|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  14th Meeting: Vienna, AT, 25 July – 2 Aug. 2013 | Document: JCTVC-N1008\_v3r1 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **High efficiency video coding (HEVC) scalable extension draft 3** | | |
| *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 | 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) |
| *Source:* | Editors | | |

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

# Abstract

This document contains draft 3 text of high efficiency video coding (HEVC) scalable extension.

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.

The modifications related to using higher layer picture as a reference for the lower layer are highlighted in this document. In revision 1 of this proposal, a bitstream restriction is proposed (highlighted in green) on the included BL reference picture in the RPL.**CONTENTS**

*Page*

[Abstract i](#_Toc366771928)

[8 Decoding process 1](#_Toc366771929)

[8.1 General decoding process 1](#_Toc366771930)

[8.1.1 Decoding process for a coded picture with nuh\_layer\_id equal to 0 2](#_Toc366771931)

[8.2 NAL unit decoding process 3](#_Toc366771932)

[8.3 Slice decoding process 3](#_Toc366771933)

[8.3.3 Decoding process for generating unavailable reference pictures 3](#_Toc366771935)

[8.3.3.1 General decoding process for generating unavailable reference pictures 3](#_Toc366771936)

[Annex C Hypothetical reference decoder 4](#_Toc366771937)

[C.4 Bitstream conformance 4](#_Toc366771938)

[Annex F Common syntax, semantics and decoding processes for multi-layer video coding extensions 5](#_Toc366771939)

[F.1 Scope 5](#_Toc366771940)

[F.2 Normative references 5](#_Toc366771941)

[F.3 Definitions 5](#_Toc366771942)

[F.4 Abbreviations 6](#_Toc366771943)

[F.5 Conventions 6](#_Toc366771944)

[F.6 Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships 6](#_Toc366771945)

[F.7 Syntax and semantics 6](#_Toc366771946)

[F.7.1 Method of specifying syntax in tabular form 6](#_Toc366771947)

[F.7.2 Specification of syntax functions, categories, and descriptors 6](#_Toc366771948)

[F.7.3 Syntax in tabular form 6](#_Toc366771949)

[F.7.3.1 NAL unit syntax 6](#_Toc366771950)

[F.7.3.2 Raw byte sequence payloads and RBSP trailing bits syntax 7](#_Toc366771951)

[F.7.3.3 Profile, tier and level syntax 18](#_Toc366771952)

[F.7.3.4 Scaling list data syntax 18](#_Toc366771953)

[F.7.3.5 Supplemental enhancement information message syntax 18](#_Toc366771954)

[F.7.3.6 Slice segment header syntax 19](#_Toc366771955)

[F.7.3.7 Short-term reference picture set syntax 21](#_Toc366771956)

[F.7.3.8 Slice segment data syntax 21](#_Toc366771957)

[F.7.4 Semantics 22](#_Toc366771958)

[F.7.4.1 General 22](#_Toc366771959)

[F.7.4.2 NAL unit semantics 22](#_Toc366771960)

[F.7.4.3 Raw byte sequence payloads, trailing bits, and byte alignment semantics 24](#_Toc366771961)

[F.7.4.4 Profile, tier and level semantics 35](#_Toc366771962)

[F.7.4.5 Scaling list data semantics 36](#_Toc366771963)

[F.7.4.6 Supplemental enhancement information message semantics 36](#_Toc366771964)

[F.7.4.7 Slice segment header semantics 36](#_Toc366771965)

[F.7.4.8 Short-term reference picture set semantics 38](#_Toc366771966)

[F.7.4.9 Slice segment data semantics 38](#_Toc366771967)

[F.8 Decoding process 38](#_Toc366771968)

[F.8.1 General decoding process 38](#_Toc366771969)

[F.8.1.1 Decoding process for starting the decoding of a coded picture with nuh\_layer\_id greater than 0 39](#_Toc366771970)

[F.8.1.2 Decoding process for ending the decoding of a coded picture with nuh\_layer\_id greater than 0 39](#_Toc366771971)

[F.8.1.3 Generation of unavailable reference pictures for pictures first in decoding order within a layer 40](#_Toc366771972)

[F.8.2 NAL unit decoding process 41](#_Toc366771973)

[F.8.3 Slice decoding processes 41](#_Toc366771974)

[F.8.3.1 Decoding process for picture order count 41](#_Toc366771975)

[F.8.3.2 Decoding process for reference picture set 42](#_Toc366771976)

[F.8.4 Decoding process for coding units coded in intra prediction mode 46](#_Toc366771977)

[F.8.5 Decoding process for coding units coded in inter prediction mode 46](#_Toc366771978)

[F.8.6 Scaling, transformation and array construction process prior to deblocking filter process 46](#_Toc366771979)

[F.8.7 In-loop filter process 46](#_Toc366771980)

[F.9 Parsing process 46](#_Toc366771981)

[F.10 Specification of bitstream subsets 46](#_Toc366771982)

[F.11 (Void) 46](#_Toc366771983)

[F.12 Byte stream format 46](#_Toc366771984)

[F.13 Hypothetical reference decoder 46](#_Toc366771985)

[F.13.1 General 46](#_Toc366771986)

[F.13.2 Operation of coded picture buffer (CPB) 46](#_Toc366771987)

[F.13.3 Operation of the decoded picture buffer (DPB) 47](#_Toc366771988)

[F.13.3.1 General 47](#_Toc366771989)

[F.13.3.2 Removal of pictures from the DPB 47](#_Toc366771990)

[F.13.3.3 Picture output 47](#_Toc366771991)

[F.13.3.4 Current decoded picture marking and storage 47](#_Toc366771992)

[F.13.4 Bitstream conformance 47](#_Toc366771993)

[F.13.5 Decoder conformance 49](#_Toc366771994)

[F.13.5.1 General 49](#_Toc366771995)

[F.13.5.2 Operation of the output order DPB 49](#_Toc366771996)

[F.14 SEI messages 51](#_Toc366771997)

[F.14.1 SEI message syntax 51](#_Toc366771999)

[F.14.1.1 Layers not present SEI message syntax 51](#_Toc366772000)

[F.14.1.2 Inter-layer constrained tile sets SEI message syntax 51](#_Toc366772001)

[F.14.2 SEI message semantics 52](#_Toc366772002)

[F.14.2.1 Layers not present SEI message semantics 52](#_Toc366772003)

[F.14.2.2 Inter-layer constrained tile sets SEI message semantics 52](#_Toc366772004)

[F.15 Video usability information 54](#_Toc366772005)

[F.15.1 General 54](#_Toc366772006)

[F.15.2 VUI syntax 54](#_Toc366772007)

[F.15.3 VUI semantics 54](#_Toc366772008)

[F.15.3.1 VUI parameters semantics 54](#_Toc366772009)

[F.15.3.2 HRD parameters semantics 54](#_Toc366772010)

[F.15.3.3 Sub-layer HRD parameters semantics 54](#_Toc366772011)

[Annex H Syntax, semantics and decoding processes for scalable extension 55](#_Toc366772012)

[H.1 Scope 55](#_Toc366772013)

[H.2 Normative references 55](#_Toc366772014)

[H.3 Definitions 55](#_Toc366772015)

[H.4 Abbreviations 55](#_Toc366772016)

[H.5 Conventions 55](#_Toc366772017)

[H.6 Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships 55](#_Toc366772018)

[H.6.1 Derivation process for reference layer sample location 55](#_Toc366772019)

[H.6.2 Derivation process for reference layer sample location used in resampling 55](#_Toc366772020)

[H.7 Syntax and semantics 56](#_Toc366772021)

[H.8 Decoding processes 56](#_Toc366772022)

[H.8.1 General decoding process 56](#_Toc366772023)

[H.8.1.1 Decoding process for a coded picture with nuh\_layer\_id greater than 0 56](#_Toc366772024)

[H.8.1.2 Decoding process for inter-layer reference picture set 56](#_Toc366772025)

[H.8.1.3 Marking process for ending the decoding of a coded picture with nuh\_layer\_id greater than 0 57](#_Toc366772026)

[H.8.1.4 Resampling process for inter layer reference pictures 57](#_Toc366772027)

[H.8.2 NAL unit decoding process 64](#_Toc366772028)

[H.8.3 Slice decoding processes 64](#_Toc366772029)

[H.8.3.1 Decoding process for picture order count 64](#_Toc366772030)

[H.8.3.2 Decoding process for reference picture set 64](#_Toc366772031)

[H.8.3.3 Decoding process for generating unavailable reference pictures 64](#_Toc366772032)

[H.8.3.4 Decoding process for reference picture lists construction 64](#_Toc366772033)

[H.8.4 Decoding process for coding units coded in intra prediction mode 65](#_Toc366772034)

[H.8.5 Decoding process for coding units coded in inter prediction mode 65](#_Toc366772035)

[H.8.6 Scaling, transformation and array construction process prior to deblocking filter process 65](#_Toc366772036)

[H.8.7 In-loop filter process 65](#_Toc366772037)

[H.9 Parsing process 65](#_Toc366772038)

[H.10 Specification of bitstream subsets 65](#_Toc366772039)

[H.11 Profiles, tiers, and levels 66](#_Toc366772040)

[H.11.1 Profiles 66](#_Toc366772041)

[H.11.1.1 General 66](#_Toc366772042)

[H.11.1.2 Scalable Main profile 66](#_Toc366772043)

[H.11.2 Tiers and levels 66](#_Toc366772044)

[H.12 Byte stream format 66](#_Toc366772045)

[H.13 Hypothetical reference decoder 66](#_Toc366772046)

[H.14 SEI messages 66](#_Toc366772047)

[H.15 Video usability information 66](#_Toc366772048)

*Replace subclause 8.1, 8.2 and 8.3.1 with the following and add subclause 8.1.1 (with differenceindicated in turquoise ).*

# Decoding process

## General decoding process

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

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, is specified as follows:

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

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

– Otherwise, TargetDecLayerIdList contains only one nuh\_layer\_id value that is equal to 0.

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 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, 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 LayerInitialisedFlag[ i ] is set equal to 0 for all values of i from 0 to 63, inclusive, and the variable FirstPicInLayerDecodedFlag[ i ] is set equal to 0 for all values of i from 1 to 63, 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 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 and the variable NoRaslOutputFlag is set equal to HandleCraAsBlaFlag.

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

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

– nuh\_layer\_id is equal to 0.

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

When the current picture 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

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

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

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

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

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

– Otherwise (separate\_colour\_plane\_flag is equal to 1), the 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).

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

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

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

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

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

– PicOutputFlag is set as follows:

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

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

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

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

## NAL unit decoding process

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

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

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

## Slice decoding process

### 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.

*Modify clause C.4 as follows:*

1. Annex C  
     
   Hypothetical reference decoder

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

* 1. Bitstream conformance

*Replace the paragraph*

"The first coded picture in a bitstream shall be an IRAP picture, i.e. an IDR picture, a CRA picture or a BLA picture".

*with*

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

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

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

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

* 1. Scope

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

* 1. Normative references

The specifications in clause 2 apply.

* 1. Definitions

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

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

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

1. associated IRAP picture: The previous *IRAP picture* in *decoding order* within the same *layer* (if present).
2. base layer: A *layer* in which all *VCL NAL units* have nuh\_layer\_id equal to 0.
3. 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*.
4. 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*.
5. collocated sample: A sample TBD.
6. direct reference layer: A *layer* that may be used for inter-layer prediction of another *layer*.
7. indirect reference layer: A *layer* that is not a *direct reference layer* of another *layer* but is a *direct reference layer* of a *layer* that is a *direct reference layer* or indirect reference *layer* of a *direct reference layer* of the *layer*.
8. 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.
9. 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.*
10. 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*.
11. layer initialisation picture (LIP): A *picture* that is an IRAP picture with NoRaslOutputFlag equal to 1 or that is contained in an *initial IRAP access unit*.
12. leading picture: A *picture* that is in the same *layer* as the *associated IRAP picture* and precedes the *associated IRAP picture* in *output order*.
13. non-base layer: A *layer* in which all *VCL NAL units* have the same nuh\_layer\_id value greater than 0.
14. picture order count: A variable that is associated with each *picture* and that uniquely identifies the associated *picture* among all *pictures* with the same value of nuh\_layer\_id in the *CVS*, and, when the associated *picture* is to be output from the *decoded picture buffer*, indicates the position of the associated *picture* in *output order* relative to the *output order* positions of the other *pictures* with the same value of nuh\_layer\_id in the same *CVS* that are to be output from the *decoded picture buffer*.
15. 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*.
16. reference picture list: A list of reference pictures that is used for inter prediction or inter-layer prediction of a P or B slice.
17. target output layer: A *layer* that is to be output.
18. trailing picture: A *picture* that is in the same *layer* as the *associated IRAP picture* and follows the *associated IRAP picture* in *output order*.
19. 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*.
20. 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. Abbreviations

The specifications in clause 4 apply.

* 1. Conventions

The specifications in clause 5 apply.

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

The specifications in clause 6 apply.

* 1. Syntax and semantics

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

* + 1. Method of specifying syntax in tabular form

The specifications in subclause 7.1 apply.

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

The specifications in subclause 7.2 apply.

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

The specifications in subclause 7.3.1 apply.

* + - * 1. General NAL unit syntax

The specifications in subclause 7.3.1.1 apply.

* + - * 1. NAL unit header syntax

The specifications in subclause 7.3.1.2 apply.

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

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

Video parameter set extension syntax

|  |  |
| --- | --- |
| vps\_extension( ) { | Descriptor |
| **avc\_base\_layer\_flag** | u(1) |
| **vps\_vui\_offset** | u(16) |
| **splitting\_flag** | u(1) |
| for( i = 0, NumScalabilityTypes = 0; i < 16; i++ ) { |  |
| **scalability\_mask\_flag**[ i ] | u(1) |
| NumScalabilityTypes += scalability\_mask\_flag[ i ] |  |
| } |  |
| for( j = 0; j < ( NumScalabilityTypes − splitting\_flag ); j++ ) |  |
| **dimension\_id\_len\_minus1**[ j ] | u(3) |
| **vps\_nuh\_layer\_id\_present\_flag** | u(1) |
| for( i = 1; i <= vps\_max\_layers\_minus1; 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) |
| } |  |
| if( NumViews > 1 ) |  |
| **view\_id\_len\_minus1** | u(4) |
| for( i = 0; i < NumViews; i++ ) |  |
| **view\_id\_val**[ i ] | u(v) |
| for( i = 1; i <= vps\_max\_layers\_minus1; i++ ) |  |
| for( j = 0; j < i; j++ ) |  |
| **direct\_dependency\_flag**[ i ][ j ] | u(1) |
| **max\_tid\_ref\_present\_flag** | u(1) |
| if( max\_tid\_ref\_present\_flag ) |  |
| for( i = 0; i < vps\_max\_layers\_minus1; i++ ) |  |
| **max\_tid\_il\_ref\_pics\_plus1**[ i ] | u(3) |
| **all\_ref\_layers\_active\_flag** | u(1) |
| **vps\_number\_layer\_sets\_minus1** | u(10) |
| **vps\_num\_profile\_tier\_level\_minus1** | u(6) |
| for( i = 1; i <= vps\_num\_profile\_tier\_level\_minus1; i ++ ) { |  |
| **vps\_profile\_present\_flag**[ i ] | u(1) |
| if( !vps\_profile\_present\_flag[ i ] ) |  |
| **profile\_ref\_minus1**[ i ] | u(6) |
| profile\_tier\_level( vps\_profile\_present\_flag[ i ], vps\_max\_sub\_layers\_minus1 ) |  |
| } |  |
| numOutputLayerSets = vps\_number\_layer\_sets\_minus1 + 1 |  |
| **more\_output\_layer\_sets\_than\_default\_flag** | u(1) |
| if( more\_output\_layer\_sets\_than\_default\_flag ) { |  |
| **num\_add\_output\_layer\_sets\_minus1** | u(10) |
| numOutputLayerSets += num\_add\_output\_layer\_sets\_minus1 + 1 |  |
| } |  |
| if( numOutputLayerSets > 1 ) |  |
| **default\_one\_target\_output\_layer\_flag** | u(1) |
| for( i = 1; i < numOutputLayerSets; i++ ) { |  |
| if( i > vps\_number\_layer\_sets\_minus1 ) { |  |
| **output\_layer\_set\_idx\_minus1**[ i ] | u(v) |
| lsIdx = output\_layer\_set\_idx\_minus1[ i ] + 1 |  |
| for( j = 0 ; j < NumLayersInIdList[ lsIdx ] − 1; j++) |  |
| **output\_layer\_flag**[ i ][ j ] | u(1) |
| } |  |
| **profile\_level\_tier\_idx**[ i ] | u(v) |
| } |  |
| **rep\_format\_idx\_present\_flag** | u(1) |
| if( rep\_format\_idx\_present\_flag ) |  |
| **vps\_num\_rep\_formats\_minus1** | u(4) |
| for( i = 0; i <= vps\_num\_rep\_formats\_minus1; i++ ) |  |
| rep\_format( ) |  |
| if( rep\_format\_idx\_present\_flag ) |  |
| for( i = 1; i <= vps\_max\_layers\_minus1; i++ ) |  |
| if( vps\_num\_rep\_formats\_minus1 > 0 ) |  |
| **vps\_rep\_format\_idx**[ i ] | u(4) |
| **max\_one\_active\_ref\_layer\_flag** | u(1) |
| **cross\_layer\_irap\_aligned\_flag** | u(1) |
| **direct\_dep\_type\_len\_minus2** | ue(v) |
| for( i = 1; i <= vps\_max\_layers\_minus1; i++ ) |  |
| for( j = 0; j < i; j++ ) |  |
| if( direct\_dependency\_flag[ i ][ j ] ) |  |
| **direct\_dependency\_type**[ i ][ j ] | u(v) |
| **single\_layer\_for\_non\_irap\_flag** | u(1) |
| for( i = 0; i <= vps\_max\_layers\_minus1 - 1; i++ ) |  |
| enable\_higher\_layer\_ref\_pic\_pred[ i ] | u(1) |
| **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

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

VPS VUI syntax

|  |  |
| --- | --- |
| vps\_vui( ){ | Descriptor |
| **bit\_rate\_present\_vps\_flag** | u(1) |
| **pic\_rate\_present\_vps\_flag** | u(1) |
| if( bit\_rate\_present\_vps\_flag | | pic\_rate\_present\_vps\_flag ) |  |
| for( i = 0; i <= vps\_number\_layer\_sets\_minus1; i++ ) |  |
| for( j = 0; j <= vps\_max\_sub\_layers\_minus1; j++ ) { |  |
| if( bit\_rate\_present\_vps\_flag ) |  |
| **bit\_rate\_present\_flag**[ i ][ j ] | u(1) |
| if( pic\_rate\_present\_vps\_flag ) |  |
| **pic\_rate\_present\_flag**[ i ][ j ] | u(1) |
| if( bit\_rate\_present\_flag[ i ][ j ] ) { |  |
| **avg\_bit\_rate**[ i ][ j ] | u(16) |
| **max\_bit\_rate**[ i ][ j ] | u(16) |
| } |  |
| if( pic\_rate\_present\_flag[ i ][ j ] ) { |  |
| **constant\_pic\_rate\_idc**[ i ][ j ] | u(2) |
| **avg\_pic\_rate**[ i ][ j ] | u(16) |
| } |  |
| } |  |
| for( i = 1; i <= vps\_max\_layers\_minus1; i++ ) |  |
| for( j = 0; j < NumDirectRefLayers[ layer\_id\_in\_nuh[ i ] ]; j++ ) { |  |
| **tile\_boundaries\_aligned\_flag**[ i ][ j ] | u(1) |
| **ilp\_restricted\_ref\_layers\_flag** | u(1) |
| if( ilp\_restricted\_ref\_layers\_flag ) |  |
| for( i = 1; i <= vps\_max\_layers\_minus1; i++ ) |  |
| for( j = 0; j < NumDirectRefLayers[ layer\_id\_in\_nuh[ i ] ]; j++ ) { |  |
| **min\_spatial\_segment\_offset\_plus1**[ i ][ j ] | ue(v) |
| if( min\_spatial\_segment\_offset\_plus1[ i ][ j ] > 0 ) { |  |
| **ctu\_based\_offset\_enabled\_flag**[ i ][ j ] | u(1) |
| if( ctu\_based\_offset\_enabled\_flag[ i ][ j ] ) |  |
| **min\_horizontal\_ctu\_offset\_plus1**[ i ][ j ] | ue(v) |
| } |  |
| } |  |
| } |  |

* + - * 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 ) { |  |
| **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( update\_rep\_format\_flag ) { |  |
| **bit\_depth\_luma\_minus8** | ue(v) |
| **bit\_depth\_chroma\_minus8** | ue(v) |
| **}** |  |
| **log2\_max\_pic\_order\_cnt\_lsb\_minus4** | ue(v) |
| **sps\_sub\_layer\_ordering\_info\_present\_flag** | u(1) |
| for( i = ( sps\_sub\_layer\_ordering\_info\_present\_flag ? 0 : sps\_max\_sub\_layers\_minus1 );  i <= sps\_max\_sub\_layers\_minus1; i++ ) { |  |
| **sps\_max\_dec\_pic\_buffering\_minus1**[ i ] | ue(v) |
| **sps\_max\_num\_reorder\_pics**[ i ] | ue(v) |
| **sps\_max\_latency\_increase\_plus1**[ i ] | ue(v) |
| } |  |
| **log2\_min\_luma\_coding\_block\_size\_minus3** | ue(v) |
| **log2\_diff\_max\_min\_luma\_coding\_block\_size** | ue(v) |
| **log2\_min\_transform\_block\_size\_minus2** | ue(v) |
| **log2\_diff\_max\_min\_transform\_block\_size** | ue(v) |
| **max\_transform\_hierarchy\_depth\_inter** | ue(v) |
| **max\_transform\_hierarchy\_depth\_intra** | ue(v) |
| **scaling\_list\_enabled\_flag** | u(1) |
| if( scaling\_list\_enabled\_flag ) { |  |
| if( nuh\_layer\_id > 0 ) |  |
| **sps\_infer\_scaling\_list\_flag** | u(1) |
| if( sps\_infer\_scaling\_list\_flag ) |  |
| **sps\_scaling\_list\_ref\_layer\_id** | u(6) |
| else { |  |
| **sps\_scaling\_list\_data\_present\_flag** | u(1) |
| if( sps\_scaling\_list\_data\_present\_flag ) |  |
| scaling\_list\_data( ) |  |
| } |  |
| } |  |
| **amp\_enabled\_flag** | u(1) |
| **sample\_adaptive\_offset\_enabled\_flag** | u(1) |
| **pcm\_enabled\_flag** | u(1) |
| if( pcm\_enabled\_flag ) { |  |
| **pcm\_sample\_bit\_depth\_luma\_minus1** | u(4) |
| **pcm\_sample\_bit\_depth\_chroma\_minus1** | u(4) |
| **log2\_min\_pcm\_luma\_coding\_block\_size\_minus3** | ue(v) |
| **log2\_diff\_max\_min\_pcm\_luma\_coding\_block\_size** | ue(v) |
| **pcm\_loop\_filter\_disabled\_flag** | u(1) |
| } |  |
| **num\_short\_term\_ref\_pic\_sets** | ue(v) |
| for( i = 0; i < num\_short\_term\_ref\_pic\_sets; i++) |  |
| short\_term\_ref\_pic\_set( i ) |  |
| **long\_term\_ref\_pics\_present\_flag** | u(1) |
| if( long\_term\_ref\_pics\_present\_flag ) { |  |
| **num\_long\_term\_ref\_pics\_sps** | ue(v) |
| for( i = 0; i < num\_long\_term\_ref\_pics\_sps; i++ ) { |  |
| **lt\_ref\_pic\_poc\_lsb\_sps**[ i ] | u(v) |
| **used\_by\_curr\_pic\_lt\_sps\_flag**[ i ] | u(1) |
| } |  |
| } |  |
| **sps\_temporal\_mvp\_enabled\_flag** | u(1) |
| **strong\_intra\_smoothing\_enabled\_flag** | u(1) |
| **vui\_parameters\_present\_flag** | u(1) |
| if( vui\_parameters\_present\_flag ) |  |
| vui\_parameters( ) |  |
| **sps\_extension\_flag** | u(1) |
| if( sps\_extension\_flag ) { |  |
| sps\_extension( ) |  |
| **sps\_extension2\_flag** | u(1) |
| if( sps\_extension2\_flag ) |  |
| while( more\_rbsp\_data( ) ) |  |
| **sps\_extension\_data\_flag** | u(1) |
| } |  |
| rbsp\_trailing\_bits( ) |  |
| } |  |

Sequence parameter set extension syntax

|  |  |
| --- | --- |
| sps\_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\_left\_offset**[ i ] | se(v) |
| **scaled\_ref\_layer\_top\_offset**[ i ] | se(v) |
| **scaled\_ref\_layer\_right\_offset**[ i ] | se(v) |
| **scaled\_ref\_layer\_bottom\_offset**[ i ] | se(v) |
| **}** |  |
| **}** |  |

* + - * 1. Picture parameter set RBSP syntax

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

* + - * 1. Supplemental enhancement information RBSP syntax

The specifications in subclause 7.3.2.4 apply.

* + - * 1. Access unit delimiter RBSP syntax

The specifications in subclause 7.3.2.5 apply.

* + - * 1. End of sequence RBSP syntax

The specifications in subclause 7.3.2.6 apply.

* + - * 1. End of bitstream RBSP syntax

The specifications in subclause 7.3.2.7 apply.

* + - * 1. Filler data RBSP syntax

The specifications in subclause 7.3.2.8 apply.

* + - * 1. Slice segment layer RBSP syntax

The specifications in subclause 7.3.2.9 apply.

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

The specifications in subclause 7.3.2.10 apply.

* + - * 1. RBSP trailing bits syntax

The specifications in subclause 7.3.2.11 apply.

* + - * 1. Byte alignment syntax

The specifications in subclause 7.3.2.12 apply.

* + - 1. Profile, tier and level syntax

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

* + - 1. Scaling list data syntax

The specifications in subclause 7.3.4 apply.

* + - 1. Supplemental enhancement information message syntax

The specifications in subclause 7.3.5 apply.

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

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

* + - * 1. Reference picture list modification syntax

The specifications in subclause 7.3.6.2 apply.

* + - * 1. Weighted prediction parameters syntax

The specifications in subclause 7.3.6.3 apply.

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

The specifications in subclause 7.3.7 apply.

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

The specifications in subclause 7.3.8.1 apply.

* + - * 1. Coding tree unit syntax

The specifications in subclause 7.3.8.2 apply.

* + - * 1. Sample adaptive offset syntax

The specifications in subclause 7.3.8.3 apply.

* + - * 1. Coding quadtree syntax

The specifications in subclause 7.3.8.4 apply.

* + - * 1. Coding unit syntax

The specifications in subclause 7.3.8.5 apply.

* + - * 1. Prediction unit syntax

The specifications in subclause 7.3.8.6 apply.

* + - * 1. PCM sample syntax

The specifications in subclause 7.3.8.7 apply.

* + - * 1. Transform tree syntax

The specifications in subclause 7.3.8.8 apply.

* + - * 1. Motion vector difference syntax

The specifications in subclause 7.3.8.9 apply.

* + - * 1. Transform unit syntax

The specifications in subclause 7.3.8.10 apply.

* + - * 1. Residual coding syntax

The specifications in subclause 7.3.8.11 apply.

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

The specifications in subclause 7.4.2.1 apply.

* + - * 1. NAL unit header semantics

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

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

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

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

**nuh\_layer\_id** specifies the identifier of the layer.

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

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

* + - * 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.

Each PPS RBSP is initially considered not active for any layer with nuh\_layer\_id greater than 0 at the start of the operation of the decoding process. At most one PPS RBSP is considered active for each non-zero nuh\_layer\_id value at any given moment during the operation of the decoding process, and the activation of any particular PPS RBSP for a particular non-zero nuh\_layer\_id value results in the deactivation of the previously-active PPS RBSP for that non-zero nuh\_layer\_id value (if any).

When a PPS RBSP (with a particular value of pps\_pic\_parameter\_set\_id) is not active for a nuh\_layer\_id value 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 and having that value of nuh\_layer\_id), it is activated for that nuh\_layer\_id value. This PPS RBSP is called the active layer PPS RBSP for that nuh\_layer\_id value until it is deactivated by the activation of another PPS RBSP for the same 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. The nuh\_layer\_id value of the NAL unit containing the PPS RBSP that is activated for nuh\_layer\_id equal to nuhLayerIdA shall be less than or equal to nuhLayerIdA. The same PPS RBSP may be the active layer PPS for more than one nuh\_layer\_id value.

Any PPS NAL unit containing the value of pps\_pic\_parameter\_set\_id for the active layer PPS RBSP for a coded picture shall have the same content as that of the active layer 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.

Each SPS RBSP is initially considered not active for any layer with nuh\_layer\_id greater than 0 at the start of the operation of the decoding process. At most one SPS RBSP is considered active for each non-zero nuh\_layer\_id value at any given moment during the operation of the decoding process, and the activation of any particular SPS RBSP for a particular non-zero nuh\_layer\_id value results in the deactivation of the previously-active SPS RBSP for that non-zero nuh\_layer\_id value (if any).

When an SPS RBSP (with a particular value of sps\_seq\_parameter\_set\_id) is not already active for a nuh\_layer\_id value and it is referred to by activation of a PPS RBSP for that nuh\_layer\_id value (in which pps\_seq\_parameter\_set\_id is equal to the sps\_seq\_parameter\_set\_id value), it is activated for that nuh\_layer\_id value. This SPS RBSP is called the active layer SPS RBSP for that nuh\_layer\_id value until it is deactivated by the activation of another SPS RBSP for the same 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 a particular nuh\_layer\_id value shall remain active for a sequence of pictures in decoding order with that nuh\_layer\_id value starting from a LIP picture having that nuh\_layer\_id value, inclusive, until either the next LIP picture with that nuh\_layer\_id value, exclusive, or the end of the CVS, whichever is earlier. The nuh\_layer\_id value the NAL unit containing the SPS RBSP that is activated for nuh\_layer\_id equal to nuhLayerIdA shall be less than or equal to nuhLayerIdA. The same SPS RBSP may be the active layer SPS for more than one nuh\_layer\_id value.

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

During operation of the decoding process for VCL NAL units with a non-zero nuh\_layer\_id value, the values of parameters of the active layer SPS for that non-zero nuh\_layer\_id value, and the active layer PPS RBSP for that non-zero nuh\_layer\_id value are considered in effect.

Order of access units and their association to CVS

The specifications in subclause 7.4.2.4.3 apply.

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

The specifications in subclause 7.4.2.4.4 apply.

Order of VCL NAL units and association to coded pictures

The specifications in subclause 7.4.2.4.5 apply.

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

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

– layerSetLayerIdList is replaced by LayerSetLayerIdList.

– numLayersInIdList is replaced by NumLayersInIdList.

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

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

**vps\_extension\_flag** equal to 0 specifies that no vps\_extension( ) syntax structure is present in the VPS RBSP syntax structure. vps\_extension\_flag equal to 1 specifies that the vps\_extension( ) syntax structure is present in the VPS RBSP syntax structure. When vps\_max\_layers\_minus1 is greater than 0, vps\_extension\_flag shall be equal to 1.

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

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

Video parameter set extension semantics

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

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

**vps\_vui\_offset** specifies the byte offset, starting from the beginning of the VPS NAL unit, of the set of fixed-length coded information starting from bit\_rate\_present\_vps\_flag, when present, in the VPS NAL unit. When present, emulation prevention bytes that appear in the VPS NAL unit are counted for purposes of byte offset identification.  
**splitting\_flag** equal to 1 indicates that the dimension\_id[ i ][ j ] syntax elements are not present and that the binary representation of the nuh\_layer\_id value in the NAL unit header are split into NumScalabilityTypes segments with lengths, in bits, according to the values of dimension\_id\_len\_minus1[ j ] and that the values of dimension\_id[ LayerIdxInVps[ nuh\_layer\_id ] ][ j ] are inferred from the NumScalabilityTypes segments. splitting\_flag equal to 0 indicates that the syntax elements dimension\_id[ i ][ j ] are present.

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

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

Table F‑1 – Mapping of ScalabiltyId to scalability dimensions

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

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

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

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

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

 (F‑1)

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

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

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

**vps\_nuh\_layer\_id\_present\_flag** equal to 1specifies that layer\_id\_in\_nuh[ i ] for i from 0 to vps\_max\_layers\_minus1, inclusive, are present. vps\_nuh\_layer\_id\_present\_flag equal to 0 specifies that layer\_id\_in\_nuh[ i ] for i from 0 to vps\_max\_layers\_minus1, 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 vps\_max\_layers\_minus1, 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 vps\_max\_layers\_minus1, 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.

If splitting\_flag is equal to 1, for i from 0 to vps\_max\_layers\_minus1, 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, for j from 0 to NumScalabilityTypes − 1, inclusive, dimension\_id[ 0 ][ j ] is inferred to be equal to 0.

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

NumViews = 1

for( i = 0; i <= vps\_max\_layers\_minus1; i++ ) {

lId = layer\_id\_in\_nuh[ i ]

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

if( scalability\_mask\_flag[ smIdx ] )

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

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

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

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

NumViews++

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

}

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

**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\_minus1 + 1 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 vps\_max\_layers\_minus1, it is inferred to be equal to 0.

The variables NumDirectRefLayers[ i ], ~~and~~ RefLayerId[ i ][ j ] SamplePredEnabledFlag[ i ][ j ], MotionPredEnabledFlag[ i ][ j ] and DirectRefLayerIdx[ i ][ j ] are derived as follows:

for( i = 0; i <= vps\_max\_layers\_minus1; 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 ]  
 SamplePredEnabledFlag[ iNuhLId ][ j ] = ( ( direct\_dependency\_type[ i ][ j ] + 1 ) & 1 )  
 MotionPredEnabledFlag[ iNuhLId ][ j ] = ( ( ( direct\_dependency\_type[ i ][ j ] + 1 ) & 2 ) >> 1 )  
 DirectRefLayerIdx[ iNuhLid ][ layer\_id\_in\_nuh[ j ] ] = NumDirectRefLayers[ iNuhLId ] – 1 }  
}

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

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

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

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

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

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

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

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

**default\_one\_target\_output\_layer\_flag** equal to 1 specifies that only the highest layer in each of the default output layer sets is a target output layer. default\_one\_target\_output\_layer\_flag equal to 0 specifies that all layers in each of the default output layer sets are target output layers.

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

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

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

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

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

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

**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 0. The value of vps\_rep\_format\_idx[ i ] shall be in the range of 0 to vps\_num\_rep\_formats\_minus1, inclusive.

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

**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.

**direct\_dep\_type\_len\_minus2** plus 2 specifies the number of bits of the direct\_dependency\_type[ i ][ j ] syntax element. 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.

**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 sample motion prediction and inter-layer motion prediction of the layer with nuh\_layer\_id equal layer\_id\_in\_nuh[ i ]. 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.

**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.

**enable\_higher\_layer\_ref\_pic\_pred[ i ]** equal to 0 specifies that within the CVS, the decoded pictures with nuh\_layer\_id greater than layer\_id\_in\_nuh[ i ], are not used as reference for pictures with nuh\_layer\_id equal to layer\_id\_in\_nuh[ i ]. enable\_higher\_layer\_ref\_pic\_pred[ i ]equal to 1 specifies that within the CVS, the decoded pictures with nuh\_layer\_id equal to layer\_id\_in\_nuh[ i + 1], when available, may be used as a reference for pictures with nuh\_layer\_id equal to layer\_id\_in\_nuh[ i ] and TemporalId greater than 0. When not present, enable\_higher\_layer\_ref\_pic\_pred[ i ]is inferred to be equal to 0.

[Note: It is a requirement of bitstream conformance that enable\_higher\_layer\_ref\_pic\_pred[ i ] shall be equal to 0 for non-SNR scalability cases.]

**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\_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** are used for inference of the values of the SPS syntax elements 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, 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 syntax element also apply.

VPS VUI semantics

**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‑2)

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‑3)

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

( x & ( 214 − 1 ) ) = = 0 (F‑4)

**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‑5)

**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‑6)

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‑7)

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

avg\_pic\_rate[ i ] = = 0 (F‑8)

**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.

**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.

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‑9)  
 ( min\_horizontal\_ctu\_offset\_plus1[ i ][ j ] – 1 ) : ( refPicWidthInCtbsY[ i ][ j ] ‑ 1 )

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

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

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

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

– 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[ i ] )  <<  curCtbLog2SizeY (F‑10)

yP = ( ctbAddr / curPicWidthInCtbsY[ i ] )  <<  curCtbLog2SizeY (F‑11)

– 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‑12)

curScaledRefLayerPicHeightInSamplesL[ i ][ j ] = curPicHeightInSamplesL[ i ]  –   
 curScaledRefLayerTopOffset[ i ][ j ] – curScaledRefLayerBottomOffset[ i ][ j ] (F‑13)

scaleFactorX[ i ][ j ] = ( ( refPicWidthInSamplesL [ i ][ j ]  << 16 ) +   
( curScaledRefLayerPicWidthInSamplesL [ i ][ j ]>> 1 ) )/curScaledRefLayerPicWidthInSamplesL [ i ][ j ] (F‑14)

scaleFactorY[ i ][ j ] = ( ( refPicHeightInSamplesL [ i ][ j ] << 16 ) +   
( curScaledRefLayerPicHeightInSamplesL >> 1 ) ) / curScaledRefLayerPicHeightInSamplesL [ i ][ j ] (F‑15)

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

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

yCol [ i ][ j ] = Clip3( 0 , ( refPicHeightInSamplesL[ i ][ j ]– 1 ), ( ( yP ‑ curScaledRefLayerTopOffset[ i ][ j ]) \* scaleFactorY[ i ][ j ] + ( 1 << 15 ) ) >> 16)) (F‑17)

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

xColCtb[ i ][ j ] = xCol[ i ][ j ]  >>  refCtbLog2SizeY[ i ][ j ] (F‑18)

yColCtb[ i ][ j ] = yCol[ i ][ j ]  >>  refCtbLog2SizeY[ i ][ j ] (F‑19)

colCtbAddr[ i ][ j ] = xColCtb[ i ][ j ] + ( yColCtb[ i ][ j ] \* refPicWidthInCtbsY[ i ][ j ] ) (F‑20)

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‑21)

yOffset[ i ][ j ] = ( min\_spatial\_segment\_offset\_plus1[ i ][ j ] – 1 ) \* refPicWidthInCtbsY[ i ][ j ] (F‑22)

refCtbAddr[ i ][ j ] = colCtbAddr[ i ][ j ] + xOffset[ i ][ j ] + yOffset[ i ][ j ] (F‑23)

* 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.
  + - * 1. Sequence parameter set RBSP semantics

The specifications in subclause 7.4.3.2 apply, with following additions and modifications.

NOTE  – All SPSs, regardless of the values of their nuh\_layer\_id, share the same value space for sps\_seq\_parameter\_set\_id. In other words, an SPS with nuh\_layer\_id equal to X and sps\_seq\_parameter\_set\_id equal to A would update the previously received SPS with nuh\_layer\_id not equal to X and sps\_seq\_parameter\_set\_id equal to A.

**sps\_max\_sub\_layers\_minus1** plus 1 specifies the maximum number of temporal sub-layers that may be present in each CVS referring to the SPS. The value of sps\_max\_sub\_layers\_minus1 shall be in the range of 0 to 6, inclusive. When not present sps\_max\_sub\_layers\_minus1 is inferred to be equal to vps\_max\_sub\_layers\_minus1.

**update\_rep\_format\_flag** equal to 1 specifies that the syntax elements 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 explicitly signalled in the SPS and all the layers with nuh\_layer\_id greater than zero that refer to this SPS use these values instead of those signalled in the VPS when the nuh\_layer\_id of the SPS is greater than 0. update\_rep\_format\_flag equal to 0 specifies that the syntax elements 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 not signalled in the SPS and all the layers that refer to this SPS use the values signaled in the VPS. When not present, the value of update\_rep\_format\_flag is inferred to be equal to 1.

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:

– If the nuh\_layer\_id of the active layer SPS 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 vps\_rep\_format\_idx[ j ]-th rep\_format( ) syntax structure in the active VPS where j is equal to LayerIdxInVps[ layerIdCurr ] 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 layer SPS are ignored.

NOTE  – The values are inferred from the VPS when an active non-base layer references an SPS which is also used by the base layer, in which case the SPS has nuh\_layer\_id equal to 0. For an active base layer, the values in the active SPS apply.

– Otherwise (the nuh\_layer\_id of the active layer SPS is greater than zero), the following applies:

– If update\_rep\_format\_flag 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 vps\_rep\_format\_idx[ j ]-th rep\_format( ) syntax structure in the active VPS, where j is equal to LayerIdxInVps[ layerIdCurr ].

– Otherwise (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.

**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  – 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‑24)

QpBdOffsetY = 6 \* bit\_depth\_luma\_minus8 (F‑25)

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‑26)

QpBdOffsetC = 6 \* bit\_depth\_chroma\_minus8 (F‑27)

bit\_depth\_chroma\_minus8 shall be in the range of 0 to 6, inclusive. bit\_depth\_chroma\_minus8 shall be less than or equal to bit\_depth\_vps\_chroma\_minus8.

**sps\_infer\_scaling\_list\_flag** equal to 1 specifies that the syntax elements of the scaling list data syntax structure of the SPS are inferred to be equal to those of the SPS that is active for the layer with nuh\_layer\_id equal to sps\_scaling\_list\_ref\_layer\_id. sps\_infer\_scaling\_list\_flag equal to 0 specifies that the syntax elements of the scaling list data syntax structure are not inferred. When not present, the value of sps\_infer\_scaling\_list\_flag is inferred to be 0.

**sps\_scaling\_list\_ref\_layer\_id** specifies the value of the nuh\_layer\_id of the layer for which the active SPS is associated with the same scaling list data as the current SPS.

The value of sps\_scaling\_list\_ref\_layer\_id shall be in the range of 0 to 62, inclusive.

When avc\_base\_layer\_flag is equal to 1, it is a requirement of bitstream conformance that the value of sps\_scaling\_list\_ref\_layer\_id shall be greater than 0.

It is a requirement of bitstream conformance that, when an SPS with nuh\_layer\_id equal to nuhLayerIdA is active for a layer with nuh\_layer\_id equal to nuhLayerIdB and sps\_infer\_scaling\_list\_flag in the SPS is equal to 1, sps\_infer\_scaling\_list\_flag shall be equal to 0 for the SPS that is active for the layer with nuh\_layer\_id equal to sps\_scaling\_list\_ref\_layer\_id.

It is a requirement of bitstream conformance that, when an SPS with nuh\_layer\_id equal to nuhLayerIdA is active for a layer with nuh\_layer\_id equal to nuhLayerIdB, the layer with nuh\_layer\_id equal to sps\_scaling\_list\_ref\_layer\_id shall be a direct or indirect reference layer of the layer with nuh\_layer\_id equal to nuhLayerIdB.

**sps\_scaling\_list\_data\_present\_flag** equal to 1 specifies that the scaling list data syntax structure is present in the SPS. sps\_scaling\_list\_data\_present\_flag equal to 0 specifies that the scaling list data syntax structure is not present in the SPS. When not present, the value of sps\_scaling\_list\_data\_present\_flag is inferred to be equal to 0.

**sps\_temporal\_id\_nesting\_flag**, when sps\_max\_sub\_layers\_minus1 is greater than 0, specifies whether inter prediction is additionally restricted for CVSs referring to the SPS. When vps\_temporal\_id\_nesting\_flag is equal to 1, sps\_temporal\_id\_nesting\_flag shall be equal to 1. When sps\_max\_sub\_layers\_minus1 is equal to 0, sps\_temporal\_id\_nesting\_flag shall be equal to 1. When not present sps\_temporal\_id\_nesting\_flag is inferred to be equal to vps\_temporal\_id\_nesting\_flag.

NOTE  – The syntax element sps\_temporal\_id\_nesting\_flag is used to indicate that temporal up-switching, i.e. switching from decoding up to any TemporalId tIdN to decoding up to any TemporalId tIdM that is greater than tIdN, is always possible in the CVS.

**sps\_extension\_flag** equal to 0 specifies that no sps\_extension( ) syntax structure is present in the SPS RBSP syntax structure. sps\_extension\_flag equal to 1 specifies that the sps\_extension( ) syntax structure is present in the SPS RBSP syntax structure.

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

Sequence parameter set 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 63, inclusive.

**scaled\_ref\_layer\_left\_offset**[ i ] specifies the horizontal offset between the top-left luma sample of the resampled i-th direct reference layer picture used for inter-layer prediction 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[ i ] is inferred to be equal to 0.

**scaled\_ref\_layer\_top\_offset**[ i ] specifies the vertical offset between the top-left luma sample of the resampled i-th direct reference layer picture used for inter-layer prediction 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[ i ] is inferred to be equal to 0.

**scaled\_ref\_layer\_right\_offset**[ i ] specifies the horizontal offset between the bottom-right luma sample of the resampled i-th direct reference layer picture used for inter-layer prediction 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[ i ] is inferred to be equal to 0.

**scaled\_ref\_layer\_bottom\_offset**[ i ] specifies the vertical offset between the bottom-right luma sample of the resampled i-th direct reference layer picture used for inter-layer prediction 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[ i ] is inferred to be equal to 0.

* + - * 1. Picture parameter set RBSP semantics

The specifications in subclause 7.4.3.3 apply, with the following modifications:

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

**num\_extra\_slice\_header\_bits** specifies the number of extra slice header bits that are present in the slice header RBSP for coded pictures referring to the PPS. num\_extra\_slice\_header\_bits shall be in the range of 0 to 2, inclusive, in bitstreams conforming to this version of this Specification. Other values for num\_extra\_slice\_header\_bits are reserved for future use by ITU-T | ISO/IEC. However, decoders shall allow num\_extra\_slice\_header\_bits to have any value.

When cross\_layer\_irap\_aligned\_flag is equal to 0, it is a requirement of bitstream conformance that num\_extra\_slice\_header\_bits shall be greater than or equal to 1.

**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 is associated with the same scaling list data as the current PPS.

The value of pps\_scaling\_list\_ref\_layer\_id shall be in the range of 0 to 62, inclusive.

When avc\_base\_layer\_flag is equal to 1, it is a requirement of bitstream conformance that pps\_scaling\_list\_ref\_layer\_id shall be greater than 0.

It is a requirement of bitstream conformance that, when a PPS with nuh\_layer\_id equal to nuhLayerIdA is active for a layer with nuh\_layer\_id equal to nuhLayerIdB and pps\_infer\_scaling\_list\_flag in the PPS is equal to 1, pps\_infer\_scaling\_list\_flag shall be equal to 0 for the PPS that is active for the layer with nuh\_layer\_id equal to pps\_scaling\_list\_ref\_layer\_id.

It is a requirement of bitstream conformance that, when a PPS with nuh\_layer\_id equal to nuhLayerIdA is active for a layer with nuh\_layer\_id equal to nuhLayerIdB, the layer with nuh\_layer\_id equal to pps\_scaling\_list\_ref\_layer\_id shall be a direct or indirect reference layer of the layer with nuh\_layer\_id equal to nuhLayerIdB.

**pps\_scaling\_list\_data\_present\_flag** equal to 1 specifies that parameters are present in the PPS to modify the scaling lists specified by the active SPS. pps\_scaling\_list\_data\_present\_flag equal to 0 specifies that the scaling list data used for the pictures referring to the PPS are inferred to be equal to those specified by the active SPS. When scaling\_list\_enabled\_flag is equal to 0, the value of pps\_scaling\_list\_data\_present\_flag shall be equal to 0. When scaling\_list\_enabled\_flag is equal to 1, sps\_scaling\_list\_data\_present\_flag is equal to 0, and pps\_scaling\_list\_data\_present\_flag is equal to 0, the default scaling list data are used to derive the array ScalingFactor as described in the scaling list data semantics 7.4.5.

* + - * 1. Supplemental enhancement information RBSP semantics

The specifications in subclause 7.4.3.4 apply.

* + - * 1. Access unit delimiter RBSP semantics

The specifications in subclause 7.4.3.5 apply.

* + - * 1. End of sequence RBSP semantics

The specifications in subclause 7.4.3.6 apply.

* + - * 1. End of bitstream RBSP semantics

The specifications in subclause 7.4.3.7 apply.

* + - * 1. Filler data RBSP semantics

The specifications in subclause 7.4.3.8 apply.

* + - * 1. Slice segment layer RBSP semantics

The specifications in subclause 7.4.3.9 apply.

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

The specifications in subclause 7.4.3.10 apply.

* + - * 1. RBSP trailing bits semantics

The specifications in subclause 7.4.3.11 apply.

* + - * 1. Byte alignment semantics

The specifications in subclause 7.4.3.12 apply.

* + - 1. Profile, tier and level semantics

The profile\_tier\_level( ) syntax structure provides profile, tier and level information used for a layer set. When the profile\_tier\_level( ) syntax structure is included in a vps\_extension( ) syntax structure, the applicable layer set to which the profile\_tier\_level( ) syntax structure applies is specified by the corresponding lsIdx variable in the vps\_extension( ) syntax structure. When the profile\_tier\_level( ) syntax structure is included in a VPS, but not in a vps\_extension( ) syntax structure, the applicable layer set to which the profile\_tier\_level( ) syntax structure applies is the layer set specified by the index 0. When the profile\_tier\_level( ) syntax structure is included in an SPS, the layer set to which the profile\_tier\_level( ) syntax structure applies is the layer set specified by the index 0.

For interpretation of the following semantics, CVS refers to the CVS subset associated with the layer set to which the profile\_tier\_level( ) syntax structure applies.

When the syntax elements general\_profile\_space, general\_tier\_flag, general\_profile\_idc, general\_profile\_compatibility\_flag[ j ], general\_progressive\_source\_flag, general\_interlaced\_source\_flag, general\_non\_packed\_constraint\_flag, general\_frame\_only\_constraint\_flag, general\_reserved\_zero\_44bits are not present for the applicable layer set, they are inferred to be equal to the corresponding values of the layer set specified by the index (profile\_layer\_set\_ref\_minus1[ lsIdx ] +1 ).

When the syntax elements sub\_layer\_profile\_space[ i ], sub\_layer\_tier\_flag[ i ], sub\_layer\_profile\_idc[ i ], sub\_layer\_profile\_compatibility\_flag[ i ][ j ], sub\_layer\_progressive\_source\_flag[ i ], sub\_layer\_interlaced\_source\_flag[ i ], sub\_layer\_non\_packed\_constraint\_flag[ i ], sub\_layer\_frame\_only\_constraint\_flag[ i ], sub\_layer\_reserved\_zero\_44bits[ i ] are not present for the applicable layer set, and they are present in or inferred for the layer set specified by the index (profile\_layer\_set\_ref\_minus1[ lsIdx ] +1 ) they are inferred to be equal to the corresponding values of the layer set specified by the index (profile\_layer\_set\_ref\_minus1[ lsIdx ] +1 ).

The specifications in subclause 7.4.4 apply, with following modifications.

**general\_tier\_flag** specifies the tier context for the interpretation of general\_level\_idc as specified in Annex A or subclause G.11 or subclause H.11.

**general\_profile\_idc**, when general\_profile\_space is equal to 0, indicates a profile to which the CVS conforms as specified in Annex A or in subclause G.11 or in subclause H.11. Bitstreams shall not contain values of general\_profile\_idc other than those specified in Annex A or subclause G.11 or in subclause H.11. Other values of general\_profile\_idc are reserved for future use by ITU-T | ISO/IEC.

**general\_profile\_compatibility\_flag**[ j ] equal to 1, when general\_profile\_space is equal to 0, indicates that the CVS conforms to the profile indicated by general\_profile\_idc equal to i as specified in Annex A or in subclause G.11 or in subclause H.11. When general\_profile\_space is equal to 0, general\_profile\_compatibility\_flag[ general\_profile\_idc ] shall be equal to 1. The value of general\_profile\_compatibility\_flag[ j ] shall be equal to 0 for any value of j that is not specified as an allowed value of general\_profile\_idc in Annex A or in subclause G.11 or in subclause H.11.

**general\_level\_idc** indicates a level to which the CVS conforms as specified in Annex A or subclause G.11 or subclause H.11. Bitstreams shall not contain values of general\_level\_idc other than those specified in Annex A or subclause G.11 or subclause H.11. Other values of general\_level\_idc are reserved for future use by ITU-T | ISO/IEC.

**sub\_layer\_profile\_present\_flag**[ i ] equal to 1, specifies that profile information is present in the profile\_tier\_level( ) syntax structure for the representation of the sub-layer with TemporalId equal to i. sub\_layer\_profile\_present\_flag[ i ] equal to 0 specifies that profile information is not present in the profile\_tier\_level( ) syntax structure for the representations of the sub-layer with TemporalId equal to i. When profilePresentFlag is equal to 0, sub\_layer\_profile\_present\_flag[ i ] shall be equal to 0.

* + - 1. Scaling list data semantics

The specifications in subclause 7.4.5 apply.

* + - 1. Supplemental enhancement information message semantics

The specifications in subclause 7.4.6 apply.

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

The specifications in subclause 7.4.7.1 apply with the following modifications.

– "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."

**poc\_reset\_flag** equal to 1 specifies that the derived picture order count for the current picture is equal to 0. poc\_reset\_flag equal to 0 specifies that the derived picture order count for the current picture may or may not be equal to 0. It is a requirement of bitstream conformance that when cross\_layer\_irap\_aligned\_flag is equal to 1, the value of poc\_reset\_flag shall be equal to 0. When not present, the value of poc\_reset\_flag is inferred to be equal to 0.

**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.

**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 variable NumActiveRefLayerPics is derived as follows:

if( nuh\_layer\_id = = 0 | | NumDirectRefLayers[ nuh\_layer\_id ] = = 0 )  
 NumActiveRefLayerPics = 0  
else if( all\_ref\_layers\_active\_flag )  
 NumActiveRefLayerPics = NumDirectRefLayers[ nuh\_layer\_id ]  
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 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 ] ]

All slices of a picture shall have the same value of inter\_layer\_pred\_layer\_idc[ i ] for each value of i in the range of 0 to NumActiveRefLayerPics − 1, inclusive.

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 ] ] ] is greater than TemporalId.

– The values of max\_tid\_il\_ref\_pics\_plus1[ LayerIdxInVps[ RefPicLayerId[ i ] ] ] 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.

* + - * 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‑28)  
 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.

* + - 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 additions.

When the current picture has nuh\_layer\_id greater than 0, the following applies.

– 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).

– 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.8.1.1 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 is invoked.

– 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.8.1.2 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 4.
2. The processes in subclause F.8.3 specify the following decoding processes using syntax elements in the slice segment layer and above:

– Variables and functions relating to picture order count are derived in subclause F.8.3.1. This needs to be invoked only for the first slice segment of a picture. It is a requirement of bitstream conformance that PicOrderCntVal shall remain unchanged within an access unit.

– The decoding process for RPS in subclause F.8.3.2 is invoked, wherein only reference pictures with a nuh\_layer\_id equal to that of CurrPic may be marked as "unused for reference" or "used for long-term reference" and any picture with a different value of nuh\_layer\_id is not marked. This needs to be invoked only for the first slice segment of a picture.

– When FirstPicInLayerDecodedFlag[ nuh\_layer\_id ] is equal to 0, the decoding process for generating unavailable reference pictures specified in subclause F.8.1.3 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 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, if LayerInitialisedFlag[ nuh\_layer\_id ] is equal to 0, PicOutputFlag is set equal to 0.

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

The following applies:

– If discardable\_flag is equal to 1, the decoded picture is marked as "unused for reference".

– Otherwise, the decoded picture is marked as "used for short-term reference".

When TemporalId is equal to HighestTid, the marking process for sub-layer non-reference pictures not needed for inter-layer prediction specified in subclause F.8.1.2.1 is invoked with latestDecLayerId equal to nuh\_layer\_id as input.

FirstPicInLayerDecodedFlag[ nuh\_layer\_id ] is set equal to 1.

* + - * 1. Marking process for sub-layer non-reference pictures not needed for inter-layer prediction

Input to this process is:

– a nuh\_layer\_id value latestDecLayerId

Output of this process is:

– potentially updated marking as "unused for reference" for some decoded pictures

NOTE – This process marks pictures that are not needed for inter or inter-layer prediction as "unused for reference". When TemporalId is less than HighestTid, the current picture may be used for reference in inter prediction and this process is not invoked.

The variables numTargetDecLayers, and latestDecIdx are derived as follows:

– numTargetDecLayers is set equal to the number of entries in TargetDecLayerIdList.

– latestDecIdx is set equal to the value of i for which TargetDecLayerIdList[ i ] is equal to latestDecLayerId.

For i in the range of 0 to latestDecIdx, inclusive, the following applies for marking of pictures as "unused for reference":

– Let currPic be the picture in the current access unit with nuh\_layer\_id equal to TargetDecLayerIdList[ i ].

– When currPic is marked as "used for reference" and is a sub-layer non-reference picture, the following applies:

– The variable currTid is set equal to the value of TemporalId of currPic.

– The variable remainingInterLayerReferencesFlag is derived as specified in the following:

remainingInterLayerReferencesFlag = 0  
 if ( currTid <= ( max\_tid\_il\_ref\_pics\_plus1[ LayerIdxInVps[ TargetDecLayerIdList[ i ] ] ] –1 ) )  
 for( j = latestDecIdx + 1; j < numTargetDecLayers; j++ )  
 for( k = 0; k < NumDirectRefLayers[ TargetDecLayerIdList[ j ] ]; k++ )  
 if( TargetDecLayerIdList[ i ] = = RefLayerId[ TargetDecLayerIdList[ j ] ][ k ] )  
 remainingInterLayerReferencesFlag = 1

– When remainingInterLayerReferenceFlag is equal to 0, currPic is marked as "unused for reference".

* + - 1. Generation of unavailable reference pictures for pictures first in decoding order within a layer

This process is invoked for a picture with nuh\_layer\_id equal to layerId, when FirstPicInLayerDecodedFlag[ layerId ] is equal to 0.

NOTE – A cross-layer random access skipped (CL-RAS) picture is a picture with nuh\_layer\_id equal to layerId such that LayerInitialisedFlag[ 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 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 nuh\_layer\_id equal to nuh\_layer\_id of the current picture and that is not a RASL picture, a RADL picture, or a sub-layer non-reference picture, and let prevPicOrderCnt be equal to PicOrderCntVal of prevTid0Pic.
* The variable prevPicOrderCntLsb is set equal to prevPicOrderCnt & ( MaxPicOrderCntLsb − 1 ).
* The variable prevPicOrderCntMsb is set equal to prevPicOrderCnt − 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 (F‑29)  
else if( (slice\_pic\_order\_cnt\_lsb > prevPicOrderCntLsb ) &&  
 ( (slice\_pic\_order\_cnt\_lsb − prevPicOrderCntLsb ) > ( MaxPicOrderCntLsb / 2 ) ) )  
 PicOrderCntMsb = prevPicOrderCntMsb − MaxPicOrderCntLsb  
else  
 PicOrderCntMsb = prevPicOrderCntMsb

PicOrderCntVal is derived as follows:

PicOrderCntVal = PicOrderCntMsb + slice\_pic\_order\_cnt\_lsb (F‑30)

NOTE  – 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.

When poc\_reset\_flag is equal to 1, the following steps apply in the order listed:

* The PicOrderCntVal of each picture that is in the DPB and belongs to the same layer as the current picture is decremented by PicOrderCntVal.
* PicOrderCntVal is 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 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‑31)

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

DiffPicOrderCnt( picA, picB ) = PicOrderCnt( picA ) − PicOrderCnt( picB ) (F‑32)

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

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 8.3.3. 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". This subclause marks only the pictures with the same value of nuh\_layer\_id and does not mark any picture with a nuh\_layer\_id different from the current picture.

NOTE  – 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 to be the nuh\_layer\_id of the current picture.

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 (F‑33)  
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 F.8.3.1.

NOTE  – 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 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. 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 nuh\_layer\_id is equal to 0 and the current picture is a BLA picture or a 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  – 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 variable offsetPicLayerId is set equal to 1 when enable\_higher\_layer\_ref\_pic\_pred[ currPicLayerId ] not equal to 0 and TemporalId is not equal to 0 for the current picture.

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 slice\_pic\_order\_cnt\_lsb equal to PocLtCurr[ i ] and nuh\_layer\_id equal to currPicLayerId + offsetPicLayerId)  
 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 + offsetPicLayerId )  
 RefPicSetLtCurr[ i ] = picX  
 else  
 RefPicSetLtCurr[ i ] = "no reference picture" (F‑34)  
for( i = 0; i < NumPocLtFoll; i++ )  
 if( !FollDeltaPocMsbPresentFlag[ i ] )  
 if( there is a reference picture picX in the DPB with slice\_pic\_order\_cnt\_lsb equal to PocLtFoll[ i ] and nuh\_layer\_id equal to currPicLayerId + offsetPicLayerId )  
 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 + offsetPicLayerId )  
 RefPicSetLtFoll[ i ] = picX  
 else  
 RefPicSetLtFoll[ i ] = "no reference picture"  
if(offsetLayerId) {  
 for( i = 0; i < NumPocLtCurr; i++ )  
 if( !CurrDeltaPocMsbPresentFlag[ i ] )  
 if( there is a reference picture picX in the DPB with slice\_pic\_order\_cnt\_lsb equal to PocLtCurr[ i ] and nuh\_layer\_id equal to currPicLayerId)  
 RefPicSetLtCurr[ i  +  NumPocLtCurr] = picX  
 else  
 RefPicSetLtCurr[ i  +  NumPocLtCurr] = "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 +  NumPocLtCurr] = picX  
 else  
 RefPicSetLtCurr[ i +  NumPocLtCurr] = "no reference picture" (F‑34)  
 for( i = 0; i < NumPocLtFoll; i++ )  
 if( !FollDeltaPocMsbPresentFlag[ i ] )  
 if( there is a reference picture picX in the DPB with slice\_pic\_order\_cnt\_lsb equal to PocLtFoll[ i ] and nuh\_layer\_id equal to currPicLayerId)  
 RefPicSetLtFoll[ i  + NumPocLtFoll] = picX  
 else  
 RefPicSetLtFoll[ i + NumPocLtFoll] = "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 + NumPocLtFoll] = picX  
 else  
 RefPicSetLtFoll[ i + NumPocLtFoll] = "no reference picture"  
}

1. All reference pictures that are included in RefPicSetLtCurr and RefPicSetLtFoll and with 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+ offsetPicLayerId)  
 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+ offsetPicLayerId)  
 RefPicSetStCurrAfter[ i ] = picX  
 else  
 RefPicSetStCurrAfter[ i ] = "no reference picture" (F‑35)

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+ offsetPicLayerId)  
 RefPicSetStFoll[ i ] = picX  
 else  
 RefPicSetStFoll[ i ] = "no reference picture"  
if(offsetPicLayerId){  
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 + NumPocStCurrBefore] = picX  
 else  
 RefPicSetStCurrBefore[ i + NumPocStCurrBefore] = "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  + NumPocStCurrBefore] = picX  
 else  
 RefPicSetStCurrAfter[ i + NumPocStCurrBefore] = "no reference picture" (F‑35)

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 + NumPocStCurrBefore] = picX  
 else  
 RefPicSetStFoll[ i + NumPocStCurrBefore] = "no reference picture"  
}

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

NOTE  – 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  – 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 subclause 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 subclause 8.3.3
  + A picture that precedes the associated IRAP picture in decoding order
* When the 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.

* + 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

The specifications in clause 10 apply.

* 1. (Void)
  2. Byte stream format

The specifications in Annex B apply.

* 1. Hypothetical reference decoder
     1. General

The specifications in subclause C.1 apply.

* + 1. Operation of coded picture buffer (CPB)

The specifications in subclause C.2 apply with the following modifications.

– Replace "a BLA access unit for which the coded picture has nal\_unit\_type equal to BLA\_W\_RADL or BLA\_N\_LP" with "a BLA access unit for which each coded picture has nal\_unit\_type equal to BLA\_W\_RADL or BLA\_N\_LP".

– Replace "a BLA access unit for which the coded picture has nal\_unit\_type equal to BLA\_W\_LP" with "a BLA access unit for which each coded picture has nal\_unit\_type equal to BLA\_W\_LP".

– Replace "picture n" with "access unit n".

– Replace "AuNominalRemovalTime[ prevNonDiscardablePic ] is the nominal removal time of the preceding picture in decoding order with TemporalId equal to 0 that is not a RASL, RADL or sub-layer non-reference picture", with "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".

* + 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 contains picture storage buffers. Each of the picture storage buffers may contain a decoded picture that is marked as "used for reference" or is held for future output. The processes specified in subclauses F.13.3.2, F.13.3.3 and F.13.3.4 are sequentially applied as specified below.

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

Decoded pictures with the same DPB output time and with PicOutputFlag equal to 1 are output in ascending order of nuh\_layer\_id values of these decoded pictures.

Let picture n 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.

* + - 1. Removal of pictures from the DPB

The specifications in subclause C.3.2 apply separately for each set of decoded pictures with a particular value of nuh\_layer\_id with the following modifications.

– Replace "The removal of pictures from the DPB before decoding of the current picture (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 access unit n (containing the current picture) and proceeds as follows:" with "The removal of pictures from the DPB before decoding of the current picture (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 picture n and proceeds as follows:".

* + - 1. Picture output

The specifications in subclause C.3.3 apply with the following modifications.

– Replace "The output of the current picture is specified as follows, " with "For each picture of the current access unit, the output of the picture is specified as follows for each picture of the access unit ".

* + - 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 picture n, CpbRemovalTime[ n ].

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

* + 1. Bitstream conformance

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

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

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

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

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

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.

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 ] ) (F‑36)

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 ] ) (F‑37)

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

Floor( deltaTime90k[ n ] ) <= InitCpbRemovalDelay[ SchedSelIdx ] <= Ceil( deltaTime90k[ n ] ) (F‑38)

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. 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.
5. For each current picture, after invocation of the process for removal of pictures from the DPB as specified in subclause C.3.2, the number of decoded pictures in the DPB, including all pictures n that are marked as "used for reference", or that have PicOutputFlag equal to 1 and AuCpbRemovalTime[ n ] less than AuCpbRemovalTime[ currPic ], where currPic is the current picture, shall be less than or equal to sps\_max\_dec\_pic\_buffering\_minus1[ HighestTid ].
6. 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.
7. For each current picture, the value of maxPicOrderCnt − minPicOrderCnt shall be less than MaxPicOrderCntLsb / 2.
8. The value of DpbOutputInterval[ n ] as given by Equation C‑17, 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.
9. 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++ ) (F‑39)  
 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. Decoder conformance
       1. General

The specifications in subclause C.5.1 apply.

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

The decoded picture buffer contains picture storage buffers. The number of picture storage buffers for nuh\_layer\_id equal to 0 is derived from the active SPS. The number of picture storage buffers for each non-zero nuh\_layer\_id value is derived from the active layer SPS for that non-zero nuh\_layer\_id value. Each of the picture storage buffers 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 F.13.5.2.2 is invoked, followed by the invocation of the process for picture decoding, marking, additional bumping, and storage as specified in subclause F.13.5.2.3. The "bumping" process is specified in subclause F.13.5.2.4 and is invoked as specified in subclauses F.13.5.2.2 and F.13.5.2.3.

Let picture n 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.

* + - * 1. Output and removal of pictures from the DPB

The output and removal of pictures from the DPB before the decoding of the current picture (but after parsing the slice header of the first slice of the current picture) happens instantaneously when the first decoding unit of the current picture is removed from the CPB and proceeds as follows:

The decoding process for RPS as specified in subclause F.8.3.2 is invoked to mark only the pictures with the same value of nuh\_layer\_id.

– If the current picture is an IRAP picture with NoRaslOutputFlag equal to 1 and with nuh\_layer\_id equal to 0 that is not picture 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, or sps\_max\_dec\_pic\_buffering\_minus1[ HighestTid ] derived from the active SPS is different from the value of pic\_width\_in\_luma\_samples, pic\_height\_in\_luma\_samples, or sps\_max\_dec\_pic\_buffering\_minus1[ HighestTid ], respectively, derived from the SPS active for the preceding picture, NoOutputOfPriorPicsFlag may (but should not) be set to 1 by the decoder under test, regardless of the value of no\_output\_of\_prior\_pics\_flag.

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

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

– Otherwise (NoOutputOfPriorPicsFlag is equal to 0), all picture storage buffers containing a picture that is marked as "not needed for output" and "unused for reference" are emptied (without output), and all non-empty picture storage buffers in the DPB are emptied by repeatedly invoking the "bumping" process specified in subclause F.13.5.2.4, and the DPB fullness is set equal to 0.

– Otherwise (the current picture is not an IRAP picture with NoRaslOutputFlag equal to 1 or with nuh\_layer\_id not equal to 0), all picture storage buffers containing a picture which are marked as "not needed for output" and "unused for reference" are emptied (without output). For each picture storage buffer that is emptied, the DPB fullness is decremented by one. The variable currLayerId is set equal to nuh\_layer\_id of the current decoded picture and when one or more of the following conditions are true, the "bumping" process specified in subclause F.13.5.2.4 is invoked repeatedly while further decrementing the DPB fullness by one for each additional picture storage buffer that is emptied, until none of the following conditions are true:

– The number of pictures with nuh\_layer\_id equal to currLayerId in the DPB that are marked as "needed for output" is greater than sps\_max\_num\_reorder\_pics[ HighestTid ] from the active SPS (when currLayerId is equal to 0) or from the active layer SPS for the value of currLayerId (when currLayerId is not equal to 0).

– sps\_max\_latency\_increase\_plus1[ HighestTid ] of the active SPS (when currLayerId is equal to 0) or the active layer SPS for the value of currLayerId is not equal to 0 and there is at least one picture with nuh\_layer\_id equal to currLayerId in the DPB that is marked as "needed for output" for which the associated variable PicLatencyCount[ currLayerId ] is greater than or equal to SpsMaxLatencyPictures[ HighestTid ] derived from the active SPS (when currLayerId is equal to 0) or from the active layer SPS for the value of currLayerId.

– The number of pictures with nuh\_layer\_id equal to currLayerId in the DPB is greater than or equal to sps\_max\_dec\_pic\_buffering\_minus1[ HighestTid ] + 1 from the active SPS (when currLayerId is equal to 0) or from the active layer SPS for the value of currLayerId.

* + - * 1. Picture decoding, marking, additional bumping, and storage

The processes specified in this subclause happen instantaneously when the last decoding unit of access unit n containing the current picture is removed from the CPB.

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

For each picture in the DPB that is marked as "needed for output" and that has a nuh\_layer\_id value equal to currLayerId, the associated variable PicLatencyCount[ currLayerId ] is set equal to PicLatencyCount[ currLayerId ] + 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 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[ currLayerId ] 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 F.13.5.2.4 is invoked repeatedly until none of the following conditions are true:

– The number of pictures with nuh\_layer\_id equal to currLayerId in the DPB that are marked as "needed for output" is greater than sps\_max\_num\_reorder\_pics[ HighestTid ] from the active SPS (when currLayerId is equal to 0) or from the active layer SPS for the value of currLayerId, if not equal to 0.

– sps\_max\_latency\_increase\_plus1[ HighestTid ] is not equal to 0 and there is at least one picture with nuh\_layer\_id equal to currLayerId in the DPB that is marked as "needed for output" for which the associated variable PicLatencyCount[ currLayerId ] that is greater than or equal to SpsMaxLatencyPictures[ HighestTid ] derived from the active SPS (when currLayerId is equal to 0) or from the active layer SPS for the value of currLayerId (when currLayerId is not equal to 0).

* + - * 1. "Bumping" process

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

1. The 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. These pictures are cropped, using the conformance cropping window specified in the active SPS for the picture with nuh\_layer\_id equal to 0 or in the active layer SPS for a non-zero nuh\_layer\_id value equal to that of the picture, the cropped pictures are output in ascending order of nuh\_layer\_id, and the pictures are marked as "not needed for output".
3. Each picture storage buffer that contains a picture marked as "unused for reference" and that was one of the pictures cropped and output is emptied.
   1. SEI messages

The specifications in Annex D together with the extensions and modifications specified in this subclause apply.

* + 1. SEI message syntax
       1. Layers not present SEI message syntax

|  |  |
| --- | --- |
| layers\_not\_present( payloadSize ) { | **Descriptor** |
| **lp\_sei\_active\_vps\_id** | u(4) |
| for( i = 0; i <= vps\_max\_layers\_minus1; 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. SEI message semantics

Table F‑2 – 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 change SEI message or the end of the CVS, whichever is earlier in decoding order |
| Inter-layer constrained tile sets | The CVS containing the SEI message |

* + - 1. Layers not present SEI message semantics

The layers not present SEI message provides a mechanism for signalling that VCL NAL units of particular layers indicated by the VPS are not present in a particular set of access units.

The target access units are defined as the set of access units starting from the access unit containing the layers not present SEI message up to but not including the next access unit, in decoding order, that contains a layers not present change SEI message or the end of the CVS, whichever is earlier in decoding order.

When present, the layers not present SEI message applies to the target access units.

A layers not present SEI message shall not be included in a scalable nesting SEI message.

A layers not present SEI message shall not be included in an SEI NAL unit with TemporalId greater than 0.

**lp\_sei\_active\_vps\_id** identifies the active VPS of the CVS containing the layers not present SEI message. The value of lp\_sei\_active\_vps\_id shall be equal to the value of vps\_video\_parameter\_set\_id of the active VPS for the VCL NAL units of the access unit containing the SEI message.

**layer\_not\_present\_flag**[ i ] equal to 1 indicates that there are no VCL NAL units with nuh\_layer\_id equal to layer\_id\_in\_nuh[ i ] present in the target access units. layer\_not\_present\_flag[ i ] equal to 0 indicates that there may or may not be VCL NAL units with nuh\_layer\_id equal to layer\_id\_in\_nuh[ i ] present in the target access units.

When layer\_not\_present\_flag[ i ] is equal to 0 and i is greater than 0, layer\_not\_present\_flag[ LayerIdxInVps[ RefLayerId[ layer\_id\_in\_nuh[ i ] ][ j ] ] ] shall be equal to 0 for all values of j in the range of 0 to NumDirectRefLayers[ layer\_id\_in\_nuh[ i ] ] − 1, inclusive.

* + - 1. Inter-layer constrained tile sets SEI message semantics

The scope of the inter-layer constrained tile sets SEI message is the complete CVS. When an inter-layer tile sets SEI message is present in any access unit of a CVS, it shall be present for the first access unit of the CVS in decoding order and may also be present for other access units of the CVS.

The inter-layer constrained tile sets SEI message shall not be present for a layer when tiles\_enabled\_flag is equal to 0 for any PPS that is active in the layer.

The inter-layer constrained tile sets SEI message shall not be present for a layer unless every PPS that is active for the layer has tile\_boundaries\_aligned\_flag equal to 1 or fulfills the conditions that would be indicated by tile\_boundaries\_aligned\_flag being equal to 1.

The presence of the inter-layer tile sets SEI message indicates that the inter-layer inter prediction process is constrained such that no sample value outside each identified tile set, and no sample value at a fractional sample position that is derived using one or more sample values outside the identified tile set, is used for inter prediction of any sample within the identified tile set.

NOTE  – 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. If 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 message. skipped\_tile\_set\_present\_flag equal to 0 does not indicate a bitstream constraint within the CVS. When not present, the value of skipped\_tile\_set\_present\_flag is inferred to be equal to 0.

**ilcts\_id**[ i ] contains an identifying number that may be used to identify the purpose of the i-th identified tile set (for example, to identify an area to be extracted from the coded video sequence for a particular purpose). The value of ilcts\_id[ i ] shall be in the range of 0 to 232 − 2, inclusive.

Values of ilcts\_id[ i ] from 0 to 255 and from 512 to 231 − 1 may be used as determined by the application. Values of ilcts\_id[ i ] from 256 to 511 and from 231 to 232 − 2 are reserved for future use by ITU-T | ISO/IEC. Decoders encountering a value of ilcts\_id[ i ] in the range of 256 to 511 or in the range of 231 to 232 − 2 shall ignore (remove from the bitstream and discard) it.

**il\_num\_tile\_rects\_in\_set\_minus1**[ i ] plus 1 specifies the number of rectangular regions of tiles in the i-th identified inter-layer constrained tile set. The value of il\_num\_tile\_rects\_in\_set\_minus1[ i ] shall be in the range of 0 to (num\_tile\_columns\_minus1 + 1) \* (num\_tile\_rows\_minus1 + 1) − 1, inclusive.

**il\_top\_left\_tile\_index**[ i ][ j ] and **il\_bottom\_right\_tile\_index**[ i ][ j ] identify the tile position of the top-left tile and the tile position of the bottom-right tile in a rectangular region of the i-th identified inter-layer constrained tile set, respectively, in tile raster scan order.

**il\_exact\_sample\_value\_match\_flag**[ i ] equal to 1 indicates that, within the CVS, when the coding tree blocks that do not belong to the inter-layer constrained tile set are not decoded and the boundaries of the inter-layer constrained tile set are treated as picture boundaries for purposes of the decoding process, the value of each sample in the inter-layer constrained tile set would be exactly the same as the value of the sample that would be obtained when all the coding tree blocks of all pictures in the coded video sequence are decoded. il\_exact\_sample\_value\_match\_flag[ i ] equal to 0 indicates that, within the CVS, when the coding tree blocks that are outside of the i-th identified inter-layer constrained tile set are not decoded and the boundaries of the inter-layer constrained tile set are treated as picture boundaries for purposes of the decoding process, the value of each sample in the identified tile set may or may not be exactly the same as the value of the same sample when all the coding tree blocks of the picture are decoded.

NOTE  – It should be feasible to use il\_exact\_sample\_value\_match\_flag equal to 1 when using certain combinations of loop\_filter\_across\_tiles\_enabled\_flag, pps\_loop\_filter\_across\_slices\_enabled\_flag, pps\_deblocking\_filter\_disabled\_flag, slice\_loop\_filter\_across\_slices\_enabled\_flag, slice\_deblocking\_filter\_disabled\_flag, sample\_adaptive\_offset\_enabled\_flag, slice\_sao\_luma\_flag, and slice\_sao\_chroma\_flag.

**ilc\_idc**[ i ] equal to 1 indicates that, within the CVS, no samples outside of the i-th identified tile set and no samples at a fractional sample position that is derived using one or more samples outside of the i-th identified tile set are used for inter-layer prediction of any sample within the i-th identified tile set with nuh\_layer\_id equal to ictsNuhLayerId, where ictsNuhLayerId is the value of nuh\_layer\_id of this message. ilc\_idc[ i ][ j ] equal to 2 indicates that, within the CVS, no prediction block in the i-th identified tile set with nuh\_layer\_id equal to ictsNuhLayerId is predicted from an inter-layer reference picture. ilc\_idc[ i ] equal to 0 indicates that, within the CVS, the inter-layer prediction process may or may not be constrained for the prediction block in the i-th identified tile set having nuh\_layer\_id equal to ictsNuhLayerId. The value of ilc\_idc[ i ] equal to 3 is reserved.

**all\_tiles\_ilc\_idc** equal to 1 indicates that, within the CVS, no sample value outside of each identified tile and no sample value at a fractional sample position that is derived using one or more samples outside of the identified tile is used for inter-layer prediction of any sample within the identified tile with nuh\_layer\_id equal to ictsNuhLayerId, where ictsNuhLayerId is the value of nuh\_layer\_id of this message. all\_tiles\_ilc\_idc equal to 2 indicates that, within the CVS, no prediction block in each identified tile with nuh\_layer\_id equal to ictsNuhLayerId is predicted from an inter-layer reference picture. all\_tiles\_ilc\_idc equal to 0 indicates that, within the CVS, the inter-layer prediction process may or may not be constrained for the tile having nuh\_layer\_id equal to ictsNuhLayerId. The value of all\_tiles\_ilc\_idc equal to 3 is reserved.

* 1. Video usability information
     1. General

The specifications in clause E.1 apply.

* + 1. VUI syntax

The specifications in clause E.2 apply.

* + 1. VUI semantics
       1. VUI parameters semantics

The specifications in clause E.3.1 apply with the following modifications and additions.

vui\_timing\_info\_present\_flag equal to 1 specifies that vui\_num\_units\_in\_tick, vui\_time\_scale, vui\_poc\_proportional\_to\_timing\_flag, and vui\_hrd\_parameters\_present\_flag are present in the vui\_parameters( ) syntax structure. vui\_timing\_info\_present\_flag equal to 0 specifies that vui\_num\_units\_in\_tick, vui\_time\_scale, vui\_poc\_proportional\_to\_timing\_flag, and vui\_hrd\_parameters\_present\_flag are not present in the vui\_parameters( ) syntax structure. It is a requirement of bitstream conformance that, when nuh\_layer\_id is greater than 0, vui\_timing\_info\_present\_flag shall be equal to 0.

* + - 1. HRD parameters semantics

The specifications in clause E.3.2 apply.

* + - 1. Sub-layer HRD parameters semantics

The specifications in clause E.3.3 apply.

1. Annex H   
     
   Syntax, semantics and decoding processes for scalable extension

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

This annex specifies syntax, semantics and decoding processes,for scalable exetnsion that use the syntax, semantics, and decoding process specified in clauses 2-9 and Annex A-F.

* 1. Scope

Decoding process and bitstreams conforming to this annex are completely specified in this annex with reference made to clauses 2-9 and Annexes A-F.

* 1. Normative references

The specifications in clause 2 apply.

* 1. Definitions

The specifications in clause F.3 apply.

* 1. Abbreviations

The specifications in clause 4 apply.

* 1. Conventions

The specifications in clause 5 apply.

* 1. Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships
     1. Derivation process for reference layer sample location

The specification in clause 6 and all its subclauses apply with the following additions.

Input to this process is a luma location ( xP, yP ) relative to the top-left luma sample of the current picture.

Output of this process is a luma location ( xRef, yRef ) relative to the top-left luma sample of the reference layer picture.

The variables xRef and yRef are derived as follows:

xRef = ( ( xP ‑ ScaledRefLayerLeftOffset ) \* ScaleFactorX + ( 1 << 15 ) ) >> 16 (H‑1)

yRef = ( ( yP ‑ ScaledRefLayerTopOffset ) \* ScaleFactorY + ( 1 << 15 ) ) >> 16 (H‑2)

* + 1. Derivation process for reference layer sample location used in resampling

Inputs to this process are

– a variable cIdx specifying the color component index,

– a sample location ( xP, yP ) relative to the top-left sample of the color component of the current picture specified by cIdx.

Output of this process is a sample location ( xRef16, yRef16 ) specifying the reference layer sample location in units of 1/16-th sample relative to the top-left sample of the reference layer picture.

The variables offsetX and offsetY are derived as follows:

offsetX = ScaledRefLayerLeftOffset / ( ( cIdx = = 0)  ?  1 :  SubWidthC) (H‑3)  
offsetY = ScaledRefLayerTopOffset / ( ( cIdx = = 0)  ?  1 :  SubHeightC) (H‑4)

The variables phaseY and addY are derived as follows:

phaseY = (cIdx = = 0) ? 0 : 1 (H‑5)

addY = ( ScaleFactorY \* phaseY + 2 ) >> 2 (H‑6)

The variables xRef16 and yRef16 are derived as follows:

xRef16 = ( ( ( xP – offsetX ) \* ScaleFactorX  + ( 1 << 11 ) ) >> 12 )  (H‑7)  
yRef16 = ( ( ( yP – offsetY ) \* ScaleFactorY + addY + ( 1 << 11 ) ) >> 12 ) –  ( phaseY << 2 ) (H‑8)

* 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.8.1apply.

* + - 1. Decoding process for a coded picture with nuh\_layer\_id greater than 0

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

1. The decoding of NAL units is specified in subclause 8.2.
2. The processes in subclause H.8.1.2 and H.8.3.4 specify the following decoding processes using syntax elements in the slice segment layer and above:

– Prior to decoding the first slice of the current picture, subclause H.8.1.2 is invoked.

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

1. The processes in subclauses H.8.4, H.8.5, H.8.6, and H.8.7 specify decoding processes using syntax elements in all syntax structure layers. It is a requirement of bitstream conformance that the coded slices of the picture shall contain slice segment data for every coding tree unit of the picture, such that the division of the picture into slices, the division of the slices into slice segments, and the division of the slice segments into coding tree units each form a partitioning of the picture.
2. After all slices of the current picture have been decoded, the marking process for ending the decoding of a coded picture with nuh\_layer\_id greater than 0 specified in subclause H.8.1.3 is invoked.
   * + 1. Decoding process for inter-layer reference picture set

Outputs of this process are updated lists of inter-layer reference pictures RefPicSetInterLayer0 and RefPicSetInterLayer1 and the variables NumActiveRefLayerPics0 and NumActiveRefLayerPics1.

The variable currLayerId is set equal to nuh\_layer\_id of the current decoded picture

The lists RefPicSetInterLayer0 and RefPicSetInterLayer1 are first emptied, NumActiveRefLayerPics0 and NumActiveRefLayerPics1 are set equal to 0 and the following applies:

for( i = 0; i < NumActiveRefLayerPics; i++ ) {  
 if( there is a picture picX in the DPB that is in the same access unit as the current picture and has  
 nuh\_layer\_id equal to RefPicLayerId[ i ] ) {  
 an interlayer reference picture rsPic is derived by invoking the subclause H.8.1.4 with picX and  
 RefPicLayerId[ i ] given as inputs   
 if( ( ViewId[ nuh\_layer\_id ] <= ViewId[ 0 ]  &&  
 ViewId[ nuh\_layer\_id ] <= ViewId[ RefPicLayerId[ i ] ] ) | |  
 ( ViewId[ nuh\_layer\_id ] >= ViewId[ 0 ] &&  
 ViewId[ nuh\_layer\_id ] >= ViewId[ RefPicLayerId[ i ] ] ) ) {  
 RefPicSetInterLayer0[ NumActiveRefLayerPics0 ] = rsPic  
 RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] is marked as "used for long-term reference"  
 } else {  
 RefPicSetInterLayer1[ NumActiveRefLayerPics1 ] = rsPic  
 RefPicSetInterLayer1[ NumActiveRefLayerPics1++ ] is marked as "used for long-term reference"  
 }  
 } else  
 RefPicSetInterLayer0[ NumActiveRefLayerPics0++ ] = "no reference picture"  
}

There shall be no entry equal to "no reference picture" in RefPicSetInterLayer0 or RefPicSetInterLayer1.

NOTE – For the profiles defined in Annex H, RefPicSetInterLayer1 is always empty since the value of ViewId[ i ] is equal to zero for all layers.

If the current picture is a RADL picture, there shall be no entry in the RefPicSetInterLayer0 or RefPicSetInterLayer1 that is a RASL picture.

NOTE – An access unit may contain both RASL and RADL pictures.

* + - 1. Marking process for ending the decoding of a coded picture with nuh\_layer\_id greater than 0

Output of this process is:

– a potentially updated marking as "used for short-term reference" for some decoded pictures.

The following applies.

for( i = 0; i < NumActiveRefLayerPics0; i++ )   
 RefPicSetInterLayer0[ i ] is marked as "used for short-term reference"

for( i = 0; i < NumActiveRefLayerPics1; i++ )  
 RefPicSetInterLayer1[ i ] is marked as "used for short-term reference"

* + - 1. Resampling process for inter layer reference pictures

Input to this process is:

– a decoded reference layer picture rlPic

– a variable rLId specifies thelayer id of reference layer picture .

Output of this process is the resampled reference layer picture rsPic.

The variables PicWidthInSamplesL and PicHeightInSamplesL are set equal to pic\_width\_in\_luma\_samples and pic\_height\_in\_luma\_samples, respectively.

The variables RefLayerPicWidthInSamplesL and RefLayerPicHeightInSamplesL are set equal to the width and height of the decoded reference layer picture rlPic in units of luma samples, respectively.

The variables PicWidthInSamplesC, PicHeightInSamplesC, RefLayerPicWidthInSamplesC, and RefLayerPicHeightInSamplesC are derived as follows:

PicWidthInSamplesC = PicWidthInSamplesL / subWidthC (H‑9)  
PicHeightInSamplesC = PicHeightInSamplesL / subHeightC (H‑10)  
RefLayerPicWidthInSamplesC = RefLayerPicWidthInSamplesL / subWidthC (H‑11)  
RefLayerPicHeightInSamplesC = RefLayerPicHeightInSamplesL / subHeightC (H‑12)

The variable currLayerId is set equal to nuh\_layer\_id of the current picture. The variable dRlIdx is set equal to DirectRefLayerIdx[ currLayerId ][ rLId ].

The variables ScaledRefLayerLeftOffset, ScaledRefLayerTopOffset, ScaledRefLayerRightOffset and ScaledRefLayerBottomOffset are derived as follows:

ScaledRefLayerLeftOffset = scaled\_ref\_layer\_left\_offset[ dRlIdx ] << 1 (H‑13)  
ScaledRefLayerTopOffset = scaled\_ref\_layer\_top\_offset[ dRlIdx] << 1 (H‑14)  
ScaledRefLayerRightOffset = scaled\_ref\_layer\_right\_offset[ dRlIdx ] << 1 (H‑15)  
ScaledRefLayerBottomOffset = scaled\_ref\_layer\_bottom\_offset[ dRlIdx ] << 1 (H‑16)

The variables ScaledRefLayerPicWidthInSamplesL and ScaledRefLayerPicHeightInSamplesL are derived as follows:

ScaledRefLayerPicWidthInSamplesL = PicWidthInSamplesL –   
 ScaledRefLayerLeftOffset  – ScaledRefLayerRightOffset (H‑17)  
ScaledRefLayerPicHeightInSamplesL = PicHeightInSamplesL –   
 ScaledRefLayerTopOffset – ScaledRefLayerBottomOffset (H‑18)

The variables ScaleFactorX and ScaleFactorY are derived as follows:

ScaleFactorX = ( ( RefLayerPicWidthInSamplesL << 16 ) + ( ScaledRefLayerPicWidthInSamplesL >> 1 ) ) / ScaledRefLayerPicWidthInSamplesL (H‑19)  
ScaleFactorY = ( ( RefLayerPicHeightInSamplesL << 16 ) + ( ScaledRefLayerPicHeightInSamplesL >> 1 ) ) / ScaledRefLayerPicHeightInSamplesL (H‑20)

The following steps are applied to derive the resampled inter layer reference picture rsPic.

– if PicWidthInSamplesL is equal to RefLayerPicWidthInSamplesL and PicHeightInSamplesL is equal to RefLayerPicHeightInSamplesL and the values of ScaledRefLayerLeftOffset, ScaledRefLayerTopOffset, ScaledRefLayerRightOffset and ScaledRefLayerBottomOffset are all equal to 0

* + rsPic is set equal to rlPic.

– otherwise, rsPic is derived as follows:

* + The PicOrderCntVal value of rsPic is set equal to the PicOrderCntVal value of rlPic.
  + When SamplePredEnabledFlag[ currLayerId ][ rLId ] is equal to 1, the picture sample resampling process as specified in subclause H.8.1.4.1 is invoked with the sample arrays of reference layer picture rlPic as input, and with the sample arrays of resampled picture rsPic as output.
  + When MotionPredEnabledFlag[ currLayerId ][ rLId ] is equal to 1, the picture motion field resampling process as specified in subclause H.8.1.4.2 is invoked with refereence layer picture rlPic and its motion field as inputs, and with the motion field of resampled picture rsPic as output.
    - * 1. Resampling process of picture sample values

Inputs to this process are:

– a ( RefLayerPicWidthInSamplesL ) x ( RefLayerPicHeightInSamplesL ) 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 ( PicWidthInSamplesL ) x ( PicHeightInSamplesL ) array rsPicSampleL of luma samples,

– a (  PicWidthInSamplesC ) x ( PicHeightInSamplesC ) array rsPicSampleCb of chroma samples of the component Cb,

– a (  PicWidthInSamplesC) x ( PicHeightInSamplesC ) array rsPicSampleCr of chroma samples of the component Cr.

The luma sample array rsPicSampleL is derived by invoking the luma sample resampling process specified in subclause H.8.1.4.1.1 with the reference luma sample array rlPicSampleL given as input.

The chroma sample array rsPicSampleCb of the chroma component Cb is derived by invoking the chroma sample resampling process specified in subclause H.8.1.4.1.2 with the reference chroma sample array rlPicSampleCb given as input.

The chroma sample array rsPicSampleCr of the chroma component Cr is derived by invoking the chroma sample resampling process specified in subclause H.8.1.4.1.2 with the reference sample array rlPicSampleCr given as input.

Resampling process of luma sample values

Input to this process is the reference luma sample array rlPicSampleL.

Output of this process is the resampled luma sample array rsPicSampleL.

The variables leftStartL, rightEndL, topStartL, and bottomEndL are derived as follows:

leftStartL = ScaledRefLayerLeftOffset  
rightEndL = PicWidthInSamplesL – ScaledRefLayerRightOffset  
topStartL = ScaledRefLayerTopOffset  
bottomEndL = PicHeightInSamplesL – ScaledRefLayerBottomOffset

The luma samples rsPicSampleL [ xP ][ yP ] with ( xP = 0 ... PicWidthInSamplesL – 1, yP = 0 ... PicHeightInSamplesL – 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 = ( PicWidthInSamplesL– ScaledRefLayerRightOffset ) / SubWidthC  
topStartC = ScaledRefLayerTopOffset / SubHeightC  
bottomEndC = ( PicHeightInSamplesL– ScaledRefLayerBottomOffset ) / SubHeightC

The chroma samples rsPicSampleC[ xPC ][ yPC ] with ( xPC = 0 ... PicWidthInSamplesC – 1, yPC = 0 ... PicHeightInSamplesC – 1) are derived by invoking the chroma sample interpolation process specified in subclause H.8.1.4.1.4 with rlPicSampleC and chroma sample location ( Clip3( leftStartC, rightEndC ‑ 1, xPC ), Clip3( topStartC, bottomEndC – 1, yPC ) ) given as inputs and rsPicSampleC[ xPC ][ yPC ] as output.

Luma sample interpolation process

Inputs to this process are

– the luma reference sample array rlPicSampleL,

– a luma sample location ( xP, yP ) relative to the top-left luma sample of the current picture.

Output of this process is a interpolated luma sample value intLumaSample.

Table H‑1 specifies the 8-tap filter coefficients fL[ p, x ] with p = 0 ... 15 and x = 0 ... 7 used for the luma resampling process.

Table H‑1 – 16-phase luma resampling filter

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| phase p | interpolation filter coefficients | | | | | | | |
| fL[ p, 0 ] | fL[ p, 1 ] | fL[ p, 2 ] | fL[ p, 3 ] | fL[ p, 4 ] | fL[ p, 5 ] | fL[ p, 6 ] | fL[ p, 7 ] |
| 0 | 0 | 0 | 0 | 64 | 0 | 0 | 0 | 0 |
| 1 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a |
| 2 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a |
| 3 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a |
| 4 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a |
| 5 | −1 | 4 | -11 | 52 | 26 | -8 | 3 | -1 |
| 6 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a |
| 7 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a |
| 8 | −1 | 4 | -11 | 40 | 40 | -11 | 4 | -1 |
| 9 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a |
| 10 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a |
| 11 | −1 | 3 | -8 | 26 | 52 | -11 | 4 | -1 |
| 12 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a |
| 13 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a |
| 14 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a |
| 15 | n/a | n/a | n/a | n/a | n/a | n/a | n/a | n/a |

The value of the interpolated luma sample IntLumaSample  is derived by applying the following ordered steps:

1. The derivation process for reference layer sample location used in resampling as specified in subclause H.6.2 is invoked with cIdx equal to 0 and luma sample location ( xP, yP ) given as the inputs and ( xRef16, yRef16 ) in units of 1/16-th sample as output.
2. The variables xRef and xPhase are derived as follows:

xRef     = ( xRef16 >> 4 ) (H‑21)

xPhase = ( xRef16 ) % 16 (H‑22)

1. The variables yRef and yPhase are derived as follows

yRef     = ( yRef16 >> 4 ) (H‑23)

yPhase = ( yRef16 ) % 16 (H‑24)

1. The variables shift1, shift2 and offset are derived as follows:

shift1 = BitDepthY – 8 (H‑25)

shift2 = 20 – BitDepthY (H‑26)

offset =1 << ( shift2 – 1). (H‑27)

1. The sample value tempArray[ n ] with n = 0 … 7, is derived as follows.

yPosRL = Clip3( 0, RefLayerPicHeightInSamplesL – 1, yRef + n – 1 ) (H‑28)

refW      = RefLayerPicWidthInSamplesL

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‑29)  
 fL[ xPhase, 5 ] \* rlPicSampleL[ Clip3( 0, refW– 1, xRef + 2), yPosRL ] +  
 fL[ xPhase, 6 ] \* rlPicSampleL[ Clip3( 0, refW– 1, xRef + 3), yPosRL ] +  
 fL[ xPhase, 7 ] \* rlPicSampleL[ Clip3( 0, refW– 1, xRef + 4), yPosRL ] ) >> shift1

1. The interpolated luma sample value intLumaSample is derived as follows.

intLumaSample = ( fL[ yPhase, 0 ] \* tempArray [ 0 ] +   
 fL[ yPhase, 1 ] \* tempArray [ 1 ] +   
 fL[ yPhase, 2 ] \* tempArray [ 2 ] +   
 fL[ yPhase, 3 ] \* tempArray [ 3 ] +   
 fL[ yPhase, 4 ] \* tempArray [ 4 ] + (H‑30)  
 fL[ yPhase, 5 ] \* tempArray [ 5 ] +   
 fL[ yPhase, 6 ] \* tempArray [ 6 ] +   
 fL[ yPhase, 7 ] \* tempArray [ 7 ] + offset ) >> shift2

intLumaSample = Clip3( 0, ( 1 << BitDepthY) – 1 , intLumaSample ) (H‑31)

Chroma sample interpolation process

Inputs to this process are:

– the chroma reference sample array rlPicSampleC,

– a chroma sample location ( xPC, yPC ) relative to the top-left chorma sample of the current picture.

Output of this process is a interpolated chroma sample value intChromaSample.

Table H‑2 specifies the 4-tap filter coefficients fC[ p, x ] with p = 0 ... 15 and x = 0 ... 3 used for the chroma resampling process.

Table H‑2 – 16-phase chroma resampling filter

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| phase p | interpolation filter coefficients | | | |
| fC[ p, 0 ] | fC[ p, 1 ] | fC[ p, 2 ] | fC[ p, 3 ] |
| 0 | 0 | 64 | 0 | 0 |
| 1 | n/a | n/a | n/a | n/a |
| 2 | n/a | n/a | n/a | n/a |
| 3 | n/a | n/a | n/a | n/a |
| 4 | −4 | 54 | 16 | −2 |
| 5 | −6 | 52 | 20 | −2 |
| 6 | −6 | 46 | 28 | −4 |
| 7 | n/a | n/a | n/a | n/a |
| 8 | −4 | 36 | 36 | −4 |
| 9 | −4 | 30 | 42 | −4 |
| 10 | n/a | n/a | n/a | n/a |
| 11 | −2 | 20 | 52 | −6 |
| 12 | n/a | n/a | n/a | n/a |
| 13 | n/a | n/a | n/a | n/a |
| 14 | −2 | 10 | 58 | −2 |
| 15 | 0 | 4 | 62 | −2 |

The value of the interpolated chroma sample value intChromaSample is derived by applying the following ordered steps:

1. The derivation process for reference layer sample location in resampling as specified in subclause H.6.2 is invoked with cIdx and chroma sample location ( xPC, yPC ) given as the inputs and ( xRef16, yRef16 ) in units of 1/16-th sample as output.
2. The variables xRef and xPhase are derived by

xRef     = ( xRef16 >> 4 ) (H‑32)

xPhase = ( xRef16 ) % 16 (H‑33)

1. The variables yRef and yPhase are derived by

yRef     = ( yRef16 >> 4 ) (H‑34)

yPhase = ( yRef16 ) % 16 (H‑35)

1. The variables shift1, shift2 and offset are derived as follows:

shift1 = BitDepthC – 8 (H‑36)

shift2 = 20 – BitDepthC (H‑37)

offset =1 << ( shift2 – 1) (H‑38)

1. The sample value tempArray[ n ] with n = 0 … 3, is derived as follows.

yPosRL = Clip3( 0 , RefLayerPicHeightInSamplesC – 1, yRef + n – 1 ) (H‑39)

refWC   = RefLayerPicWidthInSamplesC (H‑40)

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‑41)  
 fC[ xPhase, 3 ] \* rlPicSampleC[ Clip3( 0, refWC – 1, xRef + 2), yPosRL ]  ) >> shift1

1. The interpolated chroma sample value intChromaSample is derived as follows.

intChromaSample = (fC[ yPhase, 0 ] \* tempArray [ 0 ] +  
 fC[ yPhase, 1 ] \* tempArray [ 1 ] +  
 fC[ yPhase, 2 ] \* tempArray [ 2 ] + (H‑42)  
 fC[ yPhase, 3 ] \* tempArray [ 3 ] + offset ) >> shift2

intChromaSample = Clip3( 0, ( 1 << BitDepthC ) – 1 , intChromaSample ) (H‑43)

* + - * 1. Resampling process of picture motion field

Inputs to this process are:

– the decoded reference layer picture rlPic,

– the variable rlPicMotion specifying the motion field of the reference layer picture rlPic,

Output of this process is rsPicMotion specifying the motion field of the resampled picture rsPic.

The motion field of rlPic specified by rlPicMotion consists of:

– a ( RefLayerPicWidthInSamplesL ) x ( RefLayerPicHeightInSamplesL ) array predModeRL specifies the prediction modes of the reference layer picture rlPic,

– two ( RefLayerPicWidthInSamplesL ) x ( RefLayerPicHeightInSamplesL ) arrays refIdxLXRL specify the reference indices of the reference layer picture rlPic, with X = 0,1,

– two ( RefLayerPicWidthInSamplesL ) x ( RefLayerPicHeightInSamplesL ) arrays mvLXRL specify the luma motion vectors of the reference layer picture rlPic, with X = 0,1,

– two ( RefLayerPicWidthInSamplesL ) x ( RefLayerPicHeightInSamplesL ) arrays predFlagLXRL specify the prediction list utilization flags of the reference layer picture rlPic, with X = 0,1.

The resampled motion field specified by rsPicMotion consists of:

– a ( PicWidthInSamplesL ) x ( PicHeightInSamplesL ) array predMode specifies the prediction modes of the resampled picture,

– two ( PicWidthInSamplesL ) x ( PicHeightInSamplesL ) arrays refIdxLX specify the reference indexes of the resampled picture, with X = 0,1,

– two ( PicWidthInSamplesL ) x ( PicHeightInSamplesL ) arrays mvLX  specify the luma motion vectors of the resampled picture, with X = 0,1,

* two ( PicWidthInSamplesL ) x ( PicHeightInSamplesL ) arrays predFlagLX specify the prediction list utilization flags of the resampled picture, with X = 0,1.

For each luma sample location xPb = 0 ... ( ( PicWidthInSamplesL + 15 ) >> 4 ) − 1 and yPb = 0 … ( ( PicHeightInSamplesL + 15 ) >> 4) − 1,

– The variables xP and yP are set to ( xPb  << 4 ) and ( yPb  << 4 ), respectively,

– The variables predMode[xP][yP], refIdxLX[xP][yP], mvLX[xP][yP] and predFlagLX[xP][yP], with X = 0,1, of the resampled picture are derived by invoking inter layer motion parameters derivation process specified in subclause H.8.1.4.2.1 with the luma location ( xP, yP ), rlPic , predModeRL, refIdxLXRL, mvLXRL and predFlagLXRL, with X = 0,1, given as input.

Derivation process for inter layer motion parameters

Inputs to this process are

– a luma location ( xP, yP ) specifying the top-left sample of the current luma prediction block relative to the top-left luma sample of the resampled picture,

– the decoded reference layer picture rlPic

– the reference layer prediction mode array predModeRL,

– the reference layer reference index arrays refIdxL0RL and refIdxL1RL

– the reference layer motion vector arrays mvL0RL and mvL1RL

– the reference layer prediction list utilization flag arrays predFlagL0RL and predFlagL1RL.

Outputs of this process are

– a derived prediction mode predMode,

– two derived motion vectors mvL0 and mvL1

– two derived reference indices refIdxL0 and refIdxL1

– two derived prediction list utilization flags predFlagL0 and predFlagL1.

The variables predMode, mvLX, refIdxLX, refPicOrderCntLX, and predFlagLX are derived as follows.

1. The center location (xPCtr, yPCtr) of the luma prediction block is derived as follows

xPCtr = xP + 8 (H‑44)

yPCtr = yP + 8 (H‑45)

1. The derivation process for reference layer luma sample location specified in subclause H.6.1 is invoked with luma location ( xPCtr, yPCtr ) given as the inputs and ( xRef, yRef ) as output.
2. The collocated position (xRL, yRL) is derived as follows:

xRL = ( ( xRef + 4 ) >> 4 ) << 4 (H‑46)

yRL = ( ( yRef + 4 ) >> 4 ) << 4 (H‑47)

1. The prediction mode predMode[ xP ][ yP ] is derived as follows:

* If ( xRL < 0 ) or ( xRL >= RefLayerPicWidthInSamplesL ) or ( yRL < 0 ) or ( yRL >= RefLayerPicHeightInSamplesL ), predMode[ xP ][ yP ] is set to MODE\_INTRA.
* Otherwise, the following applies:

predMode[ xP ][ yP ] = predModeRL[ xRL ][ yRL ] (H‑48)

1. The variables mvL0[ xP ][ yP ], mvL1[ xP ][ yP ], refIdxL0[ xP ][ yP ], refIdxL1[ xP ][ yP ], predFlagL0[ xP ][ yP ] and predFlagL1[ xP ][ yP ] are derived as follows:

* If predMode[ xP ][ yP ] is equal to MODE\_INTER, the following applies
* The variable colPb specifies the luma prediction block covering the location given by ( xRL, yRL ) inside the reference layer picture specified by rlPic. The 16 x 16 coding block with top-left luma location ( xP, yP ) relative to the top-left luma sample of the resampled picture is set associated with the slice containing the prediction block colPb inside the reference layer picture specified by rlPic.
* For each X = 0, 1, the following applies:
  + - * + The variables refIdxLX[ xP ][ yP ] and predFlagLX[ xP ][ yP ] are derived as follows:

refIdxLX[ xP ][ yP ] = refIdxLXRL[ xRL ][ yRL ] (H‑49)

predFlagLX[ xP ][ yP ] = predFlagLXRL[ xRL ][ yRL ] (H‑50)

* + - * + The variable mvLX[ xP ][ yP ][ 0 ] is derived as follows:
        + If ScaledRefLayerPicWidthInSamplesL is not equal to RefLayerPicWidthInSamplesL, mvLX[ xP ][ yP ][ 0 ] is derived as follows:

scaleFactorMVX = Clip3( −4096, 4095, ( ( ScaledRefLayerPicWidthInSamplesL << 8 ) + ( RefLayerPicWidthInSamplesL >> 1 ) ) / RefLayerPicWidthInSamplesL) (H‑51)

mvLX[ xP ][ yP ][0] = Clip3( −32768, 32767, Sign(scaleFactorMVX \*   
mvLXRL[ xRL ][ yRL ][ 0 ] ) \*  ( ( Abs ( scaleFactorMVX \* mvLXRL[ xRL ][ yRL ][ 0 ] )  
 + 127 ) >> 8 ) ) (H‑52)

* + - * + Otherwise, the following applies:

mvLX[ xP ][ yP ][ 0 ] = mvLXRL[ xRL ][ yRL ][ 0 ] (H‑53)

* + - * + The variable mvLX[ xP ][ yP ][ 1 ] is derived as follows:
        + If ScaledRefLayerPicHeightInSamplesL is not equal to RefLayerPicHeightInSamplesL, mvLX[ xP ][ yP ][ 1 ] is derived as follows:

scaleFactorMVY = Clip3( −4096, 4095, ( ( ScaledRefLayerPicHeightInSamplesL << 8 ) + ( RefLayerPicHeightInSamplesL >> 1 ) ) / RefLayerPicHeightInSamplesL) (H‑54)

mvLX[ xP ][ yP ][ 1 ] = Clip3( −32768, 32767, Sign(scaleFactorMVY \*   
mvLXRL[ xRL ][ yRL ][ 1 ] ) \*  ( ( Abs  ( scaleFactorMVY \* mvLXRL[ xRL ][ yRL ][ 1 ] )  
 + 127 ) >> 8 ) ) (H‑55)

* + - * + Otherwise, the following applies:

mvLX[ xP ][ yP ][ 1 ] = mvLXRL[ xRL ][ yRL ][ 1 ] (H‑56)

* Otherwise (predMode[ xP ][ yP ] is equal to MODE\_INTRA), the following applies:
* both components of mvL0[ xP ][ yP ] and mvL1[ xP ][ yP ] are set to 0, refIdxL0[ xP ][ yP ] and refIdxL1[ xP ][ yP ] are set to –1, predFlagL0[ xP ][ yP ] and predFlagL1[ xP ][ yP ] are set to 0.
  + 1. NAL unit decoding process

The specification in subclause 8.2 apply.

* + 1. Slice decoding processes
       1. Decoding process for picture order count

The specifications in subclause F.8.3.1 apply.

* + - 1. Decoding process for reference picture set

The specifications in subclause F.8.3.2 apply.

* + - 1. Decoding process for generating unavailable reference pictures

The specifications in subclause 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 offsetPicLayerId is set equal to 1 when enable\_higher\_layer\_ref\_pic\_pred[ currPicLayerId ] is equal to 1 and TemporalId is greater than 0 for the current picture

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++ ) (H‑57)  
 RefPicListTemp0[ rIdx ] = RefPicSetStCurrAfter[ i ]  
 for( i = 0; i < NumPocLtCurr && rIdx < NumRpsCurrTempList0; rIdx++, i++ )  
 RefPicListTemp0[ rIdx ] = RefPicSetLtCurr[ i ]  
 for( i = 0; i < NumActiveRefLayerPics1; rIdx++, i++ )  
 RefPicListTemp0[ rIdx ] = RefPicSetInterLayer1[ i ]  
}  
while( rIdx < NumRpsCurrTempList0 << offsetPicLayerId) {  
 for( i = 0; i < NumPocStCurrBefore && rIdx < NumRpsCurrTempList0; rIdx++, i++ )  
 RefPicListTemp0[ rIdx ] = RefPicSetStCurrBefore[ i  + NumRpsCurrTempList0]  
 for( i = 0; i < NumActiveRefLayerPics0; rIdx++, i++ )  
 RefPicListTemp0[ rIdx ] = RefPicSetInterLayer0[ i  + NumRpsCurrTempList0]  
 for( i = 0; i < NumPocStCurrAfter && rIdx < NumRpsCurrTempList0; rIdx++, i++ ) (H‑57)  
 RefPicListTemp0[ rIdx ] = RefPicSetStCurrAfter[ i  + NumRpsCurrTempList0]  
 for( i = 0; i < NumPocLtCurr && rIdx < NumRpsCurrTempList0; rIdx++, i++ )  
 RefPicListTemp0[ rIdx ] = RefPicSetLtCurr[ i  + NumRpsCurrTempList0]  
 for( i = 0; i < NumActiveRefLayerPics1; rIdx++, i++ )  
 RefPicListTemp0[ rIdx ] = RefPicSetInterLayer1[ i  + NumRpsCurrTempList0]  
}

The list RefPicList0 is constructed as follows:

for( rIdx = 0; rIdx <= num\_ref\_idx\_l0\_active\_minus1; rIdx++) (H‑58)  
 RefPicList0[ rIdx ] = ref\_pic\_list\_modification\_flag\_l0 ? RefPicListTemp0[ list\_entry\_l0[ rIdx ] ] :  
 RefPicListTemp0[ rIdx ]  
if(offsetPicLayerId  &&  collocated\_from\_l0\_flag)  
 RefPicList0[ rIdx  -  1] = ref\_pic\_list\_modification\_flag\_l0 ? RefPicListTemp0[ list\_entry\_l0[ collocated\_ref\_idx ] + NumRpsCurrTempList0 ] : RefPicListTemp0[ collocated\_ref\_idx  + NumRpsCurrTempList0]

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++ ) (H‑59)  
 RefPicListTemp1[ rIdx ] = RefPicSetStCurrBefore[ i ]  
 for( i = 0; i < NumPocLtCurr && rIdx < NumRpsCurrTempList1; rIdx++, i++ )  
 RefPicListTemp1[ rIdx ] = RefPicSetLtCurr[ i ]  
 for( i = 0; i< NumActiveRefLayerPics0; rIdx++, i++ )  
 RefPicListTemp1[ rIdx ] = RefPicSetInterLayer0[ i ]  
}  
while( rIdx < NumRpsCurrTempList1 << offsetPicLayerId ) {  
 for( i = 0; i < NumPocStCurrAfter && rIdx < NumRpsCurrTempList1; rIdx++, i++ )  
 RefPicListTemp1[ rIdx ] = RefPicSetStCurrAfter[ i  + NumRpsCurrTempList1]  
 for( i = 0; i< NumActiveRefLayerPics1; rIdx++, i++ )  
 RefPicListTemp1[ rIdx ] = RefPicSetInterLayer1[ i  + NumRpsCurrTempList1]  
 for( i = 0; i < NumPocStCurrBefore && rIdx < NumRpsCurrTempList1; rIdx++, i++ ) (H‑59)  
 RefPicListTemp1[ rIdx ] = RefPicSetStCurrBefore[ i + NumRpsCurrTempList1]  
 for( i = 0; i < NumPocLtCurr && rIdx < NumRpsCurrTempList1; rIdx++, i++ )  
 RefPicListTemp1[ rIdx ] = RefPicSetLtCurr[ i + NumRpsCurrTempList1]  
 for( i = 0; i< NumActiveRefLayerPics0; rIdx++, i++ )  
 RefPicListTemp1[ rIdx ] = RefPicSetInterLayer0[ i + NumRpsCurrTempList1]  
}

When the slice is a B slice, the list RefPicList1 is constructed as follows:

for( rIdx = 0; rIdx <= num\_ref\_idx\_l1\_active\_minus1; rIdx++) (H‑60)  
 RefPicList1[ rIdx ] = ref\_pic\_list\_modification\_flag\_l1 ? RefPicListTemp1[ list\_entry\_l1[ rIdx ] ] :  
 RefPicListTemp1[ rIdx ]  
if(offsetPicLayerId  &&  !collocated\_from\_l0\_flag)  
 RefPicList1[ rIdx  -  1] = ref\_pic\_list\_modification\_flag\_l1 ? RefPicListTemp1[ list\_entry\_l1[ collocated\_ref\_idx ] + NumRpsCurrTempList1 ] : RefPicListTemp1[ collocated\_ref\_idx  + NumRpsCurrTempList1]

NOTE – Because motion vectors from inter layer reference pictures are constrained to be zero motion only, an SHVC encoder should disable temporal motion vector prediction for the current picture, by setting slice\_temporal\_mvp\_enabled\_flag to zero, when only inter-layer reference pictures exist in the reference picture lists of all slices in the current picture. This avoids the need to send any additional syntax elements such as collocated\_from\_l0\_flag and collocated\_ref\_idx.

NOTE – When offsetPicLayerId   is not equal to 0, the collocated\_ref\_idx shall be equal to the last index position in its respective list.

* + 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 F.8.5 apply with the following addtions.

It is a requirement of bitstream conformance that, for X being replaced by either 0 or 1, the variables mvLX[0] and mvLX[1] as an output of the subclause 8.5.3.1 shall be equal to 0 if the value of refIdxLX as an output of the subclause 8.5.3.1 corresponds to an 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 currPicLayerId is set to be the nuh\_layer\_id of the current picture.

It is a requirement of bitstream conformance that when enable\_higher\_layer\_ref\_pic\_pred[ currPicLayerId ] not equal to 0 and TemporalId is not equal to 0 for the current picture, for X being replaced by either 0 or 1, the value of refIdxLX as an output of the subclause 8.5.3.1 shall correspond to a picture with nuh\_layer\_id equal to currPicLayerId + 1.

* + 1. Scaling, transformation and array construction process prior to deblocking filter process

The specifications in subclause 8.6 apply.

* + 1. In-loop filter process

The specifications in subclause 8.7 apply.

* 1. Parsing process

The specifications in clause 9 apply.

* 1. Specification of bitstream subsets

The specifications in clause 10 apply.

* 1. Profiles, tiers, and levels
     1. Profiles
        1. General

TBD.

* + - 1. Scalable Main profile

Bitstreams conforming to the scalable main profile shall obey the following constraints:

– The picture resampling process of picture sample values as specified in subclause H.8.1.4.1 shall not be invoked more than once for decoding of each particular picture and the resampling process of picture motion field as specified in subclause H.8.1.4.2 shall not be invoked more than once for decoding of each particular picture. When both picture sample values and picture motion field resampling processes are invoked for decoding of a particular picture, they shall be applied to the same reference layer picture.

– When DependencyId[ i ] is equal to 1, ScalabilityId[ LayerIdxInVps [ i ]][ smIdx ] shall be equal to 0 for any smIdx value from 0 to 15, inclusive, that is not equal to 2, for any coded picture with nuh\_layer\_id equal to i.

– When avc\_base\_layer\_flag equal to 1, it is a requirement of bitstream conformance that MotionPredEnabledFlag[ iNuhLId ][ 0 ] shall be equal to 0 for iNuhLId equal to any value of nuh\_layer\_id present in the bitstream.

* + 1. Tiers and levels

TBD

* 1. Byte stream format

The specifications in subclause F.12 apply.

* 1. Hypothetical reference decoder

The specifications in subclause F.13 and its subclauses apply.

* 1. SEI messages

The specifications in Annex D and subclause F.14 and its subclauses apply.

* 1. Video usability information

The specifications in Annex F.15 apply.