): The following persistency scope of this SEI message needs to be adjusted to make it clear for multi-layer context, probably similarly as did for other SEI messages with similar syntax-specified scope.]

When an access unit contains an auxiliary picture picA with nuh\_layer\_id equal to nuhLayerIdA and AuxId[ nuhLayerIdA ] equal to AUX\_ALPHA, the alpha channel sample values of picA persist in output order until one or more of the following conditions are true:

– A new CVS begins.

– The bitstream ends.

– A picture picB in an access unit containing an alpha channel information SEI message is output having PicOrderCnt( picB ) greater than PicOrderCnt( picA ) and alpha\_channel\_cancel\_flag in the alpha channel information SEI message is equal to 1, where PicOrderCnt( picB ) and PicOrderCnt( picA ) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

**alpha\_channel\_use\_idc** equal to 0 indicates that for alpha blending purposes the decoded samples of the associated primary picture should be multiplied by the interpretation sample values of the auxiliary coded picture in the display process after output from the decoding process. alpha\_channel\_use\_idc equal to 1 indicates that for alpha blending purposes the decoded samples of the associated primary picture should not be multiplied by the interpretation sample values of the auxiliary coded picture in the display process after output from the decoding process. alpha\_channel\_use\_idc equal to 2 indicates that the usage of the auxiliary picture is unspecified. Values greater than 2 for alpha\_channel\_use\_idc are reserved for future use by ITU-T | ISO/IEC. When not present, the value of alpha\_channel\_use\_idc is inferred to be equal to 2.

**alpha\_channel\_bit\_depth\_minus8** plus 8 specifies the bit depth of the samples of the sample array of the auxiliary picture. alpha\_channel\_bit\_depth\_minus8 shall be in the range 0 to 7 inclusive. alpha\_channel\_bit\_depth\_minus8 shall be equal to bit\_depth\_luma\_minus8 of the associated primary picture.

**alpha\_transparent\_value** specifies the interpretation sample value of an auxiliary coded picture sample for which the associated luma and chroma samples of the primary coded picture are considered transparent for purposes of alpha blending. The number of bits used for the representation of the alpha\_transparent\_value syntax element is alpha\_channel\_bit\_depth\_minus8 + 9.

**alpha\_opaque\_value** specifies the interpretation sample value of an auxiliary coded picture sample for which the associated luma and chroma samples of the primary coded picture are considered opaque for purposes of alpha blending. The number of bits used for the representation of the alpha\_opaque\_value syntax element is alpha\_channel\_bit\_depth\_minus8 + 9.

**alpha\_channel\_incr\_flag** equal to 0 indicates that the interpretation sample value for each decoded auxiliary picture sample value is equal to the decoded auxiliary picture sample value for purposes of alpha blending. alpha\_channel\_incr\_flag equal to 1 indicates that, for purposes of alpha blending, after decoding the auxiliary picture samples, any auxiliary picture sample value that is greater than Min( alpha\_opaque\_value, alpha\_transparent\_value ) should be increased by one to obtain the interpretation sample value for the auxiliary picture sample, and any auxiliary picture sample value that is less than or equal to Min( alpha\_opaque\_value, alpha\_transparent\_value ) should be used without alteration as the interpretation sample value for the decoded auxiliary picture sample value. When not present, the value of alpha\_channel\_incr\_flag is inferred to be equal to 0.

**alpha\_channel\_clip\_flag** equal to 0 indicates that no clipping operation is applied to obtain the interpretation sample values of the decoded auxiliary picture. alpha\_channel\_clip\_flag equal to 1 indicates that the interpretation sample values of the decoded auxiliary picture are altered according to the clipping process described by the alpha\_channel\_clip\_type\_flag syntax element. When not present, the value of alpha\_channel\_clip\_flag is inferred to be equal to 0.

**alpha\_channel\_clip\_type\_flag** equal to 0 indicates that, for purposes of alpha blending, after decoding the auxiliary picture samples, any auxiliary picture sample that is greater than ( alpha\_opaque\_value − alpha\_transparent\_value ) / 2 is set equal to alpha\_opaque\_value to obtain the interpretation sample value for the auxiliary picture sample, and any auxiliary picture sample that is less or equal than ( alpha\_opaque\_value − alpha\_transparent\_value ) / 2 is set equal to alpha\_transparent\_value to obtain the interpretation sample value for the auxiliary picture sample. alpha\_channel\_clip\_type\_flag equal to 1 indicates that, for purposes of alpha blending, after decoding the auxiliary picture samples, any auxiliary picture sample that is greater than alpha\_opaque\_value is set equal to alpha\_opaque\_value to obtain the interpretation sample value for the auxiliary picture sample, and any auxiliary picture sample that is less than or equal to alpha\_transparent\_value is set equal to alpha\_transparent\_value to obtain the interpretation sample value for the auxiliary picture sample.

NOTE – When both alpha\_channel\_incr\_flag and alpha\_channel\_clip\_flag are equal to one, the clipping operation specified by alpha\_channel\_clip\_type\_flag should be applied first followed by the alteration specified by alpha\_channel\_incr\_flag to obtain the interpretation sample value for the auxiliary picture sample.

* + - 1. Overlay information SEI message semantics

The overlay information SEI message provides information about overlay pictures coded as auxiliary pictures. Overlay auxiliary pictures have nuh\_layer\_id equal to nuhLayerIdA and AuxId[ nuhLayerIdA ] in the range of 128 to 143, inclusive. Each overlay auxiliary picture layer is associated with one or more primary picture layers as specified below.

**overlay\_info\_cancel\_flag** equal to 1 indicates that the SEI message cancels the persistence of any previous overlay information SEI message in output order. overlay\_info\_cancel\_flagequal to 0 indicates that overlay information follows.

**overlay\_content\_aux\_id\_minus128** plus128 indicates the value of AuxId of auxiliary pictures containing overlay content. overlay\_content\_aux\_id\_minus128 shall be in the range of 0 to 15, inclusive.

**overlay\_label\_aux\_id\_minus128** plus128 indicates the value of AuxId of auxiliary pictures containing overlay label. overlay\_label\_aux\_id\_minus128 shall be in the range of 0 to 15, inclusive.

**overlay\_alpha\_aux\_id\_minus128** plus128 indicates the value of AuxId of auxiliary pictures containing overlay alpha. overlay\_alpha\_aux\_id\_minus128 shall be in the range of 0 to 15, inclusive.

**num\_overlays\_minus1** plus 1 specifies the number of overlays described by the overlay information SEI message. num\_overlays\_minus1 shall be in the range of 0 to 15, inclusive.

**overlay\_idx**[ i ] indicates the index of the i-th overlay. overlay\_idx[ i ] shall be in the range of 0 to 255, inclusive.

**overlay\_name**[ i ] indicates the name of the i-th overlay. The length of the syntax element overlay\_name[ i ] shall be less than or equal to 256 bytes.

**overlay\_content\_layer\_id**[ i ] indicates the nuh\_layer\_id value of the NAL units of the overlay content of the i-th overlay. AuxId[ overlay\_content\_layer\_id[ i ] ] shall be equal to overlay\_content\_aux\_id\_minus128**+**128 for all values of i in the range of 0 to num\_overlays\_minus1, inclusive.

The value of the variable pLid, which identifies the nuh\_layer\_id value of the primary picture which the i-th overlay is associated with, is derived as follows:

pLid = −1

for( j = 0; j < 63; j++ )

if( ViewOrderIdx[ j ] = = ViewOrderIdx[ overlay\_content\_layer\_id[ i ] ] &&

ScalabilityId[ j ][ 2 ] = = ScalabilityId[ overlay\_content\_layer\_id[ i ] ][ 2 ] &&

AuxId[ j ] = = 0 )

pLid = j [Ed. (YK): The style of the pseudo code should be aligned with other pseudo codes.]

The value of pLid shall be in the range of 0 to 62, inclusive.

**overlay\_label\_present\_flag**[ i ] equal to 1 specifies that overlay\_label\_layer\_id[ i ] is present.overlay\_label\_present\_flag[ i ] equal to 0 specifies that overlay\_label\_layer\_id[ i ] is not present.

**overlay\_label\_layer\_id**[ i ] indicates the nuh\_layer\_id value of NAL units in the overlay label of the i-th overlay. AuxId[overlay\_label\_layer\_id[ i ] ] shall be equal to overlay\_label\_aux\_id\_minus128 + 128 for all values of i in the range of 0 to num\_overlays\_minus1, inclusive.

**overlay\_alpha\_layer\_id**[ i ] indicates the nuh\_layer\_id value of NAL units in the overlay alpha of the i-th overlay. AuxId[overlay\_alpha\_layer\_id[ i ] ] shall be equal to overlay\_alpha\_aux\_id\_minus128 + 128 for all values of i in the range of 0 to num\_overlays\_minus1, inclusive.

**overlay\_alpha\_present\_flag**[ i ] equal to 1 specifies that overlay\_alpha\_layer\_id[ i ] is present.overlay\_alpha\_present\_flag[ i ] equal to 0 specifies that overlay\_alpha\_layer\_id[ i ] is not present.

**num\_overlay\_elements\_minus1**[ i ] indicates the number of overlay elements in the i-th overlay. When not present, the value of num\_over\_elements\_minus1[ i ] is inferred to be equal to 0.

**overlay\_element\_name**[ i ][ j ] indicates the name of the j-th overlay element of the i-th overlay. The length of the syntax element overlay\_element\_name[ i ][ j ] shall be less than or equal to 256 bytes.

**overlay\_element\_label\_min**[ i ][ j ] and **overlay\_element\_label\_max**[ i ][ j ] indicate the minimum and maximum values, respectively, of the range of sample values corresponding to the j-th overlay element of the i-th overlay.The length of the overlay\_element\_label\_min[ i ][ j ] and overlay\_element\_label\_max[ i ][ j ] is BitDepthY bits.

The variable overlayElementId[ i ][ x ][ y ], the overlay element identifier of the ( x, y ) sample position of the i-th overlay, is derived as follows, where sample*Label*[ x ][ y ] refers to the ( x, y ) sample position of the decoded label auxiliary picture of the i-th overlay:

for( y = 0; y < pic\_height\_in\_luma\_samples; y++ )  
 for( x = 0; x < pic\_width\_in\_luma\_samples; x++ )   
 for( i = 0; i <= number\_overlays\_minus1[ i ] ) {  
 overlayElementId[ i ][ x ][ y ] = 0  
 for( j = 0; j <= num\_overlay\_elements\_minus1[ i ]; j++ )  
 if( sample*Label*[ x ][ y ] >= overlay\_element\_label\_min[ i ][ j ] &&  
 sample*Label*[ x ][ y ] <= overlay\_element\_label\_max[ i ][ j ] )  
 overlayElementId [ i ][ x ][ y ] = j  
 }

**overlay\_info\_persistence\_flag** specifies the persistence of the overlay information SEI message. overlay\_info\_persistence\_flag equal to 0 specifies that the overlay information SEI message applies to the current decoded picture only.

Let picA be the current picture. overlay\_info\_persistence\_flag equal to 1 specifies that the overlay information SEI message persists in output order until one or more of the following conditions are true:

– A new CVS begins.

– The bitstream ends.

– A picture picB in an access unit containing an overlay information SEI message is output for which PicOrderCnt( picB ) is greater than PicOrderCnt( picA ), where PicOrderCnt( picB ) and PicOrderCnt( picA ) are the PicOrderCntVal values of picB and picA, respectively, immediately after the invocation of the decoding process for picture order count for picB.

* + - 1. Temporal motion vector prediction constraints SEI message semantics

The temporal motion vector prediction constraints SEI message indicates constraints on collocated pictures for temporal motion vector prediction. The SEI message may be used to determine whether the motion vectors of earlier pictures in decoding order no longer need to be stored and whether the motion vectors of the current picture and subsequent pictures need to be stored.

When the SEI message is not included in a scalable nesting SEI message, it applies to pictures with nuh\_layer\_id equal to that of the SEI NAL unit containing the SEI message. When the SEI message is included in a scalable nesting SEI message, it applies to the pictures with nuh\_layer\_id equal to all values of nestingLayerIdList[ i ] derived through the semantics of the containing scalable nesting SEI message. Let associatedLayerIdList be the list of nuh\_layer\_id values to which the temporal motion vector prediction constraints SEI message applies. [Ed. (AR): Check paragraph after integration of Q0183.][Ed. (YK): These should be included in the generic SEI message semantics – should be done during/after integration of Q0183.]

The temporal motion vector prediction constraints SEI message is a prefix SEI message. The temporal motion vector prediction constraints SEI message may be present in an access unit with TemporalId equal to 0 and shall not be present in an access unit with TemporalId greater than 0. [Ed. (AR): Check if sentence to be re-written based on SEI NAL unit and TemporalId of the AU. (MH): I changed references to a picture with references to an access unit, but I did not really understand the first part of AR's comment on writing the sentence based on SEI NAL unit.]

For the semantics below, currLayerId is a single nuh\_layer\_id value in associatedLayerIdList, and the semantics apply separately for each nuh\_layer\_id value in associatedLayerIdList. [Ed. (AR): Check paragraph after integration of Q0183.]

Let a set of pictures associatedPicSet be the pictures with nuh\_layer\_id equal to currLayerId from the access unit containing the SEI message, inclusive, up to but not including the first of any of the following in decoding order:

– The next access unit, in decoding order, that contains a temporal motion vector prediction constraints SEI message with an associatedLayerIdList that contains currLayerId.

– The next IDR or BLA picture, in decoding order, with nuh\_layer\_id equal to currLayerId.

– The next IRAP access unit, in decoding order, with NoClrasOutputFlag equal to 1.

**prev\_pics\_not\_used\_flag** equal to 1 indicates that the syntax elements for all coded pictures that follow the access unit containing the current picture in decoding order are constrained such that no temporal motion vector from any picture that has nuh\_layer\_id equal to any value in associatedLayerIdList and precedes the access unit containing the current picture in decoding order is used in decoding of any coded picture that follows the access unit containing the current picture in decoding order. prev\_pics\_not\_used\_flag equal to 0 indicates that the bitstream may or may not fulfill the constraints indicated by prev\_pics\_not\_used\_flag equal to 1.

NOTE 1 – When prev\_pics\_not\_used\_flag is equal to 1, decoders may empty the "motion vector storage" for all reference pictures with nuh\_layer\_id equal to currLayerId in the decoded picture buffer.

prev\_pics\_not\_used\_flag shall be equal to 1 when all of the following conditions are true:

– no\_intra\_layer\_col\_pic\_flag is equal to 1 in the previous temporal motion vector prediction constraints SEI message applying to nuh\_layer\_id equal to currLayerId.

– There is no IDR or BLA picture with nuh\_layer\_id equal to currLayerId following, in decoding order, the previous temporal motion vector prediction constraints SEI message applying to nuh\_layer\_id equal to curLayerId, and preceding, in decoding order, the current temporal motion vector prediction constraints SEI message.

– There is no IRAP access unit that contains an IRAP picture with nuh\_layer\_id equal to 0 and NoClrasOutputFlag equal to 1 following, in decoding order, the previous temporal motion vector prediction constraints SEI message applying to nuh\_layer\_id equal to currLayerId, and preceding, in decoding order, the current temporal motion vector prediction constraints SEI message.

**no\_intra\_layer\_col\_pic\_flag** equal to 1 indicates the following:

– If NumDirectRefLayers[ currLayerId ] is equal to 0, slice\_temporal\_mvp\_enabled\_flag is not present or is equal to 0 in each picture in associatedPicSet.

– Otherwise, all the pictures in associatedPicSet do not use temporal motion vector prediction or use collocated pictures with nuh\_layer\_id different from currLayerId.

When no\_intra\_layer\_col\_pic\_flag is equal to 0, no constraint for the collocated picture of the pictures in associatedPicSet is indicated.

Let NoIntraLayerColPicFlag[ currLayerId ] be equal to no\_intra\_layer\_col\_pic\_flag.

NOTE 2 – The motion vectors of the current picture with nuh\_layer\_id equal to layerId have to be stored when they may be used for temporal motion vector prediction of other pictures in the same layer or when they may be used for inter-layer motion prediction. In other words, the motion vectors of the current picture have to be stored when at least one of the following is true:

* sps\_temporal\_mvp\_enabled\_flag in the active SPS for the current picture is equal to 1 and NoIntraLayerColPicFlag[ layerId ] is equal 0.
* NoIntraLayerColPicFlag[ layerId ] is equal to 1 and there is a nuh\_layer\_id value nuhLayerIdA such that VpsInterLayerMotionPredictionEnabled[ LayerIdxInVps[ nuhLayerIdA ] ][ LayerIdxInVps[ layerId ] ] is equal to 1.

NOTE 3 – The motion vectors of a picture with nuh\_layer\_id equal to layerId need no longer be stored when the picture is marked as "unused for reference", or the picture is not used for temporal motion vector prediction of other pictures in the same layer and all pictures in the same access unit that may use the picture as a reference for inter-layer motion prediction have been decoded, or the access unit containing the picture precedes the current access unit in decoding order, where this SEI message is present with associatedLayerIdList including the nuh\_layer\_id of the picture and prev\_pics\_not\_used\_flag equal to 1. In other words, the motion vectors of a picture need no longer be stored when at least one of the following is true: [Ed. (AR): This note repeats the conditions. Is it not sufficient to say it once?]

* The picture is marked as "unused for reference".
* NoIntraLayerColPicFlag[ layerId ] is equal to 1 for the picture and the access unit containing the picture has been decoded.
* The access unit containing the picture precedes the current access unit, in decoding order, and the current access unit contains this SEI message with associatedLayerIdList including the nuh\_layer\_id value of the picture and with prev\_pics\_not\_used\_flag equal to 1.
  + - 1. Frame-field information SEI message semantics

The frame-field information SEI message may be used to indicate how the associated picture should be displayed, the source scan type of the associated picture, and whether the associated picture is a duplicate of a previous picture, in output order, of the same layer.

When a non-nested frame-field information SEI message is included in an SEI NAL unit with nuh\_layer\_id assocLayerId, it is associated to the picture within nuh\_layer\_id equal to assocLayerId within the same access unit as the SEI NAL unit. When a frame-field information SEI message is included in a scalable nesting SEI message, it is associated to each picture with nuh\_layer\_id assocLayerId equal to nestingLayerIdList[ i ] for all values of i in the range of 0 to nesting\_num\_layers\_minus1, inclusive. [Ed. (YK): These should be included in the generic SEI message semantics – should be done during/after integration of Q0183.]

When a picture with nuh\_layer\_id equal to assocLayerId is present in an access unit, frame\_field\_info\_present\_flag is equal to 1 in the active SPS for the layer with nuh\_layer\_id assocLayerId, and a non-nested picture timing SEI message is not present in an SEI NAL unit with nuh\_layer\_id equal to assocLayerId, a frame-field information SEI message associated with nuh\_layer\_id equal to assocLayerId shall be present in the access unit.

The semantics of ffinfo\_pic\_struct, ffinfo\_source\_scan\_type and ffinfo\_duplicate\_flag apply layer-wise to each value of assocLayerId.

**ffinfo\_pic\_struct** has the same semantics as the pic\_struct syntax element in the picture timing SEI message.

**ffinfo\_source\_scan\_type** has the same semantics as the source\_scan\_type syntax element in the picture timing SEI message.

**ffinfo\_duplicate\_flag** has the same semantics as the duplicate\_flag syntax element in the picture timing SEI message.

* + - 1. OLS nesting SEI message semantics

The OLS nesting SEI message provides a mechanism to associate SEI messages with one or more additional layer sets or one or more OLSs.

An OLS nesting SEI message contains one or more SEI messages.

**ols\_flag** equal to 0 specifies that the nested SEI messages are associated with additional layer sets identified through ols\_idx[ i ]. ols\_flag equal to 1 specifies that the nested SEI messages are associated with OLSs identified through ols\_idx[ i ]. When num\_add\_layer\_sets is equal to 0, ols\_flag shall be equal to 1.

**num\_ols\_indices\_minus1** plus 1 specifies the number of indices of additional layer sets or OLSs the nested SEI messages are associated with. num\_ols\_indices\_minus1 shall be in the range of 0 to 2047, inclusive.

**ols\_idx**[ i ] specifies an index of the additional layer set or the OLS specified in the active VPS to which the nested SEI messages are associated with. If ols\_flag is equal to 0, ols\_idx[ i ] shall be in the range of vps\_num\_layer\_sets\_minus1 + 1 to vps\_num\_layer\_sets\_minus1 + num\_add\_layer\_sets, inclusive. Otherwise (ols\_flag is equal to 1), ols\_idx[ i ] shall be in the range of 0 to NumOutputLayerSets – 1, inclusive.

**ols\_nesting\_zero\_bit** shall be equal to 0.

* + - 1. VPS rewriting SEI message semantics

The VPS rewriting SEI message contains a nal\_unit( ) syntax structure in which nal\_unit\_type shall be equal to VPS\_NUT. The VPS rewriting SEI message contains a VPS NAL unit that is added to the output bitstream of the non-base layer subtree extraction process as specified in subclause F.10.2.

When an OLS has an index olsIdx in the range of FirstAddLayerSetIdx to LastAddLayerSetIdx, inclusive, and the OLS includes more than one layer, a VPS rewriting SEI message shall be present within an OLS nesting SEI message with ols\_idx[ i ] equal to olsIdx.

The VPS rewriting SEI message, when present, shall be included in an OLS nesting SEI message in which ols\_flag shall be equal to 0 and ols\_idx[ i ] shall be in the range of FirstAddLayerSetIdx to LastAddLayerSetIdx, inclusive, for each value of i in the range of 0 to num\_ols\_indices\_minus1, inclusive.

* 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.

**field\_seq\_flag** equal to 1 indicates that the layers for which the SPS is an active SPS within the CVS convey pictures that represent fields, and specifies that a picture timing SEI message or a frame-field information SEI message shall be present for those layers in every access unit of the current CVS. [Ed. (MH): The previous sentence could more explicitly refer to nested and/or non-nested picture timing SEI message, and any requirements of the syntax elements of the scalable nesting SEI message.] field\_seq\_flag equal to 0 indicates that the layers for which the SPS is an active SPS within the CVS convey pictures that represent frames and that a picture timing SEI message or a frame-field information SEI message may or may not be present for the layers for which the SPS is an active SPS in any access unit of the current CVS. When field\_seq\_flag is not present, it is inferred to be equal to 0. When general\_frame\_only\_constraint\_flag is present in the SPS and is equal to 1, the value of field\_seq\_flag shall be equal to 0. When general\_frame\_only\_constraint\_flag is present in the active VPS, applies for a layer for which the SPS is an active SPS, and is equal to 1, the value of field\_seq\_flag shall be equal to 0.

NOTE 4 – The specified decoding process does not treat access units conveying pictures that represent fields or frames differently. A sequence of pictures that represent fields would therefore be coded with the picture dimensions of an individual field. For example, access units containing pictures that represent 1080i fields would commonly have cropped output dimensions of 1920x540, while the sequence picture rate would commonly express the rate of the source fields (typically between 50 and 60 Hz), instead of the source frame rate (typically between 25 and 30 Hz).

**frame\_field\_info\_present\_flag** equal to 1 specifies that picture timing SEI messages or frame-field information SEI messages are present for every picture for which this SPS is the active SPS and the picture timing SEI messages, when present, include the pic\_struct, source\_scan\_type, and duplicate\_flag syntax elements. frame\_field\_info\_present\_flag equal to 0 specifies that the pic\_struct syntax element is not present in picture timing SEI messages associated with pictures for which the SPS is the active SPS.

When frame\_field\_info\_present\_flag is present and either or both of the following conditions are true, frame\_field\_info\_present\_flag shall be equal to 1:

– field\_seq\_flag is equal to 1.

– general\_progressive\_source\_flag and general\_interlaced\_source\_flag are present in this SPS, general\_progressive\_source\_flag is equal to 1, and general\_interlaced\_source\_flag is equal to 1.

When frame\_field\_info\_present\_flag is not present, its value is inferred as follows:

– If general\_progressive\_source\_flag and general\_interlaced\_source\_flag are present in this SPS, general\_progressive\_source\_flag is equal to 1, and general\_interlaced\_source\_flag is equal to 1, frame\_field\_info\_present\_flag is inferred to be equal to 1.

– Otherwise, frame\_field\_info\_present\_flag is inferred to be equal to 0.

**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   
     
   Scalable high efficiency video coding

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

This annex specifies syntax, semantics and decoding processes for scalable high efficiency video coding that use the syntax, semantics, and decoding process specified in clauses 2-9 and Annexes 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

The specifications in clause 6 apply. Additionally, the following processes are specified.

* + 1. Derivation process for reference layer sample location

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 )  
 + RefLayerConfWinLeftOffset \* RefLayerSubWidthC (H‑1)  
yRef = ( ( ( yP − ScaledRefLayerTopOffset ) \* ScaleFactorY + ( 1 << 15 ) ) >> 16 )  
 + RefLayerConfWinTopOffset \* RefLayerSubHeightC (H‑2)

* + 1. Derivation process for reference layer sample location used in resampling

Inputs to this process are

– a variable cIdx specifying the colour component index,

– a sample location ( xP, yP ) relative to the top-left sample of the colour 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, deltaX and deltaY are derived as follows:

– If cIdx is equal to 0, the following applies:

phaseX = CrossLayerPhaseAlignmentFlag << 1 (H‑5)  
phaseY = CrossLayerPhaseAlignmentFlag << 1 (H‑6)

deltaX = ( ( RefLayerConfWinLeftOffset \* RefLayerSubWidthC ) << 4) −  
 ( CrossLayerPhaseAlignmentFlag << 3 ) (H‑7)  
deltaY = ( ( RefLayerConfWinTopOffset \* RefLayerSubHeightC ) << 4) − ( ( ( CrossLayerPhaseAlignmentFlag  
 << (3 – VertPhasePositionAdjustFlag ) ) + ( VertPhasePositionFlag << 3 ) ) (H‑8)

– Otherwise (cIdx is equal to 1), the following applies:

phaseX = CrossLayerPhaseAlignmentFlag (H‑9)  
phaseY = CrossLayerPhaseAlignmentFlag + 1 (H‑10)

deltaX = ( RefLayerConfWinLeftOffset << 4) − ( CrossLayerPhaseAlignmentFlag << 2 ) (H‑11)  
deltaY = ( RefLayerConfWinTopOffset << 4) − ( ( ( CrossLayerPhaseAlignmentFlag + 1 )  
  << ( 2 − VertPhasePositionAdjustFlag ) ) + (VertPhasePositionFlag << 3 ) ) (H‑12)

The variables addX and addY are derived as follows:

addX = ( ScaleFactorX \* phaseX + 2 ) >> 2 (H‑13)   
addY = ( ScaleFactorY \* phaseY + 2 ) >> 2 (H‑14)

The variables xRef16 and yRef16 are derived as follows:

xRef16 = ( ( ( xP − offsetX ) \* ScaleFactorX + addX + ( 1 << 11 ) ) >> 12 ) + deltaX (H‑15)  
yRef16 = ( ( ( yP − offsetY ) \* ScaleFactorY + addY + ( 1 << 11 ) ) >> 12 ) + deltaY (H‑16)

* 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:

– At the beginning of the decoding process for the first slice of the current picture, the process specified in 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 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 variables NumResampling, NumSampleResampling, and NumMotionResampling are set equal to 0.

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++ ) {  
 refPicSet0Flag = ( 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 ] ] ) )  
 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 ilRefPic is derived by invoking the process specified in subclause H.8.1.4 with picX andRefPicLayerId[ i ] given as inputs   
 if( refPicSet0Flag ) {  
 RefPicSetInterLayer0[ NumActiveRefLayerPics0 ] = ilRefPic  
 RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] is marked as "used for long-term reference"  
 } else {  
 RefPicSetInterLayer1[ NumActiveRefLayerPics1 ] = ilRefPic  
 RefPicSetInterLayer1[ NumActiveRefLayerPics1++ ] is marked as "used for long-term reference"  
 }  
 } else {  
 if( refPicSet0Flag )  
 RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] = "no reference picture"  
 else  
 RefPicSetInterLayer1[ NumActiveRefLayerPics1++ ] = "no reference picture"  
 }  
}

There shall be no entry equal to "no reference picture" in RefPicSetInterLayer0 or RefPicSetInterLayer1.

There shall be no picture that has discardable\_flag equal to 1 in RefPicSetInterLayer0 or RefPicSetInterLayer1.

NOTE 1 – 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 2 – 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. Derivation process for inter-layer reference pictures

Inputs to this process are:

– a decoded reference layer picture rlPic,

– a variable rLId specifies the value of nuh\_layer\_id of the reference layer picture.

Output of this process is the inter-layer reference picture ilRefPic.

The variables CurPicWidthInSamplesY and CurPicHeightInSamplesY 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.

It is a requirement of bitstream conformance that RefLayerBitDepthY shall be less than or equal to BitDepthY and RefLayerBitDepthC shall be less than or equal to BitDepthC.

The variables RefLayerConfWinLeftOffset, RefLayerConfWinRightOffset, RefLayerConfWinTopOffset and RefLayerConfWinBottomOffset are set equal to the cropping parameters conf\_win\_left\_offset, conf\_win\_right\_offset, conf\_win\_top\_offset and conf\_win\_bottom\_offset, respectively, of the reference layer picture rlPic. The variables RefLayerSubWidthC and RefLayerSubHeightC are set equal to the values of SubWidthC and SubHeightC of the reference layer picture rlPic, respectively.

The variables RefLayerCroppedPicWidthInSamplesY and RefLayerCroppedPicHeightInSamplesY are the width and height of the copped decoded reference layer picture rlPic in units of luma samples, respectively, and are derived as follows:

RefLayerCroppedPicWidthInSamplesY = RefLayerPicWidthInSamplesY −  
 RefLayerSubWidthC \* ( RefLayerConfWinLeftOffset + RefLayerConfWinRightOffset ) (H‑17)

RefLayerCroppedPicHeightInSamplesY = RefLayerPicHeightInSamplesY −  
 RefLayerSubHeightC \* ( RefLayerConfWinTopOffset + RefLayerConfWinBottomOffset ) (H‑18)

The variables CurPicWidthInSamplesC, CurPicHeightInSamplesC, RefLayerPicWidthInSamplesC, and RefLayerPicHeightInSamplesC are derived as follows:

CurPicWidthInSamplesC = CurPicWidthInSamplesY / SubWidthC (H‑19)  
CurPicHeightInSamplesC = CurPicHeightInSamplesY / SubHeightC (H‑20)  
RefLayerPicWidthInSamplesC = RefLayerPicWidthInSamplesY / RefLayerSubWidthC (H‑21)  
RefLayerPicHeightInSamplesC = RefLayerPicHeightInSamplesY / RefLayerSubHeightC (H‑22)

The variable currLayerId is set equal to the value of 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‑23)  
ScaledRefLayerTopOffset = scaled\_ref\_layer\_top\_offset[ rLId ] << 1 (H‑24)  
ScaledRefLayerRightOffset = scaled\_ref\_layer\_right\_offset[ rLId ] << 1 (H‑25)  
ScaledRefLayerBottomOffset = scaled\_ref\_layer\_bottom\_offset[ rLId ] << 1 (H‑26)  
VertPhasePositionAdjustFlag = vert\_phase\_position\_enable\_flag[ rLId ] (H‑27)  
VertPhasePositionFlag = vert\_phase\_position\_flag[ rLId ] (H‑28)

The variables ScaledRefLayerPicWidthInSamplesY and ScaledRefLayerPicHeightInSamplesY are derived as follows:

ScaledRefLayerPicWidthInSamplesY = CurPicWidthInSamplesY −  
 ScaledRefLayerLeftOffset − ScaledRefLayerRightOffset (H‑29)  
ScaledRefLayerPicHeightInSamplesY = CurPicHeightInSamplesY −  
 ScaledRefLayerTopOffset − ScaledRefLayerBottomOffset (H‑30)

The variables ScaleFactorX and ScaleFactorY are derived as follows:

ScaleFactorX = ( ( RefLayerCroppedPicWidthInSamplesY << 16 ) +  
 ( ScaledRefLayerPicWidthInSamplesY >> 1 ) ) / ScaledRefLayerPicWidthInSamplesY (H‑31)  
ScaleFactorY = ( ( RefLayerCroppedPicHeightInSamplesY << 16 ) +  
 ( ScaledRefLayerPicHeightInSamplesY >> 1 ) ) / ScaledRefLayerPicHeightInSamplesY (H‑32)

The following ordered steps are applied to derive the inter-layer reference picture ilRefPic.

– The variables sampleResamplingFlag and motionResamplingFlag are initialized to 0.

– The variable unequalPictureSizeFlag is derived as follows:

– If all of the following conditions are true, unequalPictureSizeFlag is set equal to 0:

– CurPicWidthInSamplesY is equal to RefLayerPicWidthInSamplesY,

– CurPicHeightInSamplesY is equal to RefLayerPicHeightInSamplesY,

– ScaledRefLayerLeftOffset is equal to RefLayerConfWinLeftOffset \* RefLayerSubWidthC,

– ScaledRefLayerTopOffset is equal to RefLayerConfWinTopOffset \* RefLayerSubHeightC,

– ScaledRefLayerRightOffset is equal to RefLayerConfWinRightOffset \* RefLayerSubWidthC,

– ScaledRefLayerBottomOffset is equal to RefLayerConfWinBottomOffset \* RefLayerSubHeightC.

– Otherwise, unequalPictureSizeFlag is set equal to 1.

– The variable currColourMappingEnableFlag is derived as follows:

– If colour\_mapping\_enabled\_flag is equal to 1 and if there exists a value of i, with i in the range of 0 to num\_cm\_ref\_layers\_minus1, inclusive, for which rLId is equal to cm\_ref\_layer\_id[ i ], currColourMappingEnableFlag is set equal to 1.

– Otherwise, currColourMappingEnableFlag is set equal to 0.

– If unequalPictureSizeFlag is equal to 0, RefLayerBitDepthY is equal to BitDepthY, RefLayerBitDepthC is equal to BitDepthC, and currColourMappingEnableFlag is equal to 0,

– ilRefPic is set equal to rlPic.

– Otherwise, the following steps apply:

– The inter-layer reference picture ilRefPic is generated as follows:

– The PicOrderCntVal value of ilRefPic is set equal to the PicOrderCntVal value of rlPic.

– When VpsInterLayerSamplePredictionEnabled[ LayerIdxInVps[ currLayerId ] ][ LayerIdxInVps[ rLId ] ] is equal to 1, the following steps apply:

– If currColourMappingEnableFlag is equal to 1, the following ordered steps apply:

– The colour mapping process as specified in subclause H.8.1.4.3 is invoked with the picture sample arrays, rlPicSampleL, rlPicSampleCb and rlPicSampleCr, of the reference layer picture rlPic as inputs, and with the colour mapped picture sample arrays, cmPicSampleL, cmPicSampleCb and cmPicSampleCr of the colour mapped reference picture cmRefPic as outputs.

– RefLayerBitDepthY and RefLayerBitDepthC are set equal to CMOutputBitDepthY and CMOutputBitDepthC, respectively.

– If unequalPictureSizeFlag is equal to 0, RefLayerBitDepthY is equal to BitDepthY, and RefLayerBitDepthC is equal to BitDepthC, the following applies:

– rsPicSampleL, rsPicSampleCb and rsPicSampleCr are set equal to cmPicSampleL, cmPicSampleCb and cmPicSampleCr, respectively.

– Otherwise, the picture sample resampling process as specified in subclause H.8.1.4.1 is invoked with the picture sample arrays, cmPicSampleL, cmPicSampleCb and cmPicSampleCr, of the colour mapped reference layer picture cmPic as inputs, and with the resampled picture sample arrays, rsPicSampleL, rsPicSampleCb and rsPicSampleCr of the inter-layer reference picture ilRefPic as outputs.

– Otherwise, the picture sample resampling process as specified in subclause H.8.1.4.1 is invoked with the picture sample arrays, rlPicSampleL, rlPicSampleCb and rlPicSampleCr, of the reference layer picture rlPic as inputs, and with the resampled picture sample arrays, rsPicSampleL, rsPicSampleCb and rsPicSampleCr of the inter-layer reference picture ilRefPic as outputs.

– sampleResamplingFlag is set equal to 1.

– When VpsInterLayerMotionPredictionEnabled[ LayerIdxInVps[ currLayerId ] ][ LayerIdxInVps[ rLId ] ] is equal to 1, the following steps apply:

– A single slice ilRefSlice of the inter-layer reference picture ilRefPic 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 ilRefSlice 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 of rlPic.

– When ilRefSlice is a P or B slice, for i in the range of 0 to num\_ref\_idx\_l0\_active\_minus1 of ilRefSlice, inclusive, the reference picture with index i in reference picture list 0 of ilRefSlice is set equal to the reference picture with index i in reference picture list 0 of the first slice of rlPic.

– When ilRefSlice is a B slice, for i in the range of 0 to num\_ref\_idx\_l1\_active\_minus1 of ilRefSlice, inclusive, the reference picture with index i in reference picture list 1 of ilRefSlice is set equal to the reference picture with index i in reference picture list 1 of the first slice of rlPic.

NOTE – When the inter-layer reference picture ilRefPic is used as the 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.

– If unequalPictureSizeFlag is equal to 0, the following applies:

– The motion field of the inter-layer reference picture ilRefPic, including the array CuPredMode specifying the prediction modes, two arrays RefIdxL0 and RefIdxL1 specifying the reference indices, two arrays MvL0 and MvL1 specifying the luma motion vectors, and two arrays PredFlagL0 and PredFlagL1 specifying the prediction list utilization flags, are set equal to those of the decoded reference layer picture rlPic, respectively.

– Otherwise, the following applies

– The picture motion field resampling process as specified in subclause H.8.1.4.2 is invoked with the reference layer picture rlPic, an array rlPredMode specifying the prediction modes CuPredMode of rlPic, two arrays rlRefIdxLX specifying the reference indices RefIdxLX of rlPic, two arrays rlMvLX specifying the luma motion vectors MvLX of rlPic, and two arrays rlPredFlagLX specifying the prediction list utilization flags PredFlagLX of rlPic, with X = 0,1 as inputs, and an array rsPredMode specifying the prediction modes CuPredMode of ilRefPic, two arrays rsRefIdxLX specifying the reference indices RefIdxLX of ilRefPic, two arrays rsMvLX specifying the luma motion vectors MvLX of ilRefPic, and two arrays rsPredFlagLX specifying the prediction list utilization flags PredFlagLX of ilRefPic, with X = 0,1 as outputs.

– motionResamplingFlag is set equal to 1.

– The following applies:

NumSampleResampling += sampleResamplingFlag (H‑33)

NumMotionResampling += motionResamplingFlag (H‑34)

NumResampling += sampleResamplingFlag | | motionResamplingFlag (H‑35)

* + - * 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 ( CurPicWidthInSamplesY ) x ( CurPicHeightInSamplesY ) array rsPicSampleL of luma samples,

– a ( CurPicWidthInSamplesC ) x ( CurPicHeightInSamplesC ) array rsPicSampleCb of chroma samples of the component Cb,

– a ( CurPicWidthInSamplesC) x ( CurPicHeightInSamplesC ) 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 = CurPicWidthInSamplesY − ScaledRefLayerRightOffset (H‑36)  
topStartL = ScaledRefLayerTopOffset  
bottomEndL = CurPicHeightInSamplesY − ScaledRefLayerBottomOffset (H‑37)

The luma samples rsPicSampleL[ xP ][ yP ] with ( xP = 0 ... CurPicWidthInSamplesY − 1, yP = 0 ... CurPicHeightInSamplesY − 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 = ( CurPicWidthInSamplesY− ScaledRefLayerRightOffset ) / SubWidthC (H‑38)  
topStartC = ScaledRefLayerTopOffset / SubHeightC  
bottomEndC = ( CurPicHeightInSamplesY− ScaledRefLayerBottomOffset ) / SubHeightC (H‑39)

The chroma samples rsPicSampleC[ xPC ][ yPC ] with ( xPC = 0 ... CurPicWidthInSamplesC − 1, yPC = 0 ... CurPicHeightInSamplesC − 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 resampled luma sample value rsLumaSample.

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 resampled luma sample rsLumaSample 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 luma sample location ( xRef16, yRef16 ) in units of 1/16-th luma sample as output.
2. The variables xRef and xPhase are derived as follows:

xRef     = ( xRef16 >> 4 ) (H‑40)  
xPhase = ( xRef16 ) % 16 (H‑41)

1. The variables yRef and yPhase are derived as follows:

yRef     = ( yRef16 >> 4 ) (H‑42)  
yPhase = ( yRef16 ) % 16 (H‑43)

1. The variables shift1, shift2 and offset are derived as follows:

shift1 = RefLayerBitDepthY − 8 (H‑44)  
shift2 = 20 − BitDepthY (H‑45)  
offset = 1 << ( shift2 − 1 ) (H‑46)

1. The sample value tempArray[ n ] with n = 0 … 7, is derived as follows:

yPosRL = Clip3( 0, RefLayerPicHeightInSamplesY − 1, yRef + n − 3 ) (H‑47)  
refW      = RefLayerPicWidthInSamplesY (H‑48)

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‑49)  
 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 resampled luma sample value rsLumaSample is derived as follows:

rsLumaSample = ( 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‑50)  
 fL[ yPhase, 5 ] \* tempArray[ 5 ] +   
 fL[ yPhase, 6 ] \* tempArray[ 6 ] +   
 fL[ yPhase, 7 ] \* tempArray[ 7 ] + offset ) >> shift2

rsLumaSample = Clip3( 0, ( 1 << BitDepthY) − 1 , rsLumaSample ) (H‑51)

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 resampled chroma sample value rsChromaSample.

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 resampled chroma sample value rsChromaSample 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 equal to 1 and chroma sample location ( xPC, yPC ) given as the inputs and chroma sample location ( xRef16, yRef16 ) in units of 1/16-th chroma sample as output.
2. The variables xRef and xPhase are derived as follows:

xRef     = ( xRef16 >> 4 ) (H‑52)  
xPhase = ( xRef16 ) % 16 (H‑53)

1. The variables yRef and yPhase are derived as follows:

yRef     = ( yRef16 >> 4 ) (H‑54)  
yPhase = ( yRef16 ) % 16 (H‑55)

1. The variables shift1, shift2 and offset are derived as follows:

shift1 = RefLayerBitDepthC − 8 (H‑56)  
shift2 = 20 − BitDepthC (H‑57)  
offset =1 << ( shift2 − 1 ) (H‑58)

1. The sample value tempArray[ n ] with n = 0 … 3, is derived as follows:

yPosRL = Clip3( 0, RefLayerPicHeightInSamplesC − 1, yRef + n − 1 ) (H‑59)  
refWC   = RefLayerPicWidthInSamplesC (H‑60)

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‑61)  
 fC[ xPhase, 3 ] \* rlPicSampleC[ Clip3( 0, refWC − 1, xRef + 2 ), yPosRL ] ) >> shift1

1. The resampled chroma sample value rsChromaSample is derived as follows:

rsChromaSample = (fC[ yPhase, 0 ] \* tempArray[ 0 ] +  
 fC[ yPhase, 1 ] \* tempArray[ 1 ] +  
 fC[ yPhase, 2 ] \* tempArray[ 2 ] + (H‑62)  
 fC[ yPhase, 3 ] \* tempArray[ 3 ] + offset ) >> shift2

rsChromaSample = Clip3( 0, ( 1 << BitDepthC ) − 1 , rsChromaSample ) (H‑63)

* + - * 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 ( CurPicWidthInSamplesY ) x ( CurPicHeightInSamplesY ) array rsPredMode specifies the prediction modes of the resampled picture,

– two ( CurPicWidthInSamplesY ) x ( CurPicHeightInSamplesY ) arrays rsRefIdxLX specify the reference indexes of the resampled picture, with X = 0,1,

– two ( CurPicWidthInSamplesY ) x ( CurPicHeightInSamplesY ) arrays rsMvLX specify the luma motion vectors of the resampled picture, with X = 0,1,

– two ( CurPicWidthInSamplesY ) x ( CurPicHeightInSamplesY ) arrays rsPredFlagLX 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 ... ( ( CurPicWidthInSamplesY + 15 ) >> 4 ) − 1 and yPb = 0 … ( ( CurPicHeightInSamplesY + 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 rsPredMode[ xP ][ yP ], rsRefIdxLX[ xP ][ yP ], rsMvLX[ xP ][ yP ] and rsPredFlagLX[ 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 sample location ( xP, yP ), rlPredMode, rlRefIdxLX, rlMvLX and rlPredFlagLX, with X = 0,1, given as inputs.

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 rsPredMode[ xP ][ yP ],

– two derived motion vectors rsMvL0 and rsMvL1[ xP ][ yP ],

– two derived reference indices rsRefIdxL0 and rsRefIdxL1[ xP ][ yP ],

– two derived prediction list utilization flags rsPredFlagL0 and rsPredFlagL1[ xP ][ yP ].

rsPredMode[ xP ][ yP ], rsMvLX[ xP ][ yP ], rsRefIdxLX[ xP ][ yP ], and rsPredFlagLX[ xP ][ yP ], with X = 0, 1, are derived by the following ordered steps:

1. The center location ( xPCtr, yPCtr ) of the luma prediction block is derived as follows:

xPCtr = xP + 8 (H‑64)  
yPCtr = yP + 8 (H‑65)

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 the reference layer luma sample location ( xRef, yRef ) as output.
2. The rounded reference layer luma sample location ( xRL, yRL ) is derived as follows:

xRL = ( ( xRef + 4 ) >> 4 ) << 4 (H‑66)  
yRL = ( ( yRef + 4 ) >> 4 ) << 4 (H‑67)

1. The prediction mode predMode[ xP ][ yP ] is derived as follows:

– If ( xRL < 0 ) or ( xRL >= RefLayerPicWidthInSamplesY ) or ( yRL < 0 ) or ( yRL >= RefLayerPicHeightInSamplesY ),

rsPredMode[ xP ][ yP ] = MODE\_INTRA (H‑68)

– Otherwise, the following applies:

rsPredMode[ xP ][ yP ] = rlPredMode[ xRL ][ yRL ] (H‑69)

1. rsMvL0[ xP ][ yP ], rsMvL1[ xP ][ yP ], rsRefIdxL0[ xP ][ yP ], rsRefIdxL1[ xP ][ yP ], rsPredFlagL0[ xP ][ yP ] and rsPredFlagL1[ xP ][ yP ] are derived as follows:

– If rsPredMode[ xP ][ yP ] is equal to MODE\_INTER, the following applies:

– For each X = 0, 1, rsRefIdxLX[ xP ][ yP ] and rsPredFlagLX[ xP ][ yP ] are derived as follows:

rsRefIdxLX[ xP ][ yP ] = rlRefIdxLX[ xRL ][ yRL ] (H‑70)  
rsPredFlagLX[ xP ][ yP ] = rlPredFlagLX[ xRL ][ yRL ] (H‑71)

– For each X = 0, 1, rsMvLX[ xP ][ yP ][ 0 ] is derived as follows:

– If ScaledRefLayerPicWidthInSamplesY is not equal to RefLayerCroppedPicWidthInSamplesY, the following applies:

scaleFactorMVX = Clip3( −4096, 4095, ( ( ScaledRefLayerPicWidthInSamplesY << 8 )  
 + ( RefLayerCroppedPicWidthInSamplesY >> 1 ) ) /   
 RefLayerCroppedPicWidthInSamplesY ) (H‑72)

rsMvLX[ xP ][ yP ][0] = Clip3( −32768, 32767, Sign(scaleFactorMVX \*  
 rlMvLX[ xRL ][ yRL ][ 0 ] ) \* ( ( Abs ( scaleFactorMVX \* rlMvLX[ xRL ][ yRL ][ 0 ] )  
 + 127 ) >> 8 ) ) (H‑73)

– Otherwise, the following applies:

rsMvLX[ xP ][ yP ][ 0 ] = rlMvLX[ xRL ][ yRL ][ 0 ] (H‑74)

– For each X = 0, 1, rsMvLX[ xP ][ yP ][ 1 ] is derived as follows:

– If ScaledRefLayerPicHeightInSamplesY is not equal to RefLayerCroppedPicHeightInSamplesY, the following applies:

scaleFactorMVY = Clip3( −4096, 4095, ( ( ScaledRefLayerPicHeightInSamplesY << 8 )  
 + ( RefLayerCroppedPicHeightInSamplesY >> 1 ) ) /   
 RefLayerCroppedPicHeightInSamplesY ) (H‑75)

rsMvLX[ xP ][ yP ][ 1 ] = Clip3( −32768, 32767, Sign(scaleFactorMVY \*   
 rlMvLX[ xRL ][ yRL ][ 1 ] ) \* ( ( Abs ( scaleFactorMVY \* rlMvLX[ xRL ][ yRL ][ 1 ] )  
 + 127 ) >> 8 ) ) (H‑76)

– Otherwise, the following applies:

rsMvLX[ xP ][ yP ][ 1 ] = rlMvLX[ xRL ][ yRL ][ 1 ] (H‑77)

– Otherwise (rsPredMode[ xP ][ yP ] is equal to MODE\_INTRA), the following applies:

– Both components of rsMvL0[ xP ][ yP ] and rsMvL1[ xP ][ yP ] are set to 0, rsRefIdxL0[ xP ][ yP ] and rsRefIdxL1[ xP ][ yP ] are set to −1, rsPredFlagL0[ xP ][ yP ] and rsPredFlagL1[ xP ][ yP ] are set to 0.

* + - * 1. Colour mapping 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 ( RefLayerPicWidthInSamplesY ) x ( RefLayerPicHeightInSamplesY ) array cmPicSampleL of luma samples,

– a ( RefLayerPicWidthInSamplesC ) x ( RefLayerPicHeightInSamplesC ) array cmPicSampleCb of chroma samples of the component Cb,

– a ( RefLayerPicWidthInSamplesC ) x ( RefLayerPicHeightInSamplesC ) array cmPicSampleCr of chroma samples of the component Cr.

The following applies:

– The luma samples cmPicSampleL[ xP ][ yP ] with ( xP = 0 ... RefLayerPicWidthInSamplesY − 1, yP = 0 ... RefLayerPicHeightInSamplesY − 1) are derived by invoking the colour mapping process of luma sample values as specified in subclause H.8.1.4.3.1 with luma sample location ( xP, yP ), sample arrays rlPicSampleL, rlPicSampleCb and rlPicSampleCr given as inputs and cmPicSampleL[ xP ][ yP ] as output.

– The chroma samples cmPicSampleCb[ xC ][ yC ] with ( xC = 0 ... RefLayerPicWidthInSamplesC − 1, yC = 0 ... RefLayerPicHeightInSamplesC − 1) are derived by invoking the colour mapping process of chroma sample values as specified in subclause H.8.1.4.3.2 with chroma sample location ( xC, yC ), sample arrays rlPicSampleL, rlPicSampleCb rlPicSampleCr, and cIdx equal to 0 given as inputs and cmPicSampleCb[ xC ][ yC ] as output.

– The chroma samples cmPicSampleCr[ xC ][ yC ] with ( xC = 0 ... RefLayerPicWidthInSamplesC − 1, yC = 0 ... RefLayerPicHeightInSamplesC − 1) are derived by invoking the colour mapping process of chroma sample values as specified in subclause H.8.1.4.3.2 with chroma sample location ( xC, yC ), sample arrays rlPicSampleL, rlPicSampleCb, rlPicSampleCr, and cIdx equal to 1 given as inputs and cmPicSampleCr[ xC ][ yC ] as output.

Colour mapping process of luma sample values

Inputs to this process are

– a luma sample location ( xP, yP ) specifying the luma sample location relative to the top-left luma sample of the reference layer picture,

– the luma reference layer sample array rlPicSampleY

– the chroma reference layer sample array rlPicSampleCb of the Cb component

– the chroma reference layer sample array rlPicSampleCr of the Cr component

Output of the process is a colour mapped luma sample value cmLumaSample

The chroma sample location ( xPC, yPC ) is set equal to ( xP >> 1, yP >> 1 ).

The value of cmLumaSample is derived by applying the following ordered steps:

1. The variables yShift2Idx, cShift2Idx are derived as follows.

yShift2Idx = CMInputBitDepthY − cm\_octant\_depth − cm\_y\_part\_num\_log2 (H‑86)

cShift2Idx = CMInputBitDepthC − cm\_octant\_depth (H‑87)

1. The variables nMappingShift, and nMappingOffset are derived as follows.

nMappingShift = 10 + CMInputBitDepthY − CMOutputBitDepthY (H‑88)

nMappingOffset = 1 << (nMappingShift − 1) (H‑89)

1. The following applies:

– If xP % 2 is equal to 0 and yP % 2 is equal to 0:

yP2C = Max( 0 , yPC − 1 )

uTemp = ( rlPicSampleCb[ xPC ][ yPC ] \* 3) + rlPicSampleCb[ xPC ][ yP2C ] ) + 2 ) >> 2 (H‑90)

vTemp = ( rlPicSampleCr[ xPC ][ yPC ] \* 3) + rlPicSampleCr[ xPC ][ yP2C ] ) + 2 ) >> 2 (H‑91)

– Otherwise, if xP % 2 is equal to 0 and yP % 2 is equal to 1:

yP2C = Min( yPC + 1, RefLayerPicHeightInSamplesC − 1 )

uTemp = ( rlPicSampleCb[ xPC ][ yPC ] \* 3) + rlPicSampleCb[ xPC ][ yP2C ] + 2 ) >> 2 (H‑92)

vTemp = ( rlPicSampleCr[ xPC ][ yPC ] \* 3) + rlPicSampleCr[ xPC ][ yP2C ] + 2 ) >> 2 (H‑93)

– Otherwise, if xP % 2 is equal to 1 and yP % 2 is equal to 0:

xP2C = Min( xPC + 1, RefLayerPicWidthInSamplesC − 1 )

yP2C = Max( 0, yPC − 1 )

uTemp = ( rlPicSampleCb[ xPC ][ yP2C ] + rlPicSampleCb[ xP2C ][ yP2C ] + ( rlPicSampleCb[ xPC][ yPC ] + rlPicSampleCb[ xP2C ][ yPC ] ) \* 3 + 4 ) >> 3 (H‑94)

vTemp = ( rlPicSampleCr[ xPC ][ yP2C ] + rlPicSampleCr[ xP2C ][ yP2C ] + ( rlPicSampleCr[ xPC ][ yPC ] + rlPicSampleCr[ xP2C ][ yPC ] ) \* 3 + 4 ) >> 3 (H‑95)

– Otherwise, (if xP % 2 is equal to 1 and yP % 2 is equal to 1)

xP2C = Min( xPC + 1, RefLayerPicWidthInSamplesC − 1 )

yP2C = Min( yPC + 1, RefLayerPicHeightInSamplesC − 1 )

uTemp = ( ( rlPicSampleCb[ xPC ][ yPC ] + rlPicSampleCb[ xP2C ][ yPC ]) \* 3 + rlPicSampleCb[ xPC ][ yP2C ] + rlPicSampleCb[ xP2C ][ yP2C ] + 4 ) >> 3 (H‑96)

vTemp = ( ( rlPicSampleCr[ xPC ][ yPC ] + rlPicSampleCr[ xP2C ][ yPC ] ) \* 3 + rlPicSampleCr[ xPC ][ yP2C ] + rlPicSampleCr[ xP2C ][ yP2C ] + 4 ) >> 3 (H‑97)

1. The value of cmLumaSample is derived as follows:

yIdx = rlPicSampleY[ xP ][ yP ] >> yShift2Idx (H‑98)

uIdx = ( cm\_octant\_depth = = 1 ) ? ( uTemp >= CMThreshU ) : ( uTemp >> cShift2Idx ) (H‑99)

vIdx = ( cm\_octant\_depth = = 1 ) ? ( vTemp >= CMThreshV ) : ( vTemp >> cShift2Idx ) (H‑100)

cmLumaSample =   
( LutY[ yIdx ][ uIdx ][ vIdx ][ 0 ] \* rlPicSampleY[ xP ][ yP ]  
 + LutY[ yIdx ][ uIdx ][ vIdx ][ 1 ] \* uTemp + LutY[ yIdx ][ uIdx ][ vIdx ][ 2 ] \* vTemp  
 + nMappingOffset ) >> nMappingShift ) + LutY[ yIdx ][ uIdx ][ vIdx ][ 3 ] (H‑104)

cmLumaSample = Clip3(0, (1 << CMOutputBitDepthY ) − 1, cmLumaSample ) (H‑105)

Colour mapping process of chroma sample values

Inputs to this process are

– a chroma sample location ( xPC, yPC ) specifying the chroma sample location relative to the top-left chroma sample of the reference layer picture,

– the luma reference sample array rlPicSampleY,

– the chroma reference sample array rlPicSampleCb of the Cb component,

– the chroma reference sample array rlPicSampleCr of the Cr component,

– a variable cIdx specifying the chroma component index.

Output of the process is a colour mapped chorma sample value cmChromaSample.

The luma sample location ( xP, yP ) is set equal to ( xPC << 1, yPC << 1 ).

The colour mapping table LutC is set to LutU if cIdx is equal to 0, and set to LutV otherwise.

The value of cmChromaSample is derived by applying the following ordered steps:

1. The variables yShift2Idx, cShift2Idx are derived as follows.

yShift2Idx = CMInputBitDepthY − cm\_octant\_depth − cm\_y\_part\_num\_log2 (H‑106)

cShift2Idx = CMInputBitDepthC − cm\_octant\_depth (H‑107)

1. The variables nMappingShift, and nMappingOffset are derived as follows.

nMappingShift = yShift2Idx + cShift2Idx (H‑108)

nMappingOffset = 1 << (nMappingShift − 1) (H‑109)

1. The variable yTemp is derived as follows:

yTemp = ( rlPicSampleY[ xP ][ yP ] + rlPicSampleY[ xP ][ yP + 1 ] + 1 ) >> 1 (H‑110)

1. The value of cmChromaSample is derived as follows:

yIdx = yTemp >> yShift2Idx (H‑111)

uIdx = ( cm\_octant\_depth = = 1 ) ?   
( rlPicSampleCb[ xPC ][ yPC ] > = CMThreshU ) : ( rlPicSampleCb[ xPC ][ yPC ] >> cShift2Idx ) (H‑112)

vIdx = ( cm\_octant\_depth = = 1 ) ? ( rlPicSampleCr[ xPC ][ yPC ] > = CMThreshV ) : ( rlPicSampleCr[ xPC ][ yPC ] >> cShift2Idx ) (H‑113)

cmChromaSample = ( LutC[ yIdx ][ uIdx ][ vIdx ][ 0 ] \* yTemp  
 + LutC[ yIdx ][ uIdx ][ vIdx ][ 1 ] \* rlPicSampleCb[ xPC ][ yPC ]  
 + LutC[ yIdx ][ uIdx ][ vIdx ][ 2 ] \* rlPicSampleCr[ xPC ][ yPC ]  
 + nMappingOffset ) >> nMappingShift ) + LutC[ yIdx ][ uIdx ][ vIdx ][ 3 ] (H‑117)

cmChromaSample = Clip3(0, ( 1 << CMOutputBitDepthC ) − 1, cmChromaSample ) (H‑118)

* + 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

The specifications in subclause F.8.3.4 apply.

NOTE – Because bitstreams conforming to this Annex are constrained to allow only zero-valued motion vectors for inter prediction using inter layer reference pictures, it is suggested that a scalable encoder should disable temporal motion vector prediction for the current picture (by setting slice\_temporal\_mvp\_enabled\_flag to zero) when the reference picture lists of all slices in the current picture include only inter-layer reference pictures. This way, the encoder would be able to avoid the need to send the slice segment header syntax elements collocated\_from\_l0\_flag and collocated\_ref\_idx.

* + 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 additions.

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 inter-layer reference picture. 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 inter-layer reference picture.

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.

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 and Scalable Main 10 profiles

Bitstreams containing OLSs conforming to the Scalable Main or Scalable Main 10 profiles shall obey the following constraints on a derived sub-bitstream for each of the OLSs indicated to conform to the Scalable Main or Scalable Main 10 profile, respectively, with layerSetIdx being the layer set for the OLS, and the sub-bitstream being derived by invoking the sub-bitstream extraction process as specified in subclause F.10 with tIdTarget equal to 6 and with layerIdListTarget containing the nuh\_layer\_id values of the layers that are included in the layer set with the index layerSetIdx and are primary picture layers.

Bitstreams containing OLSs conforming to the Scalable Main or Scalable Main 10 profiles shall also obey the following constraints on the base layer bitstream derived by invoking the sub-bitstream extraction process as specified in subclause F.10 with tIdTarget equal to 6 and with layerIdListTarget containing only one nuh\_layer\_id value that is equal to 0 as inputs.

The base layer bitstream derived from bitstreams conforming to the Scalable Main profile 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 shall have general\_profile\_idc equal to 1 or general\_profile\_compatibility\_flag[ 1 ] equal to 1.

The base layer bitstream derived from bitstreams conforming to the Scalable Main 10 profile shall obey the following constraints:

– The base layer bitstream shall obey all constraints of the Main 10 profile specified in subclause A.3.3.

– SPSs of the base layer bitstream shall have general\_profile\_idc equal to 1 or 2, or general\_profile\_compatibility\_flag[ 1 ] or general\_profile\_compatibility\_flag[ 2 ] equal to 1.

The derived sub-bitstream for an OLS conforming to the Scalable Main or Scalable Main 10 profiles shall obey the following constraints:

– All active SPSs for the sub-bitstream shall have chroma\_format\_idc equal to 1 only.

– CtbLog2SizeY derived from any active SPS for the sub-bitstream shall be in the range of 4 to 6, inclusive.

– The variables NumResampling, NumSampleResampling, and NumMotionResampling shall be less than or equal to 1 for each decoded picture with nuh\_layer\_id included in layerIdListTarget that was used to derive the sub-bitstream.

– ScalabilityId[ j ][ smIdx ] shall be equal to 0 for any smIdx value not equal to 2 and for any value of j such that layer\_id\_in\_nuh[ j ] is among layerIdListTarget that was used to derive the sub-bitstream.

– For a layer with layer id iNuhLId equal to any of nuh\_layer\_id included in layerIdListTarget that was used to derive the sub-bitstream, 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.

– All active PPSs shall have pps\_extension\_type\_flag[ i ] equal to 0 only for i in the range of 0 to 6, inclusive.

– DependencyId[ i ] shall not be equal to DependencyId[ j ] for any values of i and j among layerIdListTarget that was used to derive the sub-bitstream such that i is not equal to j.

The derived sub-bitstream for an OLS conforming to the Scalable Main profile shall obey the following constraints:

– All active SPSs for the sub-bitstream shall have bit\_depth\_luma\_minus8 equal to 0 only.

– All active SPSs for the sub-bitstream shall have bit\_depth\_chroma\_minus8 equal to 0 only.

– All active PPSs for the sub-bitstream shall have colour\_mapping\_enabled\_flag equal to 0 only.

The derived sub-bitstream for an OLS conforming to the Scalable Main 10 profile shall obey the following constraints:

– All active SPSs for the sub-bitstream shall have bit\_depth\_luma\_minus8 in the range of 0 to 2, inclusive.

– All active SPSs for the sub-bitstream shall have bit\_depth\_chroma\_minus8 in the range of 0 to 2, inclusive.

* + 1. Tiers and levels
       1. Profile specific tier and level limits for the Scalable Main and Scalable Main 10 profiles

Bitstreams containing OLSs conforming to the Scalable Main or Scalable Main 10 profiles shall obey the following constraints on a derived sub-bitstream for the OLS, with layerSetIdx being the layer set for the OLS conforming to the Scalable Main or Scalable Main 10 profile, respectively, derived by invoking the sub-bitstream extraction process as specified in subclause F.10 with tIdTarget equal to 6 and with TargetDecLayerIdList containing the nuh\_layer\_id values of the layer set with the index layerSetIdx:

1. Each layer in the TargetDecLayerIdList shall obey the General tier and level limits in A.4.1, and the Profile-specific level limits a), b), c), d), g), h), i), and j) for the Main and Main 10 profiles specified in A.4.2.
2. The value of TotalPicSizeInSamplesY shall be less than or equal to 2 \* MaxLumaPs, where MaxLumaPs is specified in Table A‑1, and where TotalPicSizeInSamplesY is derived as follows:

TotalPicSizeInSamplesY = 0

for (i = 0; i <= 62; i++)

if layer i in TargetDecLayerIdList

TotalPicSizeInSamplesY += PicSizeInSamplesY of the layer with nuh\_layer\_id equal to i

1. The nominal removal time of access unit n (with n greater than 0) from the CPB, as specified in subclause C.2.3, shall satisfy the constraint that AuNominalRemovalTime[ n ] − AuCpbRemovalTime[ n − 1 ] is greater than or equal to Max( TotalPicSizeInSamplesY ÷ (2 \* MaxLumaSr), fR ) for the value of TotalPicSizeInSamplesY of access unit n − 1, where MaxLumaSr is the value specified in Table A‑2 that applies to access unit n − 1.
2. For the VCL HRD parameters, BitRate[ i ] shall be less than or equal to 2 \* CpbBrVclFactor \* MaxBR for at least one value of i in the range of 0 to cpb\_cnt\_minus1[ HighestTid ], inclusive, where BitRate[ i ] is specified in subclause E.2.3 based on parameters selected as specified in subclause C.1 and MaxBR is specified in Table A‑2 in units of CpbBrVclFactor bits/s.
3. For the NAL HRD parameters, BitRate[ i ] shall be less than or equal to 2 \* CpbBrNalFactor \* MaxBR for at least one value of i in the range of 0 to cpb\_cnt\_minus1[ HighestTid ], inclusive, where BitRate[ i ] is specified in subclause E.2.3 based on parameters selected as specified in subclause C.1 and MaxBR is specified in Table A‑2 in units of CpbBrNalFactor bits/s.
   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.