|  |  |
| --- | --- |
| **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**  1st Meeting: Stockholm, SE, 16–20 July 2012 | Document: JCT2-A1007\_r1 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Solutions considered for NAL unit header and video parameter set for HEVC extensions** | | |
| *Status:* | Output Document of JCT-3V | | |
| *Purpose:* | Draft | | |
| *Author(s) or Contact(s):* | Ye-Kui Wang, Ying Chen 5775 Morehouse Drive San Diego, CA 92121, USA | Tel: Email: | 1-858-651-8345 [yekuiw@qualcomm.com](mailto:yekuiw@qualcomm.com)  1-858-845-6589 [cheny@qualcomm.com](mailto:cheny@qualcomm.com) |
| *Source:* | Editor | | |

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

# Abstract

This document includes the specification of the "straw man" designs, jointly agreed by JCT-VC and JCT-3V, for NAL unit header and video parameter set for HEVC extensions. Two alternative video parameter set designs are provided, one for each of the following two approaches:

1. Approach 1: The mapping of each value of the 6-bit layer\_id (i.e., the reserved\_zero\_6bits in the HEVC base specification) in the NAL unit header to specific scalability dimension IDs (e.g. view\_order\_index, depth\_flag, dependency\_id and quality\_id) is signalled in a look-up table in the video parameter set.
2. Approach 2: The 6-bit layer\_id (i.e., the reserved\_zero\_6bits in the HEVC base specification) in the NAL unit header is partitioned into specific scalability dimension IDs (e.g. view\_order\_index, depth\_flag, dependency\_id and quality\_id) and the partitioning method is signalled in the video parameter set.

It is expected that an aligned design is to be used for all the potential multiview, 3DV and scalability extensions of HEVC.

The NAL unit header and video parameter set designs are provided in Sections 1 and 2, respectively.

# NAL unit syntax and semantics

Syntax changes relative to the NAL unit syntax in the HEVC base specification are highlighted.

|  |  |
| --- | --- |
| nal\_unit( NumBytesInNALunit ) { | Descriptor |
| **forbidden\_zero\_bit** | f(1) |
| **nal\_unit\_type** | u(6) |
| **layer\_id** // reserved\_zero\_6bits in the base spec | u(6) |
| **temporal\_id\_plus1** | u(3) |
| NumBytesInRBSP = 0 |  |
| for( i = 2; i < NumBytesInNALunit; i++ ) { |  |
| if( i + 2 < NumBytesInNALunit && next\_bits( 24 ) = = 0x000003 ) { |  |
| **rbsp\_byte[** NumBytesInRBSP++ **]** | b(8) |
| **rbsp\_byte[** NumBytesInRBSP++ **]** | b(8) |
| i += 2 |  |
| **emulation\_prevention\_three\_byte** /\* equal to 0x03 \*/ | f(8) |
| } else |  |
| **rbsp\_byte[** NumBytesInRBSP++ **]** | b(8) |
| } |  |
| } |  |

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

Semantics for other syntax elements are as specified in the HEVC base specification.

# Video parameter set

## Video parameter set RBSP syntax and semantics

Syntax changes relative to the video parameter set RBSP syntax in the HEVC base specification are highlighted.

|  |  |
| --- | --- |
| video\_parameter\_set\_rbsp( ) { | Descriptor |
| **video\_parameter\_set\_id** | u(4) |
| **vps\_temporal\_id\_nesting\_flag** | u(1) |
| **reserved\_zero\_2bits** | u(2) |
| **max\_num\_layers\_minus1** //reserved\_zero\_6bits in the base spec | u(6) |
| **vps\_max\_sub\_layers\_minus1** | u(3) |
| profile\_level( 1, vps\_max\_sub\_layers\_minus1 ) |  |
| **next\_essential\_info\_byte\_offset** //reserved\_zero\_12bits in the base spec | u(12) |
| for( i = 0; i <= vps\_max\_sub\_layers\_minus1; i++ ) { |  |
| **vps\_max\_dec\_pic\_buffering[** i **]** | ue(v) |
| **vps\_max\_num\_reorder\_pics[** i **]** | ue(v) |
| **vps\_max\_latency\_increase[** i **]** | ue(v) |
| } |  |
| **num\_hrd\_parameters** | ue(v) |
| for( i = 0; i < num\_hrd\_parameters; i++ ) { |  |
| if( i > 0 ) |  |
| op\_point( i ) |  |
| hrd\_parameters( i = = 0, vps\_max\_sub\_layers\_minus1 ) |  |
| } |  |
| **bit\_equal\_to\_one** | u(1) |
| vps\_extension( ) |  |
| **vps\_extension\_flag** | u(1) |
| if( vps\_extension\_flag ) |  |
| while( more\_rbsp\_data( ) ) |  |
| **vps\_extension\_data\_flag** | u(1) |
| rbsp\_trailing\_bits( ) |  |
| } |  |

**max\_num\_layers\_minus1** plus 1 specifies the maximum number of layers in the coded video sequences referring to the video parameter set.

**next\_essential\_info\_byte\_offset** specifies the byte offset of the next set of fixed-length coded information in the video parameter set NAL unit, starting from the beginning of the NAL unit.

NOTE 3 –Video parameter set information for non-base layer or view starts from a byte-aligned position of the video parameter set NAL unit, with fixed-length coded information that is essential for session negotiation and/or capability exchange. The byte offset specified by next\_essential\_info\_byte\_offset would then help to locate and access those essential information in the video parameter set 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 video parameter set that is essential for session negotiation and/or capability exchange.

**max\_num\_layers\_minus1** plus 1 specifies the maximum number of layers in the coded video sequences referring to the video parameter set.

**bit\_equal\_to\_one** shall be equal to 1.

Semantics for other syntax elements are as specified in the HEVC base specification.

## Video parameter set extension syntax and semantics

### Approach 1 (layer\_id not partitioned into specific scalability dimension IDs)

|  |  |
| --- | --- |
| vps\_extension( ) { | Descriptor |
| while( !byte\_aligned( ) ) |  |
| **vps\_extension\_byte\_alignment\_reserved\_one\_bit** | u(1) |
| // layer specific information |  |
| for( i = 1; i <= vps\_max\_layers\_minus1; i++ ) { |  |
| // mapping of layer ID to scalability dimension IDs |  |
| **num\_dimensions\_minus1**[ i ] | u(4) |
| for( j = 0; j <= num\_dimensions\_minus1; j++ ) { |  |
| **dimension\_type**[ i ][ j ] | u(4) |
| **dimension\_id**[ i ][ j ] | u(8) |
| } |  |
| // layer dependency |  |
| **num\_direct\_ref\_layers**[ i ] | u(6) |
| for( j = 0; j < num\_direct\_ref\_layers[ i ]; j++ ) |  |
| **ref\_layer\_id**[ i ][ j ] | u(6) |
| } |  |
| } |  |

**vps\_extension\_byte\_alignment\_reserved\_one\_bit** shall be equal to 1.

**num\_dimensions\_minus1**[ i ] plus 1 specifies the number of dimension types and IDs signalled for the i-th layer.

**dimension\_type**[ i ][ j ] specifies the j-th scalability dimension type of the i-th layer, which has layer\_id equal to i, as specified in the following table [Ed. (YK): Editorial improvements may be needed.]:

|  |  |
| --- | --- |
| dimension\_type[ i ][ j ] | dimension\_id[ i ][ j ] |
| 0 | view order idx |
| 1 | depth flag |
| 2 | dependency ID |
| 3 | quality ID |
| 4..15 | reserved |

The value of dimension\_type[ i ][ j ] shall be in the range of 0 to 3, inclusive, in bitstreams conforming to this Recommendation | International Standard. Other values for dimension\_type[ i ][ j ] are reserved for future use by ITU-T | ISO/IEC. Decoders shall ignore values of dimension\_type[ i ][ j ] that are not in the range of 0 to 3, inclusive.

**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.

**num\_direct\_ref\_layers**[ i ] specifies the number of layers the i-th layer directly depends on. [Ed.(YK): Add the exact meaning of a layer directly depending on another layer.]

**ref\_layer\_id**[ i ][ j ] identifies the j-th layer the i-th layer directly depends on.

### Approach 2 (layer\_id partitioned into specific scalability dimension IDs)

|  |  |
| --- | --- |
| vps\_extension( ) { | Descriptor |
| while( !byte\_aligned( ) ) |  |
| **vps\_extension\_byte\_alignment\_reserved\_one\_bit** | u(1) |
| // scalability type and layer\_id partitioning method |  |
| **scalability\_type** | u(4) |
| for( i = 0; i < MaxDim( scalability\_type ); i++ ) |  |
| **layer\_id\_dim\_len**[ i ] | u(3) |
| // layer specific information |  |
| for( i = 0; i <= max\_num\_layers\_minus1; i++ ) { |  |
| **vps\_layer\_id[** i **]** | u(6) |
| // layer dependency |  |
| **num\_direct\_ref\_layers**[ i ] | u(6) |
| for( j = 0; j < num\_direct\_ref\_layers[ i ]; j++ ) |  |
| **ref\_layer\_id**[ i ][ j ] | u(6) |
| } |  |
| } |  |

**vps\_extension\_byte\_alignment\_reserved\_one\_bit** shall be equal to 1.

**scalability\_type** specifies the scalability types in use in the coded video sequence and the dimensions signaled through layer\_id in the NAL unit header. When scalability\_type is equal to 0, the coded video sequence conforms to the base HEVC specification, thus layer\_id of all NAL units is equal to 0 and there are no NAL units belonging to an enhancement layer or view. Higher values of scalability\_type are interpreted as shown in the following table:

|  |  |  |
| --- | --- | --- |
| **scalability\_type** | **MaxDim(scalability\_type)** | **Scalability dimensions** |
| 0 | 1 | none (base HEVC) |
| 1 | 2 | spatial and quality |
| 2 | 3 | spatial, quality, reserved |
| 3 | 4 | spatial, quality, reserved, reserved |
| 4 | 2 | multiview and depth |
| 5 | 3 | multiview, depth, reserved |
| 6 | 4 | multiview, depth, reserved, reserved |
| 7 | 4 | multiview, spatial, quality and depth |
| 8 | 5 | multiview, spatial, quality, depth, reserved |
| 9 | 6 | multiview, spatial, quality, depth, reserved, reserved |
| 10...15 | reserved | reserved |

**layer\_id\_dim\_len**[ i ] specifies the length, in bits, of the i-th scalability dimension ID. The sum of the values layer\_id\_dim\_len[ i ] for all i values in the range of 0 to 7 shall be less than or equal to 6.

**vps\_layer\_id**[ i ] specifies the value of layer\_id of the i-th layer to which the following layer dependency information applies.

**num\_direct\_ref\_layers**[ i ] specifies the number of layers the i-th layer directly depends on. [Ed.(YK): Add the exact meaning of a layer directly depending on another layer.]

**ref\_layer\_id**[ i ][ j ] identifies the j-th layer the i-th layer directly depends on.