|  |  |
| --- | --- |
| **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-A0177 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **BoG on high-level syntax for extension planning** | | |
| *Status:* | Input Document | | |
| *Purpose:* | Report | | |
| *Author(s) or Contact(s):* | Ye-Kui Wang 5775 Morehouse Drive San Diego, CA 92121, USA | Tel: Email: | 1-858-651-8345 [yekuiw@qualcomm.com](mailto:yekuiw@qualcomm.com) |
| *Source:* | BoG on high-level syntax for extension planning | | |

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

# Abstract

The following was concluded by the JCT-VC and JCT-3V on Tuesday, 17 July, 2012:

*In future extensions, the reserved\_zero\_6bits in the NAL unit header has been anticipated to be used as a layer ID, for which the VPS would identify the purpose. Within that concept, either:*

1. *Approach 1: The mapping of a value of the 6 bits to that information would be a general LUT in the VPS.*
2. *Approach 2: There would be a partitioning of the 6 bits into distinct bit fields – e.g., dependency ID and quality ID and view ID and depth flag*

*For either approach, combining the 6-bit reserved\_zero\_6bits with the 3-bit temporal\_id\_plus1 would allow further flexibility.*

*It was suggested that both approaches could actually be used – e.g., controlled by a flag. It was suggested that it would be easier to keep some bits as reserved when using the 2nd approach. The base spec does not necessarily need to have any of this defined, except for a scheme that would combine reserved\_zero\_6bits and temporal\_id\_plus1.*

Based on the above, a BoG on high-level syntax for extension planning was created to

* + to produce a document containing a "straw man" for a flexible approach – particularly focusing on the 1st method
  + to collect in the document some possible alternatives

The BoG firstly met from 18:20 to 20:30 on Tuesday, 17 July, 2012, with the following agenda:

1. Overview of NAL unit, VPS,VUI and HRD syntaxes taking into account adoptions at this meeting
2. Collect a list of questions to be answered for each of the following two approaches, as well as other questions
3. Review 3V VPS design in the software
4. Discuss and tentatively conclude on each of the questions

At the first BoG meeting, questions specific for both approaches and other questions were collected and the list of questions for the 1st approach were discussed and tentatively concluded with a simple answer (see Section 1).

The BoG met the second time from 11:45 to 14:00 on Wednesday, 18 July, 2012. All the questions remained from the 1st BoG meeting were discussed and answers to these questions were agreed. At the second meeting, the BoG reviewed the documents listed in Section 4, and further agreed on the following:

* To include in the VPS extension profile and level for all operation points involving at least one enhancement layer, with similar function as for the base layer for temporal subsets
* It is not needed to include HRD information in the VPS extension as it is already possible to include HRD parameters for all operation points in the VPS base

The BoG met the 3rd time from 11:15 to 12:30 on July 19, 2012, and agreed two sets of "straw man" designs, one for each of the two approaches, as described in Section 3.

At the third BoG meeting, the BoG was not able to agree on the syntax for inclusion in the "straw man" designs of profile and level information for operation points including at least one enhancement layer in VPS extensions, though there is already inclusion of profile and level information for the base layer operation point, and the BoG agreed at its 2nd meeting to include such information in the VPS extension. The BoG suggestion that this is discussed.

The BoG makes the following recommendation:

* Adopt the agreed two sets of VPS extension designs, one for each of the two approaches, as described in Section 3 of the BoG report, as the "straw man" designs.

The BoG suggests to discuss on whether to include, in the "straw man" designs, the syntax and semantics for inclusion of profile and level information for operation points including at least one enhancement layer, as described in Section 4.

For information, HEVC base spec NAL unit, VPS,VUI and HRD syntaxes taking into account adoptions at this meeting (subject to minor changes per the detailed JCT-VC meeting notes) have been provided in Section 2, and the list of input documents are listed in Section 5.

# Questions and answers

## Questions to be answered for the 1st approach

The list of questions collected for the 1st approach and the answers agreed by the BoG at its first meeting are as follows.

1. Whether operation point (OP) information should be included in the VPS?
   1. OP dependency information? Not right now.
   2. Other OP information
2. Whether to include layer dependency information into VPS?
   1. Yes, for both HSVC H3DV
3. Whether to include representation format information?
   1. Spatial resolutions, bit depth, color format? Not right now.
4. Bitrate and frame rate information? Not right now.
   1. Also for the base spec?
5. Other SPS syntax elements, e.g., VUI? Not right now.
   1. Also for the base spec?
6. How many extensibility types to support and how?
   1. Beyond dependency\_id, quality\_id, view\_id (view order index), depth\_flag?
      1. Extensibility would be needed - we don't specify any specific other dimension types.

Example: extension\_type u(v)

0: HSVC, with dependency\_id (3 bits), quality\_id (4 bits) or just 8 bits for all IDs

1: H3DV, with view\_idx (4 bits), depth\_flag (1 bit) just 8 bits for view\_idx

Other values reserved

* 1. Extensibility type backward compatibility - Yes

1. Also include temporal\_id (which means we need to change the base spec also to combine temporal\_id\_plus1 and the reserved\_zero\_6bits)? No.
   1. This solution would not enable adaptation based on the 9-bit ID without parsing the LUT
2. Whether to add a version number? Not right now

## Questions to be answered for the 2nd approach

The following questions have been collected for the 2nd approach at the first BoG meeting. The BoG planned to meet further to discuss and concludes on these questions.

1. Fixed partitioning or configurable partitioning? Configurable.
2. 6 bits or 9 bits (or even more)?

It seems that for approach 2, wherein partitioning of the 6 bit to fields is already done, it is more favorable to combine temporal\_id too, knowing that combining the 9 bits altogether, this solution would not enable adaptation based on the 9-bit ID without parsing the LUT.

1. Whether the partitioning method is signalled in a NAL unit header (e.g., the VPS NAL unit header, 2nd byte) or in a NAL unit payload, or both?

The only difference here is that putting the information VPS NAL unit header is easier to access, as it is earlier and would not be contemplated by start code emulation prevention process, while putting in the NAL unit payload can avoid irregular syntax design for NAL unit header syntax. It seems that putting the information into the NAL unit payload is more favorable by the BoG participants.

## Other related questions to be answered

The following other related questions have been collected at the first BoG meeting. The BoG planned to meet further to discuss and concludes on these questions.

1. Coexistence of scalability and 3DV?
   1. Support that or not? Yes, as that is included in the requirement. At least the architecture of the design should not preclude it.
   2. If yes, to what extent? The design should be extensible in a way that backward compatibility is possible for a possible future extension that supports the coexistence.
2. Whether to support multi-standard extensions? Yes, as that is included in the requirement. At least the architecture of the design should not preclude it.

Suggestion: To have a flag in the VPS to indicate whether the base layer is HEVC or not. If not, signal which standard. - Not now, as we don't have a complete solution on the table, e.g., if the base layer is not HEVC, what is the NAL unit header syntax and so on.

Question: Should this indication, if any, be included the base spec? Probably not.

1. Whether to allow use of the reserved bits for more than layer IDs?
   1. The reserved bits are generally for layer IDs
   2. However, within the scope of approach 2, if only a subset of the reserved bits are sufficient to represent all the scalability dimension IDs, then the remaining bits can be reserved or used for other purposes, when needed

# HEVC base specification syntaxes

## NAL unit header syntax

|  |  |
| --- | --- |
| nal\_unit( NumBytesInNALunit ) { | Descriptor |
| **forbidden\_zero\_bit** | f(1) |
| **nal\_unit\_type** | u(6) |
| **reserved\_zero\_6bits** | 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) |
| } |  |
| } |  |

## Video parameter set RBSP syntax

|  |  |
| --- | --- |
| video\_parameter\_set\_rbsp( ) { | Descriptor |
| **video\_parameter\_set\_id** | u(4) |
| **vps\_temporal\_id\_nesting\_flag** | u(1) |
| **reserved\_zero\_2bits** | u(2) |
| **reserved\_zero\_6bits** | u(6) |
| **vps\_max\_sub\_layers\_minus1** | u(3) |
| profile\_level( 1, vps\_max\_sub\_layers\_minus1 ) |  |
| **reserved\_zero\_12bits** | 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 ) |  |
| } |  |
| **vps\_extension\_flag** | u(1) |
| if( vps\_extension\_flag ) |  |
| while( more\_rbsp\_data( ) ) |  |
| **vps\_extension\_data\_flag** | u(1) |
| rbsp\_trailing\_bits( ) |  |
| } |  |

### Profile and level syntax

|  |  |
| --- | --- |
| profile\_level( ProfilePresentFlag, MaxNumSubLayersMinus1 ) { |  |
| if( ProfilePresentFlag ) { |  |
| **profile\_space** | u(3) |
| **profile\_idc** | u(5) |
| for( i = 0; i < 32; i++ ) |  |
| **profile\_compatability\_flag[** i **]** | u(1) |
| **constraint\_flags** | u(16) |
| } |  |
| **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( ProfilePresentFlag && sub\_layer\_profile\_present\_flag[ i ] ) { |  |
| **sub\_layer\_profile\_space[** i **]** | u(3) |
| **sub\_layer\_profile\_idc[** i **]** | u(5) |
| for( j = 0; j < 32; j++ ) |  |
| **sub\_layer\_profile\_compatability\_flag[** i **][** j **]** | u(1) |
| **sub\_layer\_constraint\_flags[** i **]** | u(16) |
| } |  |
| if( sub\_layer\_level\_present\_flag[ i ] ) |  |
| **sub\_layer\_level\_idc**[ i ] | u(8) |
| } |  |
| } |  |

### Operating point syntax

|  |  |
| --- | --- |
| op\_point( opIdx ) { | Descriptor |
| **op\_num\_layer\_id\_values\_minus1**[ opIdx ] | ue(v) |
| for( i = 0; i <= op\_num\_layer\_id\_values\_minus1; i++ ) |  |
| **op\_layer\_id**[ opIdx ][ i ] | u(6) |
| } |  |

## SPS syntax

|  |  |
| --- | --- |
| seq\_parameter\_set\_rbsp( ) { | Descriptor |
| **video\_parameter\_set\_id** | u(4) |
| **sps\_max\_sub\_layers\_minus1** | u(3) |
| **reserved\_zero\_bit** | u(1) |
| profile\_level( 0, sps\_max\_sub\_layers\_minus1 ) |  |
| **seq\_parameter\_set\_id** | ue(v) |
| **chroma\_format\_idc** | ue(v) |
| if( chroma\_format\_idc = = 3 ) |  |
| **separate\_colour\_plane\_flag** | u(1) |
| **sps\_max\_temporal\_layers\_minus1** | u(3) |
| **pic\_width\_in\_luma\_samples** | ue(v) |
| **pic\_height\_in\_luma\_samples** | ue(v) |
| **pic\_cropping\_flag** | u(1) |
| if( pic\_cropping\_flag ) { |  |
| **pic\_crop\_left\_offset** | ue(v) |
| **pic\_crop\_right\_offset** | ue(v) |
| **pic\_crop\_top\_offset** | ue(v) |
| **pic\_crop\_bottom\_offset** | ue(v) |
| } |  |
| **bit\_depth\_luma\_minus8** | ue(v) |
| **bit\_depth\_chroma\_minus8** | ue(v) |
| [Ed. (BB): chroma bit depth present in HM software but not used further ] |  |
| **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\_max\_pic\_order\_cnt\_lsb\_minus4** | ue(v) |
| for( i = 0; i <= sps\_max\_temporal\_layers\_minus1; i++ ) { |  |
| **sps\_max\_dec\_pic\_buffering[** i **]** | ue(v) |
| **sps\_num\_reorder\_pics[** i **]** | ue(v) |
| **sps\_max\_latency\_increase[** i **]** | ue(v) |
| } |  |
| **restricted\_ref\_pic\_lists\_flag** | u(1) |
| if( restricted\_ref\_pic\_lists\_flag ) |  |
| **lists\_modification\_present\_flag** | u(1) |
| **log2\_min\_coding\_block\_size\_minus3** | ue(v) |
| **log2\_diff\_max\_min\_coding\_block\_size** | ue(v) |
| **log2\_min\_transform\_block\_size\_minus2** | ue(v) |
| **log2\_diff\_max\_min\_transform\_block\_size** | ue(v) |
| if( pcm\_enabled\_flag ) { |  |
| **log2\_min\_pcm\_coding\_block\_size\_minus3** | ue(v) |
| **log2\_diff\_max\_min\_pcm\_coding\_block\_size** | ue(v) |
| } |  |
| **max\_transform\_hierarchy\_depth\_inter** | ue(v) |
| **max\_transform\_hierarchy\_depth\_intra** | ue(v) |
| **scaling\_list\_enable\_flag** | u(1) |
| if( scaling\_list\_enable\_flag ) { |  |
| **sps\_scaling\_list\_data\_present\_flag** | u(1) |
| if( sps\_scaling\_list\_data\_present\_flag ) |  |
| scaling\_list\_param( ) |  |
| } |  |
| **chroma\_pred\_from\_luma\_enabled\_flag** | u(1) |
| **transform\_skip\_enabled\_flag** | u(1) |
| **seq\_loop\_filter\_across\_slices\_enabled\_flag** | u(1) |
| **asymmetric\_motion\_partitions\_enabled\_flag** | u(1) |
| **nsrqt\_enabled\_flag** | u(1) |
| **sample\_adaptive\_offset\_enabled\_flag** | u(1) |
| **adaptive\_loop\_filter\_enabled\_flag** | u(1) |
| if( pcm\_enabled\_flag ) |  |
| **pcm\_loop\_filter\_disable\_flag** | u(1) |
| **sps\_temporal\_id\_nesting\_flag** | u(1) |
| [Ed. (BB): x y padding syntax missing here, present in HM software ] |  |
| **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) |
| **sps\_temporal\_mvp\_enable\_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 ) |  |
| while( more\_rbsp\_data( ) ) |  |
| **sps\_extension\_data\_flag** | u(1) |
| rbsp\_trailing\_bits( ) |  |
| } |  |

## VUI and HRD parameters syntaxes

### VUI syntax

|  |  |
| --- | --- |
| vui\_parameters( ) { | Descriptor |
| **aspect\_ratio\_info\_present\_flag** | u(1) |
| if( aspect\_ratio\_info\_present\_flag ) { |  |
| **aspect\_ratio\_idc** | u(8) |
| if( aspect\_ratio\_idc = = Extended\_SAR ) { |  |
| **sar\_width** | u(16) |
| **sar\_height** | u(16) |
| } |  |
| } |  |
| **overscan\_info\_present\_flag** | u(1) |
| if( overscan\_info\_present\_flag ) |  |
| **overscan\_appropriate\_flag** | u(1) |
| **video\_signal\_type\_present\_flag** | u(1) |
| if( video\_signal\_type\_present\_flag ) { |  |
| **video\_format** | u(3) |
| **video\_full\_range\_flag** | u(1) |
| **colour\_description\_present\_flag** | u(1) |
| if( colour\_description\_present\_flag ) { |  |
| **colour\_primaries** | u(8) |
| **transfer\_characteristics** | u(8) |
| **matrix\_coefficients** | u(8) |
| } |  |
| } |  |
| **chroma\_loc\_info\_present\_flag** | u(1) |
| if( chroma\_loc\_info\_present\_flag ) { |  |
| **chroma\_sample\_loc\_type\_top\_field** | ue(v) |
| **chroma\_sample\_loc\_type\_bottom\_field** | ue(v) |
| } |  |
| **neutral\_chroma\_indication\_flag** | u(1) |
| **field\_seq\_flag** | u(1) |
| hrd\_parameters( 1, sps\_max\_sub\_layers\_minus1 ) |  |
| **bitstream\_restriction\_flag** | u(1) |
| if( bitstream\_restriction\_flag ) { |  |
| **tiles\_fixed\_structure\_flag** | u(1) |
| **motion\_vectors\_over\_pic\_boundaries\_flag** | u(1) |
| **max\_bytes\_per\_pic\_denom** | ue(v) |
| **max\_bits\_per\_mincu\_denom** | ue(v) |
| **log2\_max\_mv\_length\_horizontal** | ue(v) |
| **log2\_max\_mv\_length\_vertical** | ue(v) |
| } |  |
| } |  |

### HRD parameters syntax

|  |  |
| --- | --- |
| hrd\_parameters( commonInfPresentFlag, MaxNumSubLayersMinus1 ) { | Descriptor |
| if( commonInfPresentFlag ) { |  |
| **timing\_info\_present\_flag** | u(1) |
| if( timing\_info\_present\_flag ) { |  |
| **num\_units\_in\_tick** | u(32) |
| **time\_scale** | u(32) |
| } |  |
| **nal\_hrd\_parameters\_present\_flag** | u(1) |
| **vcl\_hrd\_parameters\_present\_flag** | u(1) |
| if( nal\_hrd\_parameters\_present\_flag | | vcl\_hrd\_parameters\_present\_flag ){ |  |
| **sub\_pic\_cpb\_params\_present\_flag** | u(1) |
| if( sub\_pic\_cpb\_params\_present\_flag ) |  |
| **tick\_divisor\_minus2** | u(8) |
| **bit\_rate\_scale** | u(4) |
| **cpb\_size\_scale** | u(4) |
| **initial\_cpb\_removal\_delay\_length\_minus1** | u(5) |
| **cpb\_removal\_delay\_length\_minus1** | u(5) |
| **dpb\_output\_delay\_length\_minus1** | u(5) |
| } |  |
| } |  |
| for( i = 0; i <= MaxNumSubLayersMinus1; i++ ) { |  |
| **fixed\_pic\_rate\_flag**[ i ] | u(1) |
| if( fixed\_pic\_rate\_flag[ i ] ) |  |
| **pic\_duration\_in\_tc\_minus1**[ i ] | ue(v) |
| **low\_delay\_hrd\_flag**[ i ] | u(1) |
| **cpb\_cnt\_minus1**[ i ] | ue(v) |
| if( nal\_hrd\_parameters\_present\_flag ) |  |
| hrd\_parameters\_sub\_layer( i ) |  |
| if( vcl\_hrd\_parameters\_present\_flag ) |  |
| hrd\_parameters\_sub\_layer( i ) |  |
| } |  |
| } |  |

### Sub-layer HRD parameters syntax

|  |  |
| --- | --- |
| hrd\_parameters\_sub\_layer( tId ) { | Descriptor |
| for( SchedSelIdx = 0; SchedSelIdx <= cpb\_cnt\_minus1[ i ]; SchedSelIdx++ ) { |  |
| **bit\_rate\_value\_minus1[** SchedSelIdx **]** | ue(v) |
| **cpb\_size\_value\_minus1[** SchedSelIdx **]** | ue(v) |
| **cbr\_flag[** SchedSelIdx **]** | u(1) |
| } |  |
| } |  |

# HEVC extension specification syntaxes

## NAL unit header syntax

|  |  |
| --- | --- |
| nal\_unit( NumBytesInNALunit ) { | Descriptor |
| **forbidden\_zero\_bit** | f(1) |
| **nal\_unit\_type** | u(6) |
| **layer\_id** // reserved\_zero\_6bits | 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) |
| } |  |
| } |  |

## Video parameter set RBSP syntax

|  |  |
| --- | --- |
| 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 | 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 | 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( ) |  |
| } |  |

## Video parameter set extension syntax and semantics

### Approach 1

|  |  |
| --- | --- |
| vps\_extension( ) { | Descriptor |
| while( !byte\_aligned( ) ) |  |
| **vps\_extension\_byte\_alignment\_reserved\_zero\_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\_zero\_bit** shall be equal to 0.

**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

|  |  |
| --- | --- |
| vps\_extension( ) { | Descriptor |
| while( !byte\_aligned( ) ) |  |
| **vps\_extension\_byte\_alignment\_reserved\_zero\_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\_zero\_bit** shall be equal to 0.

**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, unspecified |
| 3 | 4 | spatial, quality, unspecified, unspecified |
| 4 | 2 | multiview and depth |
| 5 | 3 | multiview, depth, unspecified |
| 6 | 4 | multiview, depth, unspecified, unspecified |
| 7 | 4 | multiview, spatial, quality and depth |
| 8 | 5 | multiview, spatial, quality, depth, unspecified |
| 9 | 6 | multiview, spatial, quality, depth, unspecified, unspecified |
| 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.

# VPS extension syntaxes and semantics to be discussed

### Approach 1

Additional syntax elements compared to the approach 1 syntax in Section 3 are highlighted.

|  |  |
| --- | --- |
| vps\_extension( ) { | Descriptor |
| while( !byte\_aligned( ) ) |  |
| **vps\_extension\_byte\_alignment\_reserved\_zero\_bit** | u(1) |
| **num\_additional\_layer\_operation\_points** | u(8) |
| **num\_additional\_profile\_level\_sets\_minus1** | u(8) |
| // 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) |
| } |  |
| // operation point specific information |  |
| for( i = 0; i <= num\_additional\_profile\_level\_sets\_minus1; i++ ) |  |
| profile\_level\_info( 1, vps\_max\_sub\_layers\_minus1 ) |  |
| for( i = 1; i <= num\_additional\_layer\_operation\_points; i++ ) |  |
| if( num\_additional\_profile\_level\_sets > 0 ) { |  |
| op\_point( i ) |  |
| **profile\_level\_idx[** i **]** | u(8) |
| } |  |
| } |  |

**num\_additional\_layer\_operation\_points** specifies the maximum number of additional layer operation points present in the codded video sequences the video parameter set applies. [Ed. (YK): Add the definition of "layer operation point", to state that temporal subsets of layer operation points are not considered as layer operation points.]

**num\_additional\_profile\_level\_sets** specifies the number of additional sets of profile and level information present in the video parameter set.

**profile\_level\_index[** i **]** specifies additional set of profile and level information in the video parameter set that is applicable to the i-th operation point.

### Approach 2

Additional syntax elements compared to the approach 2 syntax in Section 3 are highlighted.

|  |  |
| --- | --- |
| vps\_extension( ) { | Descriptor |
| while( !byte\_aligned( ) ) |  |
| **vps\_extension\_byte\_alignment\_reserved\_zero\_bit** | u(1) |
| **num\_additional\_layer\_operation\_points** | u(8) |
| **num\_additional\_profile\_level\_sets\_minus1** | u(8) |
| // 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) |
| } |  |
| // operation point specific information |  |
| for( i = 0; i <= num\_additional\_profile\_level\_sets\_minus1; i++ ) |  |
| profile\_level\_info( 1, vps\_max\_sub\_layers\_minus1 ) |  |
| for( i = 1; i <= num\_additional\_layer\_operation\_points; i++ ) |  |
| if( num\_additional\_profile\_level\_sets > 0 ) { |  |
| op\_point( i ) |  |
| **profile\_level\_idx[** i **]** | u(8) |
| } |  |
| } |  |

**num\_additional\_layer\_operation\_points** specifies the maximum number of additional layer operation points present in the codded video sequences the video parameter set applies. [Ed. (YK): Add the definition of "layer operation point", to state that temporal subsets of layer operation points are not considered as layer operation points.]

**num\_additional\_profile\_level\_sets** specifies the number of additional sets of profile and level information present in the video parameter set.

**profile\_level\_index[** i **]** specifies additional set of profile and level information in the video parameter set that is applicable to the i-th operation point.

# List of input documents

[JCTVC-J0432](http://phenix.it-sudparis.eu/jct/doc_end_user/current_document.php?id=6298) On NAL Unit Header and Video Parameter Set Design [B. Choi, J. Kim, J. Park (Samsung)] [late]

[**JCT2-A0115**](http://phenix.int-evry.fr/jct2/doc_end_user/current_document.php?id=131)**/**[**m26102**](http://phenix.int-evry.fr/mpeg/doc_end_user/current_document.php?id=39263&id_meeting=153) **3D-HLS: On NAL Unit Header and Video Parameter Set Design for HEVC 3DV [B. Choi, J. Kim, J. Park (Samsung)]**

The proposed method belongs to the Approach 2 category.

[JCTVC-J0124](http://phenix.it-sudparis.eu/jct/doc_end_user/current_document.php?id=5987) AHG10: On video parameter set for HEVC extensions [Y. Chen, Y.-K. Wang (Qualcomm)]

[**JCT2-A0099**](http://phenix.int-evry.fr/jct2/doc_end_user/current_document.php?id=91)**/**[**m26054**](http://phenix.int-evry.fr/mpeg/doc_end_user/current_document.php?id=39214&id_meeting=153) **3D-HLS: Video parameter set for 3D-HEVC [**[**Y.Chen**](mailto:cheny@qualcomm.com)**, Y.-K. Wang (Qualcomm)]**

The proposed method belongs to the Approach 1 category.

The proposal proposes to include the following information in the VPS extension:

* Byte alignment before the profile and level information for operation points
* Profile and level information for operation points
* Representation format information for operation points
* Bitrate and frame rate information for operation points
* Operation point dependency information - signal layer dependency information instead
* VUI

[JCTVC-J0257](http://phenix.it-sudparis.eu/jct/doc_end_user/current_document.php?id=6120) AHG9/AHG10: Design of the Video Parameter Set [R. Skupin, V. George, T. Schierl]

[**JCT2-A0121**](http://phenix.int-evry.fr/jct2/doc_end_user/current_document.php?id=178)**/**[**m26214**](http://phenix.int-evry.fr/mpeg/doc_end_user/current_document.php?id=39375&id_meeting=153) **3D-HLS: Design of the Video Parameter Set for 3D-HEVC [Robert Skupin, Valeri George, Thomas Schierl (Fraunhofer HHI)] [late]**

This document proposes one approach 1 method and one approach 2 method.

The following shortcoming was identified for the approach 1 method during the discussion:

* The current design has an extensibility problem: a decoder that does not understand a reserved scalability type would not be able to decode a subset of the bitstream using that reserved scalability type. Fixed in a revision (v5) of the document.
* The part involving vps\_extension\_type is not working.
* This proposal signals profile & level information for each layer, not for each operation point.

[JCTVC-J0075](http://phenix.it-sudparis.eu/jct/doc_end_user/current_document.php?id=5938) AHG10 Hooks for Scalable Coding: Video Parameter Set Design [M. M. Hannuksela (Nokia)]

The proposed method basically belongs to the Approach 1 category (but it claimed to be also applicable to approach 2).

The following shortcoming was identified during the discussion: Parsing and tracking the layer dependency information would be needed to derive all scalability dimension IDs for bitstream adaptation/extraction.

[JCTVC-J0576](http://phenix.int-evry.fr/jct/doc_end_user/current_document.php?id=6456) VPS syntax for scalable and 3D extensions [J. Boyce (Vidyo)]

Discussed at the 3rd BoG meeting.