Joint Collaborative Team on 3D Video Coding Extension Development of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11
3rd Meeting: Geneva, CH , 17–23 Jan. 2013

Document: JCT3V-C1004_d1

Title: **MV-HEVC Draft Text 3 (ISO/IEC 23008-2 PDAM2)**

Status: Output Document of JCT-3V

Purpose: Working Draft of MV-HEVC

Author(s) or Contact(s):

Gerhard Tech
Fraunhofer HHI
Email: gerhard.tech@hhi.fraunhofer.de

Krzysztof Wegner
Poznan University of Technology
Email: kwegner@multimedia.edu.pl

Ying Chen
Qualcomm Incorporated
Email: cheny@qti.qualcomm.com

Miska Hannuksela
Nokia Corporation
Email: miska.hannuksela@nokia.com

Jill Boyce
Vidyo
Email: jill@vidyo.com

Source: Editors

ABSTRACT

The MV-HEVC Draft Text 3 is based on JCTVC-L0452-spec-text-r1.doc from

JCT3V-C0238 Common specification text for scalable and multi-view extensions (revision of JCTVC-L0188 straw-man text) [M. M. Hannuksela, K. Ugur, J. Lainema, D. Rusanovskyy (Nokia), J. Chen, V. Seregin, Y.-K. Wang, Y. Chen, L. Guo, M. Karczewicz (Qualcomm), Y. Ye (InterDigital), J. Boyce (Vidyo)].

and has been updated to JCTVC-L1003_v19.doc.

Modifications in long sections copied from the HEVC spec are highlighted in turquoise. Open issues and editor's notes are highlighted in yellow.

Ed. Notes (Draft 3) (changes to JCT3V-B1004)

- (Review GT2)
- (Review MH2)
- (JCT3V-C0238 Marking Process) Replace targetDecLayerIdList by TargetDecLayerIdList in F.8.1.2.1.
- (Ed. Notes 01) Incorporated and removed editor's notes as discussed.
- (JCT3V-0059) Update to new terminology and simplification of text.
- (Fix References) Fix of references and numbering.
- (JCTVC-L0363) Fixed byte alignment corrupted when re-introduction of profilePresentFlag in profile_tier_level for JCTVC-L0180.
- (JCTVC-L0180) Updated of semantics and modified profile_tier_level syntax structure.
- (Review GT01)
- (JCT3V-C0078) Incorporated disparity vector constraints.
- (Update to latest HEVC spec), Updated to JCTVC-L1003_v19.doc.
- (Review Miska01)
- (MVC-CY01) Review, typo corrections, editorial improvement and alignment with B1004.
- (JCT3V-C0085) Integration of JCT3V-C0085: slice type constraint.
- (JCT3V-C0238) Incorporated common specification text for scalable multi-view extensions.

Ed. Notes (WD2) (based on JCT3V-A1004)

- (MVC-MH) Review, typo corrections, editorial improvement, and editor’s notes
- (MVE-06) Incorporated introductory paragraph for view dependency change SEI message.
- (MVE-05) Incorporated invocation of sub-bitstream extraction process in general decoding process
- (MVE-04) Fixed construction of layerId list in general decoding process
- (MVC-KW) Review, typo corrections, editorial improvement.

- (MVE-03) Replacement of changes marks related to base spec by highlighting
- (MVE-01/JCT3V-B00046) Incorporated editorial note
- (MVC-GT) Review, typo corrections, editorial improvement.
- (MVC-CY) Review, typo corrections, editorial improvement.
- (MVE-02) Incorporated initial version of HRD text.
- (MVN-01/JCT3V-B0063) Incorporated view dependency change SEI.

Ed. Notes (WD1) (based on : JCT3V-A0012)

- (Rev3, KW) Review and small corrections
- (Rev2, GT) Review and text improvement
- Missing part in general decoding process
- (Replacement view_id by layer_id)
- Font issue fix.
- (Fix: picture marking)
- (Rev1, CY), Review and small corrections
- (MV07) Fix references
- (MV06) Improvement and update of interview prediction text.
- (MV02,MV03) Update of high level syntax and definitions
- (MV09) general HEVC decoding process
- (MV08) Additional sections/placeholders
- (MV04, MV05) Removal of low-level and depth tools
- (MV01) Removed HEVC spec
- Update of low level specification to match HEVC text specification 8(d7)

CONTENTS

Page

Replace subclause 8.1, 8.2 and 8.3 with the following (with additions indicated in turquoise).

8 Decoding process

8.1 General decoding process

The input of this process is a bitstream and the output 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 subclause 10.1 is applied with the bitstream, HighestTid and TargetDecLayerIdList as inputs and the output is assigned to a bitstream referred to as BitstreamToDecode.

The following applies 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 S_L.

– Otherwise (chroma_format_idc is not equal to 0), the current picture consists of 3 sample arrays S_L, S_{Cb}, S_{Cr}.

The decoding process for the current picture takes as input 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).

The decoding process is specified such that all decoders will produce numerically identical cropped output pictures. Any decoding process that produces identical cropped output 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 has nuh_layer_id equal to 0, the decoding process for a coded picture with nuh_layer_id equal to 0 specified in subclause 8.1.1 is invoked.

8.1.1 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.

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

– If the current picture with a particular nuh_layer_id is the first picture with that particular nuh_layer_id in the bitstream in decoding order, the first picture with that particular nuh_layer_id that follows an end of sequence NAL unit in decoding order, an IDR picture, or a BLA picture, 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.

Each picture referred to in this clause is a complete coded picture.

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 the 3 sample arrays of the current picture with the NAL units with colour_plane_id equal to 0 being assigned to S_L, the NAL units with colour_plane_id equal to 1 being assigned to S_{Cb}, and the NAL units with colour_plane_id equal to 2 being assigned to S_{Cr}.

NOTE – The variable ChromaArrayType is derived as 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 with chroma_format_idc being 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 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 (which needs to be invoked only for the first slice segment of a picture).
   – The decoding process for reference picture set in subclause 8.3.2 is invoked, wherein reference pictures may be marked as "unused for reference" or "used for long-term reference" (which 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).
   – After all slices of the current picture have been decoded, the decoded picture is marked as "used for short-term reference".
3. The processes in subclauses 8.4, 8.5, 8.6, and 8.7 specify decoding processes using syntax elements in the coding tree unit layer and above.

8.2 NAL unit decoding process

Inputs to this process are NAL units of the access unit containing the current picture.

Outputs of this process are the 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 operates the decoding processes specified for the RBSP syntax structure in the NAL unit as follows.

Subclause 8.3 describes the decoding process for VCL NAL units.

NAL units with nal_unit_type equal to VPS_NUT, SPS_NUT, and PPS_NUT contain VPSs, SPSs, and PPSs, respectively. SPSs are used in the decoding processes of other NAL units as determined by reference to an SPS within the PPSs or active parameter sets SEI messages. PPSs are used in the decoding processes of other NAL units as determined by reference to a PPS within the slice segment headers.

8.3 Slice decoding process

8.3.1 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.

– The variable prevPicOrderCntLsb is set equal to slice_pic_order_cnt_lsb of prevTid0Pic.

– The variable prevPicOrderCntMsb is set equal to PicOrderCntMsb of prevTid0Pic.

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

– If the current picture is an IRAP picture with NoRaslOutputFlag equal to 1, PicOrderCntMsb is set equal to 0.

– Otherwise, PicOrderCntMsb is derived as follows:

```
if( ( slice_pic_order_cnt_lsb <  prevPicOrderCntLsb )  &&
        ( ( prevPicOrderCntLsb − slice_pic_order_cnt_lsb )  >=  ( MaxPicOrderCntLsb / 2 ) ) )
    PicOrderCntMsb = prevPicOrderCntMsb + MaxPicOrderCntLsb                    (8-1)
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

$$PicOrderCntVal = PicOrderCntMsb + slice\_pic\_order\_cnt\_lsb \quad (8\text{-}2)$$

NOTE 1 – All IDR pictures will have PicOrderCntVal equal to 0 since slice_pic_order_cnt_lsb is inferred to be 0 for IDR pictures and prevPicOrderCntLsb and prevPicOrderCntMsb are both set equal to 0.

The value of PicOrderCntVal shall be in the range of -2^{31} to $2^{31} - 1$, inclusive. In one CVS, the PicOrderCntVal values for any two coded pictures shall not be the same.

The function PicOrderCnt(picX) is specified as follows:

PicOrderCnt(picX) = PicOrderCntVal of the picture picX (8-3)

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

DiffPicOrderCnt(picA, picB) = PicOrderCnt(picA) − PicOrderCnt(picB) (8-4)

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 -2^{15} to $2^{15} - 1$, inclusive.

NOTE 2 – 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.

Annex F

Syntax, semantics and decoding processes for multiview coding

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

This annex specifies syntax, semantics and decoding processes for multiview coding.

F.1 Scope

Decoding processes conforming to this annex are completely specified in this annex with reference made to clauses 2-9 and Annexes A-E and Annex G.

F.2 Normative references

The specifications in clause 2 apply.

F.3 Definitions

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

[Ed. (YK&MH): Definitions should be checked and potentially refined, including: BLA AU, coded video sequence, output order, picture order count, RADL AU, RASL AU, IRAP AU, IRAP picture, (reference picture), reference picture set, STSA AU, TSA AU]

[Ed. (MH): The present version of this annex uses the same definition for an access unit as in clause 3, which essentially states that an access unit contains one coded picture (with a particular value of nuh_layer_id). One coded picture is defined below essentially identically to a view component in MVC. It is an open issue whether an access unit should instead be defined to contain all view components with the same POC value.]

F.3.1 associated IRAP picture: The previous *IRAP picture* in *decoding order* within the same *layer* (if present).

F.3.2 base layer: A *layer* in which all *VCL NAL units* have nuh_layer_id equal to 0.

F.3.3 coded picture: A *coded representation* of a *picture* comprising *VCL NAL units* with a particular value of nuh_layer_id and containing all *coding tree units* of the *picture*.

F.3.4 inter-layer prediction: A *prediction* in manner that is dependent on data elements (e.g. sample values or motion vectors) of *reference pictures* with another value of nuh_layer_id than that for the current *picture*.

F.3.5 leading picture: A *picture* that is in the same *layer* as the *associated IRAP picture* and precedes the *associated IRAP picture* in *output order*.

F.3.6 non-base layer: A *layer* in which all *VCL NAL units* have the same nuh_layer_id value greater than 0.

F.3.7 target output layer: A *layer* that is to be output.

F.3.8 trailing picture: A *picture* that is in the same *layer* as the *associated IRAP picture* and follows the *associated IRAP picture* in *output order*.

F.3.9 view: a sequence of pictures with an identical value of ViewId.

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

F.4 Abbreviations

The specification in clause 4 apply.

F.5 Conventions

The specification in clause 5 apply.

F.6 Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships

The specification in clause 6 apply.

F.7 Syntax and semantics

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

F.7.1 Method of specifying syntax in tabular form

The specifications in subclause 7.1 apply.

F.7.2 Specification of syntax functions, categories, and descriptors

The specifications in subclause 7.2 apply.

F.7.3 Syntax in tabular form

F.7.3.1 NAL unit syntax

The specifications in subclause 7.3.1 apply.

F.7.3.1.1 General NAL unit syntax

The specifications in subclause 7.3.1.1 apply.

F.7.3.1.2 NAL unit header syntax

The specifications in subclause 7.3.1.2 apply.

F.7.3.2 Raw byte sequence payloads and RBSP trailing bits syntax

F.7.3.2.1 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) {	
vps_extension( )	
vps_extension2_flag	u(1)
if(vps_extension2_flag)	
while(more_rbsp_data( ))	
vps_extension_data_flag	u(1)
}	
rbsp_trailing_bits( )	
}	

F.7.3.2.1.1 Video parameter set extension syntax

vps_extension( ) {	Descriptor
while(!byte_aligned( ))	
vps_extension_byte_alignment_reserved_one_bit	u(1)
avc_base_layer_flag	u(1)
splitting_flag	u(1)
for(i = 0, NumScalabilityTypes = 0; i < 16; i++) {	
scalability_mask[i]	u(1)
NumScalabilityTypes += scalability_mask[i]	
}	
for(j = 0; j <NumScalabilityTypes; 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)
for(j = 0; j < NumScalabilityTypes; j++)	
dimension_id[i][j]	u(v)
}	
for(lsIdx = 1; lsIdx <= vps_num_layer_sets_minus1; lsIdx ++) {	
vps_profile_present_flag[lsIdx]	u(1)
if(!vps_profile_present_flag[lsIdx])	
profile_layer_set_ref_minus1[lsIdx]	ue(v)
profile_tier_level(vps_profile_present_flag[lsIdx], vps_max_sub_layers_minus1)	
}	
num_output_layer_sets	ue(v)
for(i = 0; i < num_output_layer_sets; i++) {	
output_layer_set_idx[i]	ue(v)
lsIdx = output_layer_set_idx[i]	
for(j = 0 ; j <= vps_max_layer_id; j++)	
if(layer_id_included_flag[lsIdx][j])	
output_layer_flag[lsIdx][j]	u(1)
}	
for(i = 1; i <= vps_max_layers_minus1; i++)	
for(j = 0; j < i; j++)	
direct_dependency_flag[i][j]	u(1)
}	

[Ed. (YK&MH): consider if profile_layer_set_ref_minus1[i] and output_layer_set_idx[i] should be fixed-length coded as u(10).]

F.7.3.2.2 Sequence parameter set RBSP syntax

The specifications in subclause 7.3.2.2 apply, with the following modifications.

– "profile_tier_level(sps_max_sub_layers_minus1)" is replaced by
"profile_tier_level(1, sps_max_sub_layers_minus1)"

F.7.3.2.3 Picture parameter set RBSP syntax

The specifications in subclause 7.3.2.3 apply.

F.7.3.2.4 Supplemental enhancement information RBSP syntax

The specifications in subclause 7.3.2.4 apply.

F.7.3.2.5 Access unit delimiter RBSP syntax

The specifications in subclause 7.3.2.5 apply.

F.7.3.2.6 End of sequence RBSP syntax

The specifications in subclause 7.3.2.6 apply.

F.7.3.2.7 End of bitstream RBSP syntax

The specifications in subclause 7.3.2.7 apply.

F.7.3.2.8 Filler data RBSP syntax

The specifications in subclause 7.3.2.8 apply.

F.7.3.2.9 Slice segment layer RBSP syntax

The specifications in subclause 7.3.2.9 apply.

F.7.3.2.10 RBSP slice segment trailing bits syntax

The specifications in subclause 7.3.2.10 apply.

F.7.3.2.11 RBSP trailing bits syntax

The specifications in subclause 7.3.2.11 apply.

F.7.3.2.12 Byte alignment syntax

The specifications in subclause 7.3.2.12 apply.

F.7.3.3 Profile, tier and level syntax

[Ed. (GT): In base spec the profilePresentFlag has been removed although used by required by JCTVC-180. Therefore it is re-incorporated here. See editor's note in semantics.]

profile_tier_level(profilePresentFlag, maxNumSubLayersMinus1) {	**Descriptor**
if(profilePresentFlag) {	
general_profile_space	u(2)
general_tier_flag	u(1)
general_profile_idc	u(5)
for(i = 0; i < 32; i++)	
general_profile_compatibility_flag[i]	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)
}	
}	

F.7.3.4 Scaling list data syntax

The specifications in subclause 7.3.4 apply

F.7.3.5 Supplemental enhancement information message syntax

The specifications in subclause 7.3.5 apply

F.7.3.6 Slice segment header syntax

F.7.3.6.1 General slice segment header syntax

The specifications in subclause 7.3.6.1 apply

F.7.3.6.2 Short-term reference picture set syntax

The specifications in subclause 7.3.6.2 apply

F.7.3.6.3 Reference picture list modification syntax

The specifications in subclause 7.3.6.3 apply

F.7.3.6.4 Weighted prediction parameters syntax

The specifications in subclause 7.3.6.4 apply

F.7.3.7 Slice segment data syntax

F.7.3.7.1 General slice segment data syntax

The specifications in subclause 7.3.7.1 apply.

F.7.3.7.2 Coding tree unit syntax

The specifications in subclause 7.3.7.2 apply.

F.7.3.7.3 Sample adaptive offset syntax

The specifications in subclause 7.3.7.3 apply.

F.7.3.7.4 Coding quadtree syntax

The specifications in subclause 7.3.7.4 apply.

F.7.3.7.5 Coding unit syntax

The specifications in subclause 7.3.7.5 apply.

F.7.3.7.6 Prediction unit syntax

The specifications in subclause 7.3.7.6 apply.

F.7.3.7.7 PCM sample syntax

The specifications in subclause 7.3.7.7 apply.

F.7.3.7.8 Transform tree syntax

The specifications in subclause 7.3.7.8 apply.

F.7.3.7.9 Motion vector difference syntax

The specifications in subclause 7.3.7.9 apply.

F.7.3.7.10 Transform unit syntax

The specifications in subclause 7.3.7.10 apply.

F.7.3.7.11 Residual coding syntax

The specifications in subclause 7.3.7.11 apply.

F.7.4 Semantics

F.7.4.1 NAL unit semantics

F.7.4.1.1 General NAL unit semantics

The specifications in subclause 7.4.1.1 apply.

F.7.4.1.2 NAL unit header semantics

The specifications in subclause 7.4.1.2 apply with following modifications and additions.

nuh_layer_id specifies the identifier of the layer.

When the nal_unit_type value nalUnitTypeA is equal to IDR_W_DLP, IDR_N_LP, BLA_W_LP, BLA_W_DLP or

BLA_N_LP for a coded picture with a particular PicOrderCntVal value and within a particular CVS, the nal_unit_type value shall be equal to nalUnitTypeA for all VCL NAL units of all coded pictures with the same particular PicOrderCntVal value and within the same particular CVS.

[Ed. (GT): Shouldn't this also include CRA pictures, when they are first in the CVS or HandleCraAsBlaFlag is equal to 1? Hence, restrictions for HandleCraAsBlaFlag might be require as well as:

When NoRaslOutputFlag is equal to 1 for a coded picture with a particular nalUnitTypeA and a particular PicOrderCntVal value and within a particular CVS, the nal_unit_type value shall be equal to nalUnitTypeA for all VCL NAL units of all coded pictures with the same particular PicOrderCntVal value and within the same particular CVS.

]

F.7.4.1.3 Encapsulation of an SODB within an RBSP (informative)

The specifications in subclause 7.4.1.3 apply.

F.7.4.1.4 Order of NAL units and association to coded pictures, access units, and coded video sequences

The specifications in subclause 7.4.1.4 apply with the following additions.

A coded picture with nuh_layer_id equal to nuhLayerIdA and with a PicOrderCntVal value equal to picOrderCntValA shall precede in decoding order all coded pictures with nuh_layer_id greater than nuhLayerIdA and with a PicOrderCntVal value equal to picOrderCntValA, if present.

F.7.4.1.4.1 Order of VPS, SPS and PPS RBSPs and their activation

The specifications in subclause 7.4.1.4.1 apply with the following additions.

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, unless the PPS is 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, unless the SPS is 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 an IDR or BLA picture having that nuh_layer_id value, inclusive, until either the next IDR or BLA 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 shall be considered in effect.

F.7.4.1.4.2 Order of access units and association to CVS

The specifications in subclause 7.4.1.4.2 apply.

F.7.4.1.4.3 Order of NAL units and coded pictures and association to access units

The specifications in subclause 7.4.1.4.3 apply.

F.7.4.1.4.4 Order of VCL NAL units and association to coded pictures

The specifications in subclause 7.4.1.4.4 apply.

F.7.4.2 Raw byte sequence payloads, trailing bits, and byte alignment semantics

F.7.4.2.1 Video parameter set RBSP semantics

The specifications in subclause 7.4.2.1 apply with following modifications and additions.

– layerSetLayerIdList is replaced by LayerSetLayerIdList

– numLayersInIdList is replaced by NumLayersInIdList

vps_extension_offset specifies the byte offset of the next set of fixed-length coded information in the VPS NAL unit, starting from the beginning of the NAL unit.

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_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 1 in bitstreams conforming to Annex F of this Recommendation | International Standard. 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.

F.7.4.2.1.1 Video parameter set extension semantics

vps_extension_byte_alignment_reserved_one_bit shall be equal to 1.

avc_base_layer_flag equal to 1 specifies that the base layer conforms to Rec. ITU-T H.264 | ISO/IEC 14496-10, equal to 0 specifies that it conforms to this specification.

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

When avc_base_layer_flag 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. ITU-T 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.

splitting_flag equal to 1 indicates that the bits of the nuh_layer_id syntax element in the NAL unit header are split into n segments with a length, in bits, according to the values of the dimension_id_len_minus1[i] syntax element and that the n segments are associated with the n scalability dimensions indicated in scalability_mask_flag[i]. When splitting_flag is equal to 1, the value of the j-th segment of the nuh_layer_id of i-th layer shall be equal to the value of dimension_id[i][j]. splitting_flag equal to 0 does not indicate the above constraint.

NOTE 1 – When splitting_flag is equal to 1, i.e. the restriction reported in the semantics of the dimension_id[i][j] syntax element is obeyed, scalable identifiers can be derived from the nuh_layer_id syntax element in the NAL unit header by a bit masked copy as an alternative to the derivation as reported in the semantics of the dimension_id[i][j] syntax element. 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[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[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	multiview	ViewId
1-15	Reserved	

dimension_id_len_minus1[j] plus 1 specifies the length, in bits, of the dimension_id[i][j] syntax element.

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.

$$\mathrm{dimBitOffset}[\,j\,] = \sum_{\mathrm{dimIdx}=0}^{j-1} (\mathrm{dimension\_id\_len\_minus1}[\,\mathrm{dimIdx}\,] + 1) \tag{F-1}$$

vps_nuh_layer_id_present_flag specifies whether the layer_id_in_nuh[i] syntax is 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 a range from 0 to vps_max_layers_minus1, inclusive, when not present, the value of layer_id_in_nuh[i] 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].

[Ed. (MH): When splitting_flag is equal to 1, the MSBs of layer_id_in_nuh should be required to be 0 if the total number of bits in segments is less than 6]

For i in a range from 0 to vps_max_layers_minus1, inclusive, the variable LayerIdInVps[layer_id_in_nuh[i]] is set equal to i.

dimension_id[i][j] specifies the identifier of the j-th scalability dimension type of the i-th layer. When not present, the value of dimension_id[i][j] is inferred to be equal to 0. The number of bits used for the representation of dimension_id[i][j] is dimension_id_len_minus1[j] + 1 bits. When splitting_flag is equal to 1, it is a requirement of bitstream conformance that dimension_id[i][j] shall be equal to (layer_id_in_nuh[i] & ((1 << (dimension_id_len_minus1[j] + 1) − 1) >> dimBitOffset[j])).

The variables ScalabilityId[layerIdInVps][scalabilityMaskIndex] and ViewId[layerIdInNuh] are derived as follows:

```
for (i = 0; i <= vps_max_layers_minus1; i++) {
    for( smIdx= 0, j =0; smIdx< 16; smIdx ++ )
        if( ( i  !=  0 )  &&  scalability_mask[ smIdx ] )
            ScalabilityId[ i ][ smIdx ] = dimension_id[ i ][ j++ ]
        else
            ScalabilityId[ i ][ smIdx ] = 0
    ViewId[ layer_id_in_nuh[ i ] ] = ScalabilityId[ i ][ 0 ]
}
```

[Ed. (MH/CY): ViewId[0], i.e. the ViewId value for the base view, is constrained to be equal to 0 according to the semantics above, whereas in the previous versions of MV-HEVC specification (e.g. JCT3V-B1004) it was possible to assign a non-zero ViewId value for the base view and ViewId was defined in a way similar to MVC. Consequently, it was possible to assign a non-base view with a smaller ViewId value than the ViewId value of the base view, which is no longer possible in the present version of the annex. In the current 3D-HEVC working draft, the ViewId values are used for motion vector scaling and essentially indicate relative camera positions. The constraint that the ViewId value for the base view is equal to 0 has a consequence that it is essentially considered to

be an outmost view in the camera setup. It is an open issue whether to define ViewId in the same way as in previous versions of MV-HEVC and similar to MVC.]

vps_profile_present_flag[lsIdx] equal to 1 specifies that the profile and tier information for layer set lsIdx is present in the profile_tier_level( ) syntax structure. vps_profile_present_flag[lsIdx] equal to 0 specifies that profile and tier information for layer set lsIdx is not present in the profile_tier_level( ) syntax structure and is inferred.

profile_layer_set_ref_minus1[lsIdx] indicates that the profile and tier information for the lsIdx-th layer set is inferred to be equal to the profile and tier information from the (profile_layer_set_ref_minus1[lsIdx] + 1)-th layer set. The value of profile_layer_set_ref_minus1[lsIdx] + 1 shall be less than lsIdx.

num_output_layer_sets specifies the number of layer sets for which output layers are specified with output_layer_set_index[i] and output_layer_flag[lsIdx][j]. When not present, the value of num_output_layer_sets is inferred to be equal to 0.

output_layer_set_idx[i] specifies the index lsIdx of the layer set for which output_layer_flag[lsIdx][j] is present.

[Ed. (GT): Following constraint should be added here: "The value of output_layer_set_idx[i] shall be in the range of 0 to vps_num_layer_sets_minus1, inclusive."]

output_layer_flag[lsIdx][j] equal to 1 specifies that the layer with nuh_layer_id equal to j is a target output layer of the lsIdx-th layer set. A value of output_layer_flag[lsIdx][j] equal to 0 specifies that the layer with nuh_layer_id equal to j is not a target output layer of the lsIdx-th layer set.

When output_layer_flag[lsIdx][j] is not present for lsIdx in the range of 0 to vps_num_layer_sets_minus1, inclusive and for j in the range of 0 to 63, inclusive, output_layer_flag[lsIdx][j] is inferred to be equal to (j = = LayerSetLayerIdList[lsIdx][NumLayersInIdList[lsIdx] – 1]).

NOTE 1 – In other words, when a layer set is not included among those indicated by output_layer_set_idx[i], the layer with the greatest value of nuh_layer_id within the layer set is the only target output layer of the layer set.

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] are derived as follows:

```
for( i = 1; i <= vps_max_layers_minus1; i++ )
    for( j = 0, NumDirectRefLayers[ i ] = 0; j < i; j++ )
        if( direct_dependency_flag[ i ][ j ]  = =  1 )
            RefLayerId[ i ][ NumDirectRefLayers[ i ]++ ] = layer_id_in_nuh[ j ]
```

F.7.4.2.2 Sequence parameter set RBSP semantics

The specifications in subclause 7.4.2.2 apply.

F.7.4.2.3 Picture parameter set RBSP semantics

The specifications in subclause 7.4.2.3 apply.

F.7.4.2.4 Supplemental enhancement information RBSP semantics

The specifications in subclause 7.4.2.4 apply.

F.7.4.2.5 Access unit delimiter RBSP semantics

The specifications in subclause 7.4.2.5 apply.

F.7.4.2.6 End of sequence RBSP semantics

The specifications in subclause 7.4.2.6 apply.

F.7.4.2.7 End of bitstream RBSP semantics

The specifications in subclause 7.4.2.7 apply.

F.7.4.2.8 Filler data RBSP semantics

The specifications in subclause 7.4.2.8 apply.

F.7.4.2.9 Slice segment layer RBSP semantics

The specifications in subclause 7.4.2.9 apply.

F.7.4.2.10 RBSP slice segment trailing bits semantics

The specifications in subclause 7.4.2.10 apply.

F.7.4.2.11 RBSP trailing bits semantics

The specifications in subclause 7.4.2.11 apply.

F.7.4.2.12 Byte alignment semantics

The specifications in subclause 7.4.2.12 apply.

F.7.4.3 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.

[Ed. (GT): The case when profile_tier_level( ) is included in an SPS need to be specified. E.g.as for HRD parameters:

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 for which the associated layer identifier list contains all nuh_layer_id values present in the CVS.]

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[i], 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 7.4.3 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.

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. Bitstreams shall not contain values of general_profile_idc other than those specified in Annex A or subclause G.11. Other values of general_profile_idc are reserved for future use by ITU-T | ISO/IEC.

general_profile_compatibility_flag[i] 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. 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[i] shall be equal to 0 for any value of i that is not specified as an allowed value of general_profile_idc in Annex A or in subclause G.11.

general_level_idc indicates a level to which the CVS conforms as specified in Annex A or subclause G.11.

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.

[Ed. (GT): The last constraint has been added to enable the reintroduction of profilePresentFlag in a similar way as in HEVC Draft 9 needed for JCTVC-L0180, while keeping byte alignment as specified in JCTVC-L0363 intact. In HEVC draft 9 sub_layer_profile_present_flag and sub layer profile data are only present when profilePresentFlag is equal to 1. To be consistent with at least the presence of sub layer profile data and to not comprise the byte alignment adopted with JCTVC-L0363, here sub_layer_profile_present_flag is also present when profilePresentFlag is equal to 0, but constraint to be equal to 0. This requires further discussion.]

F.7.4.4 Scaling list data semantics

The specifications in subclause 7.4.4 apply

F.7.4.5 Supplemental enhancement information message semantics

The specifications in subclause 7.4.5 apply

F.7.4.6 Slice segment header semantics

F.7.4.6.1 General slice segment header semantics

The specifications in subclause 7.4.6.1 apply.

F.7.4.6.2 Short-term reference picture set semantics

The specifications in subclause 7.4.6.2 apply

F.7.4.6.3 Reference picture list modification semantics

The specifications in subclause 7.4.6.3 apply.

F.7.4.6.4 Weighted prediction parameters semantics

The specifications in subclause 7.4.6.4 apply

F.7.4.7 Slice segment data semantics

F.7.4.7.1 General slice segment data semantics

The specifications in subclause 7.4.7.1 apply.

F.7.4.7.2 Coding tree unit semantics

The specifications in subclause 7.4.7.2 apply.

F.7.4.7.3 Sample adaptive offset semantics

The specifications in subclause 7.4.7.3 apply.

F.7.4.7.4 Coding quadtree semantics

The specifications in subclause 7.4.7.4 apply.

F.7.4.7.5 Coding unit semantics

The specifications in subclause 7.4.7.5 apply.

F.7.4.7.6 Prediction unit semantics

The specifications in subclause 7.4.7.6 apply.

F.7.4.7.7 PCM sample semantics

The specifications in subclause 7.4.7.7 apply.

F.7.4.7.8 Transform tree semantics

The specifications in subclause 7.4.7.8 apply.

F.7.4.7.9 Motion vector difference semantics

The specifications in subclause 7.4.7.9 apply.

F.7.4.7.10 Transform unit semantics

The specifications in subclause 7.4.7.10 apply.

F.7.4.7.11 Residual coding semantics

The specifications in subclause 7.4.7.11 apply.

F.8 Decoding process

F.8.1 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 the 3 sample arrays of the current picture with the NAL units with colour_plane_id equal to 0 being assigned to S_L, the NAL units with colour_plane_id equal to 1 being assigned to S_{Cb}, and the NAL units with colour_plane_id equal to 2 being assigned to S_{Cr}.

    NOTE – The variable ChromaArrayType is derived as 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 with chroma_format_idc being equal to 0.

– The following ordered steps apply for the decoding of the current picture.
  – 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 ViewId[nuh_layer_id] is greater than 0, the decoding process for a coded picture with nuh_layer_id greater than 0 specified in subclause G.8.1 is invoked. [Ed. (MH): Note that ViewId equal to 0 indicates the base view, as specified by the semantics of the dimension_id syntax element.] [Ed. (MH): In future scalable extensions, other types of scalability identifiers may be examined here, and appropriate decoding processes invoked for them.]
  – 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.

F.8.1.1 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 8.2.
2. The following ordered steps specify the 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 (which 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 reference picture set in subclause 8.3.2 is invoked for pictures with nuh_layer_id equal to that of CurrPic, wherein reference pictures may be marked as "unused for reference" or "used for long-term reference" (which needs to be invoked only for the first slice segment of a picture).

[Ed. (GT): The above expression "8.3.2 is invoked for pictures with nuh_layer_id equal to that of CurrPic" needs to be clarified further.].

– When CurrPic is a BLA picture or 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).

F.8.1.2 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, PicOutputFlag is set equal to pic_output_flag.

The following applies:

– 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.

F.8.1.2.1 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.

The following applies for marking of pictures as "unused for reference":

```
for( i = 0; i <= latestDecIdx; i++ ) {
    if( the picture with picture order count equal to PicOrderCntVal and
        with nuh_layer_id equal to TargetDecLayerIdList[ i ] is not marked as
        "unused for reference" and is a sub-layer non-reference picture ) {
        remainingInterLayerReferencesFlag = 0
        for( j = latestDecIdx + 1; j < numTargetDecLayers; j++ )
            for( k = 0; k < NumDirectRefLayers[ LayerIdInVps[ TargetDecLayerIdList[ j ] ] ]; k++ )
                if( TargetDecLayerIdList[ i ]  = =  RefLayerId[ LayerIdInVps[ TargetDecLayerIdList[ j ] ][ k ] )
                    remainingInterLayerReferencesFlag = 1
        if( !remainingInterLayerReferencesFlag )
            mark the picture with picture order count equal to PicOrderCntVal and
                with nuh_layer_id equal to TargetDecLayerIdList[ i ] as "unused for reference"
    }
}
```

F.9 Parsing process

The specifications in clause 9 apply.

F.10 Specification of bitstream subsets

The specifications in clause 10 apply.

F.11 (Void)

(void) [Ed. (MH): no profiles are intended to be specified as part of this annex, as it only specifies common syntax and semantics and some decoding processes for multiview extension and potential future scalable extensions.]

F.12 Byte stream format

The specifications in Annex B apply.

F.13 Hypothetical reference decoder

F.13.1 General

The specifications in subclause C.1 apply.

F.13.2 Operation of coded picture buffer (CPB)

The specifications in subclause C.2 apply.

F.13.3 Operation of the decoded picture buffer (DPB)

The specifications in subclause C.3 apply separately for each set of decoded pictures with a particular value of nuh_layer_id.

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.

F.13.4 Bitstream conformance

The specifications in subclause C.4 apply.

F.13.5 Decoder conformance

F.13.5.1 General

The specifications in subclause C.5.1 apply.

F.13.5.2 Operation of the output order DPB

F.13.5.2.1 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. At HRD initialization, the DPB is empty. 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.

F.13.5.2.2 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 access unit containing the current picture is removed from the CPB and proceeds as follows.

The decoding process for reference picture set as specified in subclause 8.3.2 is invoked.

– If the current picture is an IRAP picture with NoRaslOutputFlag equal to 1 and with nuh_layer_id equal to 0, the following applies.

  1. When the IDR or BLA picture with nuh_layer_id equal to 0 is not the first picture decoded and the value of pic_width_in_luma_samples or pic_height_in_luma_samples or sps_max_dec_pic_buffering_minus1[HighestTid] for any possible value of i derived from the active SPS is different from the value of pic_width_in_luma_samples or pic_height_in_luma_samples or sps_max_dec_pic_buffering_minus1[HighestTid] derived from the SPS that was active for the preceding

picture, respectively, no_output_of_prior_pics_flag is inferred to be equal to 1 by the HRD, regardless of the actual value of no_output_of_prior_pics_flag.

NOTE – Decoder implementations should try to handle picture or DPB size changes more gracefully than the HRD in regard to changes in pic_width_in_luma_samples, pic_height_in_luma_samples or sps_max_dec_pic_buffering_minus1 [HighestTid].

2. When the IRAP picture is a CRA picture with NoRaslOutputFlag equal to 1, no_output_of_prior_pics_flag is inferred to be equal to 1, regardless of the actual value of no_output_of_prior_pics_flag.
3. When no_output_of_prior_pics_flag is equal to 1 or is inferred to be equal to 1, all picture storage buffers in the DPB are emptied without output of the pictures they contain.
4. When no_output_of_prior_pics_flag is not equal to 1 and is not inferred to be equal to 1, 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.13.5.2.3.

– Otherwise (the current picture is not an IRAP picture with NoRaslOutputFlag equal to 1 and with nuh_layer_id equal to 0), picture storage buffers containing a picture which are marked as "not needed for output" and "unused for reference" are emptied (without output). 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.13.5.2.3 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.

  – 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.

F.13.5.2.3 Picture decoding, marking, additional bumping, and storage

The following happens 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.13.5.2.3 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.

– 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..

F.13.5.2.4 "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 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 included one of the pictures that was cropped and output is emptied.

F.14 SEI messages

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

F.14.1 SEI message syntax

F.14.1.1 Layer dependency change SEI message syntax

layer_dependency_change(payloadSize) {	**Descriptor**
active_vps_id	u(4)
for(i = 1; i <= vps_max_layers_minus1; i++)	
for(j = 0; j < NumDirectRefLayers[i]; j++)	
ref_layer_disable_flag[i][j]	u(1)
}	

F.14.2 SEI message semantics

F.14.2.1 Layer dependency change SEI message semantics

This SEI message indicates that the layer dependency information changes starting with the current access unit containing the SEI message and is always interpreted with respect to the active VPS. When present, the layer dependency change SEI message applies to the target access unit set that consists of the current access unit and all the subsequent access units, in decoding order, until the next layer dependency change SEI message or the end of the CVS, whichever is earlier in decoding order.

NOTE 1 – The reference layers for any layer are always a subset of those indicated by the active VPS.

NOTE 2 – Layer dependency change SEI messages do not have a cumulative effect.

Some of the layers indicated by the following syntax elements may not be present in the target access unit set.

active_vps_id identifies an active VPS that contains the layer dependency relationship information. The value of active_vps_id shall be equal to the value of video_parameter_set_id of the active VPS for the VCL NAL units of the access unit containing the SEI message.

ref_layer_disable_flag[i][j] equal to 1 indicates that no picture with nuh_layer_id equal to RefLayerId[i][j] is present in any of the reference picture lists after reference picture list modification for pictures with nuh_layer_id equal to layer_id_in_nuh[i] within the target access unit set. ref_layer_disable_flag[i][j] equal to 0 indicates pictures with nuh_layer_id equal to RefLayerId[i][j] may be present in the reference picture lists after reference picture list modification for pictures with nuh_layer_id equal to layer_id_in_nuh[i] within the target access unit set. ref_layer_disable_flag[i][j] shall be equal to 1, if ref_layer_disable_flag[i][j] was equal to 1 in an earlier layer dependency change SEI message for the same CVS.

F.15 Video usability information

The specifications in Annex E apply.

Annex G

Picture management and profiles for multiview coding

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

This annex specifies syntax, semantics, decoding processes, picture management and profiles for multiview coding that use the syntax, semantics, and decoding process specified in clauses 2-9 for the slice segment data and all layers below it.

G.1 Scope

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

G.2 Normative references

The specifications in clause 2 apply.

G.3 Definitions

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

G.3.1 reference picture list: A list of reference pictures that is used for inter prediction or inter-layer prediction of a P or B slice.

G.4 Abbreviations

The specification in clause 4 apply.

G.5 Conventions

The specification in clause 5 apply.

G.6 Source, coded, decoded and output data formats, scanning processes, and neighbouring relationships

The specification in clause 6 apply.

G.7 Syntax and semantics

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

G.7.1 Method of specifying syntax in tabular form

The specifications in subclause F.7.1 apply.

G.7.2 Specification of syntax functions, categories, and descriptors

The specifications in subclause F.7.2 apply.

G.7.3 Syntax in tabular form

G.7.3.1 NAL unit syntax

The specifications in subclause F.7.3.1 and all its subclauses apply.

G.7.3.2 Raw byte sequence payloads, trailing bits, and byte alignment syntax

G.7.3.2.1 Video parameter set RBSP syntax

The specifications in subclause F.7.3.2.1 apply.

G.7.3.2.2 Sequence parameter set RBSP syntax

seq_parameter_set_rbsp( ) {	**Descriptor**
sps_video_parameter_set_id	u(4)
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)
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)
}	
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) {	
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( )	
}	

G.7.3.2.2.1 Sequence parameter set extension syntax

sps_extension( ) {	**Descriptor**
inter_view_mv_vert_constraint_flag	u(1)
}	

G.7.3.2.3 Picture parameter set RBSP syntax

The specifications in subclause F.7.3.2.3 apply.

G.7.3.2.4 Supplemental enhancement information RBSP syntax

The specifications in subclause F.7.3.2.4 apply.

G.7.3.2.5 Access unit delimiter RBSP syntax

The specifications in subclause F.7.3.2.5 apply.

G.7.3.2.6 End of sequence RBSP syntax

The specifications in subclause F.7.3.2.6 apply.

G.7.3.2.7 End of bitstream RBSP syntax

The specifications in subclause F.7.3.2.7 apply.

G.7.3.2.8 Filler data RBSP syntax

The specifications in subclause F.7.3.2.8 apply.

G.7.3.2.9 Slice segment layer RBSP syntax

The specifications in subclause F.7.3.2.9 apply.

G.7.3.2.10 RBSP slice segment trailing bits syntax

The specifications in subclause F.7.3.2.10 apply.

G.7.3.2.11 RBSP trailing bits syntax

The specifications in subclause F.7.3.2.11 apply.

G.7.3.2.12 Byte alignment syntax

The specifications in subclause F.7.3.2.12 apply.

G.7.3.3 Profile, tier and level syntax

The specifications in subclause F.7.3.3 apply.

G.7.3.4 Supplemental enhancement information message syntax

The specifications in subclause F.7.3.4 apply.

G.7.3.5 NAL unit syntax

The specifications in subclause F.7.3.5 apply.

G.7.3.6 Slice segment header syntax

The specifications in subclause F.7.3.6 apply.

G.7.3.7 Slice segment data syntax

The specifications in subclause F.7.3.7 and all its subclauses apply

G.7.4 Semantics

G.7.4.1 NAL unit semantics

The specifications in subclause F.7.4.1 and all its subclauses apply.

G.7.4.2 Raw byte sequence payloads, trailing bits, and byte alignment semantics

G.7.4.2.1 Video parameter set RBSP semantics

The specifications in subclause F.7.4.2.1 and all its subclauses apply.

G.7.4.2.2 Sequence parameter set RBSP

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

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. When vps_max_layers_minus1 is greater than 0, sps_extension_flag shall be equal to 1.

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.

G.7.4.2.2.1 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.

G.7.4.2.3 Picture parameter set RBSP semantics

The specifications in subclause F.7.4.2.3 apply.

G.7.4.2.4 Supplemental enhancement information RBSP semantics

The specifications in subclause F.7.4.2.4 apply.

G.7.4.2.5 Access unit delimiter RBSP semantics

The specifications in subclause F.7.4.2.5 apply.

G.7.4.2.6 End of sequence RBSP semantics

The specifications in subclause F.7.4.2.6 apply.

G.7.4.2.7 End of bitstream RBSP semantics

The specifications in subclause F.7.4.2.7 apply.

G.7.4.2.8 Filler data RBSP semantics

The specifications in subclause F.7.4.2.8 apply.

G.7.4.2.9 Slice segment layer RBSP semantics

The specifications in subclause F.7.4.2.9 apply.

G.7.4.2.10 RBSP slice segment trailing bits semantics

The specifications in subclause F.7.4.2.10 apply.

G.7.4.2.11 RBSP trailing bits semantics

The specifications in subclause F.7.4.2.11 apply.

G.7.4.2.12 Byte alignment semantics

The specifications in subclause F.7.4.2.12 apply.

G.7.4.3 Profile, tier and level semantics

The specifications in subclause F.7.4.3 apply.

G.7.4.4 Supplemental enhancement information message semantics

The specifications in subclause F.7.4.4 apply.

G.7.4.5 NAL unit semantics

The specifications in subclause F.7.4.5 apply.

G.7.4.6 Slice segment header semantics

G.7.4.6.1 General slice segment header semantics

The specifications in subclause F.7.4.6.1 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."

G.7.4.6.2 Short-term reference picture set semantics

The specifications in subclause F.7.4.6.2 apply.

G.7.4.6.3 Reference picture list modification semantics

The specifications in subclause F.7.4.6.3 apply with following modifications.

– Equation (7-58) specifying the derivation of NumPocTotalCurr is replaced by:

[Ed. (YK): With the addition of the inter-layer stuff, this variable name gets confusing. Maybe it should be changed to "NumPicTotalCurr". Purely editorial - can be done later on, even in Version 1.]

```
NumPocTotalCurr = 0
for( i = 0; i < NumNegativePics[ CurrRpsIdx ]; i++)
  if(UsedByCurrPicS0[ CurrRpsIdx ][ i ] = = 1)
      NumPocTotalCurr++
for( i = 0; i < NumPositivePics[ CurrRpsIdx ]; i++)                    (G-1)
  if(UsedByCurrPicS1[ CurrRpsIdx ][ i ] = = 1)
      NumPocTotalCurr++
for( i = 0; i < num_long_term_sps + num_long_term_pics; i++ )
  if( UsedByCurrPicLt[ i ] = = 1)
      NumPocTotalCurr++
NumPocTotalCurr += NumDirectRefLayers[ LayerIdInVps[ nuh_layer_id ] ]
```

G.7.4.6.4 Weighted prediction parameters semantics

The specifications in subclause F.7.4.6.4 apply.

G.7.4.7 Slice segment data semantics

The specifications in subclause F.7.4.7 and all its subclauses apply

G.8 Decoding processes

G.8.1 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 following steps specify the decoding processes using syntax elements in the slice segment layer and above:
   – Prior to decoding the first slice of the current picture, subclause G.8.1.1 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 G.8.3.4 is invoked for derivation of reference picture list 0 (RefPicList0), and when decoding a B slice, reference picture list 1 (RefPicList1).
   – After the decoding of the last slice of the current picture, the marking process for ending the decoding of a coded picture with nuh_layer_id greater than 0 specified in subclause G.8.1.2 is invoked.
3. The following ordered steps specify the decoding processes using syntax elements in the coding tree unit layer and above.
   – The processes in subclauses 8.4, 8.5, 8.6, and 8.7 are invoked for the decoding processes using syntax elements in the coding tree unit layer and above.

G.8.1.1 Decoding process for inter-layer reference picture set

Output of this process is an updated list of inter-layer pictures RefPicSetInterLayer.

The list RefPicSetInterLayer is first emptied and then derived as follows.

```
for( i = 0; i < NumDirectRefLayers[ LayerIdInVps[ nuh_layer_id ] ]; i++ ) {
    RefPicSetInterLayer[ i ]  = the picture with picture order count equal to PicOrderCnt and
        nuh_layer_id equal to RefLayerId[ LayerIdInVps[ nuh_layer_id ][ i ] ]
    RefPicSetInterLayer[ i ] is marked as "used for long-term reference"
}
```

G.8.1.2 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 < NumDirectRefLayers[LayerIdInVps[nuh_layer_id]]; i++)
RefPicSetInterLayer[i] is marked as "used for short-term reference"

G.8.2 NAL unit decoding process

G.8.3 Slice decoding processes

G.8.3.1 (void)

(void)

G.8.3.2 (void)

(void)

G.8.3.3 (void)

(void)

G.8.3.4 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 8.5.3.2.1. 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 list RefPicList0, and for B slices RefPicList1, are derived as follows.

The variable NumRpsCurrTempList0 is set equal to Max(num_ref_idx_l0_active_minus1 + 1, NumPocTotalCurr) 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 < NumPocStCurrAfter && rIdx < NumRpsCurrTempList0; rIdx++, i++ )          (G-2)
        RefPicListTemp0[ rIdx ] = RefPicSetStCurrAfter[ i ]
    for( i = 0; i < NumPocLtCurr && rIdx < NumRpsCurrTempList0; rIdx++, i++ )
        RefPicListTemp0[ rIdx ] = RefPicSetLtCurr[ i ]
    for( i = 0; i < NumDirectRefLayers[ LayerIdInVps[ nuh_layer_id ] ]; rIdx++, i++ )
        RefPicListTemp0[ rIdx ] = RefPicSetInterLayer[ i ]

}
```

The list RefPicList0 is constructed as follows:

```
for( rIdx = 0; rIdx <= num_ref_idx_l0_active_minus1; rIdx++)                              (G-3)
    RefPicList0[ rIdx ] = ref_pic_list_modification_flag_l0 ? RefPicListTemp0[ list_entry_l0[ rIdx ] ] :
                                                              RefPicListTemp0[ rIdx ]
```

When the slice is a B slice, the variable NumRpsCurrTempList1 is set equal to Max(num_ref_idx_l1_active_minus1 + 1, NumPocTotalCurr) 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 < NumPocStCurrBefore && rIdx < NumRpsCurrTempList1; rIdx++, i++ )          (G-4)
        RefPicListTemp1[ rIdx ] = RefPicSetStCurrBefore[ i ]
    for( i = 0; i < NumPocLtCurr && rIdx < NumRpsCurrTempList1; rIdx++, i++ )
        RefPicListTemp1[ rIdx ] = RefPicSetLtCurr[ i ]
    for( i = 0; i< NumDirectRefLayers[ LayerIdInVps[ nuh_layer_id ] ] ; rIdx++, i++ )
        RefPicListTemp1[ rIdx ] = RefPicSetInterLayer[ i ]
}
```

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

```
for( rIdx = 0; rIdx <= num_ref_idx_l1_active_minus1; rIdx++)                    (G-5)
    RefPicList1[ rIdx ] = ref_pic_list_modification_flag_l1 ? RefPicListTemp1[ list_entry_l1[ rIdx ] ] :
                                                              RefPicListTemp1[ rIdx ]
```

G.8.4 Decoding process for coding units coded in intra prediction mode

The specifications in subclause 8.4 apply.

G.8.5 Decoding process for coding units coded in inter prediction mode

The specifications in subclause 8.5 apply.

G.8.6 Scaling, transformation and array construction process prior to deblocking filter process

The specifications in subclause 8.6 apply.

G.8.7 In-loop filter process

The specifications in subclause 8.7 apply.

G.9 Parsing process

The specifications in clause 9 apply.

G.10 Specification of bitstream subsets

The specifications in clause 10 apply.

G.11 Profiles, tiers, and levels

G.11.1 Profiles

G.11.1.1 General

TBD.

G.11.1.2 Stereo Main profile

Bitstreams conforming to the Stereo Main profile shall obey the following constraints:

– The sub-bitstream resulting from the sub-bitstream extraction process with any value of tIdTarget and a value of 0 in layerIdListTarget as inputs shall conform to the Main profile.

– The bitstream shall contain one layer with nuh_layer_id equal to i for which ViewId[i] is greater than 0.

– When ViewId[i] is greater than 0, inter_view_mv_vert_constraint_flag shall be equal to 1 in the sps_extension( ) syntax structure of the active layer SPS of any coded pictures with nuh_layer_id equal to i.

[Ed. (GT): For future scalable extensions additional constraints disallowing scalability might be added here.]

G.11.2 Tiers and levels

TBD

G.12 Byte stream format

The specifications in subclause F.12 apply.

G.13 Hypothetical reference decoder

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

G.14 SEI messages

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

G.15 Video usability information

The specifications in subclause F.15 apply.