|  |  |
| --- | --- |
| **Joint Collaborative Team on Video Coding (JCT-VC)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29/WG 11**  27th Meeting: Hobart, AU, 31 March – 7 April 2017 | Document: JCTVC-AA0028 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **AHG7: Signaling of decoded motion constrained tile set hash** | | |
| *Status:* | Input Document to JCT-VC | | |
| *Purpose:* | Proposal | | |
| *Author(s) or Contact(s):* | Jonathan Taquet  Nael Ouedraogo  Franck Denoual  Frederic Maze | Email: | [jonathan.taquet@crf.canon.fr](mailto:jonathan.taquet@crf.canon.fr)  [nael.ouedraogo@crf.canon.fr](mailto:nael.ouedraogo@crf.canon.fr)  [franck.denoual@crf.canon.fr](mailto:franck.denoual@crf.canon.fr)  [frederic.maze@crf.canon.fr](mailto:frederic.maze@crf.canon.fr) |
| *Source:* | Canon Research Centre France | | |

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

# Abstract

This draft document proposes modifications to the on-going Amd3 on HEVC specification. It describes a SEI message for decoded motion-constrained tiles sets (MCTS) hashes. This SEI message provides syntax for signaling the hash of decoded picture data at a motion-constrained tile set’s rectangular region of tiles granularity. The purpose of this proposed SEI message is to facilitate the verification of the correctness of the decoding of motion constrained tile sets. A motion-constrained tile set hash can be used to verify that the motion-constrained tile set is still correctly decoded in any decoding context, potentially different than the encoding one (for example in case of motion-constrained tile sets composition with motion constrained tile sets coming from different bit-streams).

# Problem Statement

The temporal motion constrained tile sets SEI message allows to signal independently decodable tile sets. A decoder may select and decode one or several motion constrained tile sets. In such a case, Decoded Picture Hash SEI message is meaningless since the decoder has not decoded the entire picture.

As per HEVC specification, the inter prediction process is constrained for some motion constrained tile sets such that “no sample value outside each identified tile set, and no sample value at a fractional sample position that is derived using one or more sample values outside the identified tile set, is used for inter prediction of any sample within the identified tile set”. Even if this can be true in the encoding context, there is no guarantee that it will still be true when decoding the motion-constrained tile set in a different context. For instance, if the motion constrained tile set is extracted and decoded without its original neighborhood or with a different one, the motion vector predictors’ list (and/or the merge motion vectors’ list) can be derived differently for coding tree blocks on the right and on the bottom boundaries of a motion constrained tile set. This different derivation can generate decoding mismatches on the pixel values.

One example of tile sets combination is a mix of tile sets corresponding to a region of interest from a first bitstream containing an encoded video at a high quality with other tile sets from one or more other bitstreams containing the same encoded video but at a lower quality level.

The semantic of the temporal motion constrained tile sets SEI message does not guarantee a correct decoding in any situations, but with a suited encoder configuration, it is possible to avoid such kind of decoding errors. To verify that a given context is suited for decoding motion-constrained tile set without error, it is proposed to use hashes of decoded motion-constrained tile set at the rectangular region of tiles granularity.

Moreover, the proposed SEI message defines parameters to signal a cropping area onto which the decoded motion-constrained tile set hash is computed. This allows to check that a motion constrained tile set is properly decoded only within a cropped area of the motion-constrained tile set, for example to not take into account the pixels on the border of the motion-constrained tile set rectangular region of tiles when loop filters are activated at tile borders.

# Proposal

*In Clause D.2.1 and D.3.1 insert the following syntax modifications given in red font:*

D.2.1 General SEI message syntax

|  |  |
| --- | --- |
| sei\_payload( payloadType, payloadSize ) { | Descriptor |
| if( nal\_unit\_type = = PREFIX\_SEI\_NUT ) |  |
| if( payloadType = = 0 ) |  |
| buffering\_period( payloadSize ) |  |
| … |  |
| else if( payloadType = = 153 ) |  |
| mcts\_extraction\_info\_nesting( payloadSize ) |  |
| else if( payloadType = = 154 ) |  |
| decoded\_mcts\_hash ( payloadSize ) |  |
| … |  |
| else if( payloadType = = 160 ) |  |
| layers\_not\_present( payloadSize ) /\* specified in Annex F \*/ |  |
| } |  |
| } |  |

D.3.1 General SEI message semantics

The list SingleLayerSeiList is set to consist of the payloadType values 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 129, 131, 132, and 134 to 154, inclusive.

The list VclAssociatedSeiList is set to consist of the payloadType values 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 131, 132, and 134 to 154, inclusive.

The list PicUnitRepConSeiList is set to consist of the payloadType values 0, 1, 2, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 129, 131, 132, 133, and 135 to 154, inclusive.

*In Table D.1, append the following rows to the end of the table:*

|  |  |
| --- | --- |
| decoded\_mcts\_hash | The access unit containing the SEI message |

*Furthermore, in Clause D.2 and D.3 insert the following additional syntax and semantics:*

D.2.45 Decoded motion-constrained tile set hash SEI message syntax

|  |  |
| --- | --- |
| decoded\_mcts\_hash( payloadSize ) { | **Descriptor** |
| **num\_mcts\_hash\_minus1** | ue(v) |
| for( n = 0; n <= num\_mcts\_hash\_minus1; n++ ) { |  |
| **mcts\_hash\_id**[ n ] | ue(v) |
| **mcts\_hash\_tile\_rect\_idx**[ n ] | ue(v) |
| **crop\_mcts\_hash**[ n ] | u(1) |
| if( crop\_mcts\_hash[ n ] ) { |  |
| **crop\_mcts\_hash\_with\_same\_margins**[ n ] | u(1) |
| if( crop\_mcts\_hash\_with\_same\_margins[ n ] ) |  |
| **crop\_mcts\_hash\_margin**[ n ] | ue(v) |
| else { |  |
| **crop\_mcts\_hash\_top\_margin**[ n ] | ue(v) |
| **crop\_mcts\_hash\_bottom\_margin**[ n ] | ue(v) |
| **crop\_mcts\_hash\_left\_margin**[ n ] | ue(v) |
| **crop\_mcts\_hash\_right\_margin**[ n ] | ue(v) |
| } |  |
| **}** |  |
| **mcts\_hash\_type**[ n ] | u(8) |
| for( cIdx = 0; cIdx < ( chroma\_format\_idc = = 0 ? 1 : 3 ); cIdx++ ) { |  |
| if( mcts\_hash\_type = = 0 ) |  |
| for( i = 0; i < 16; i++) |  |
| **mcts\_md5**[ n ][ cIdx ][ i ] | b(8) |
| else if( mcts\_hash\_type = = 1 ) |  |
| **mcts\_crc**[ n ] [ cIdx ] | u(16) |
| else if( mcts\_hash\_type = = 2 ) |  |
| **mcts\_checksum**[ n ] [ cIdx ] | u(32) |
| } |  |
| } |  |
| } |  |

D.3.45 Decoded motion-constrained tile set hash SEI message semantics

This message provides hashes for one ore more motion-constrained tile set.

NOTE 1 – The decoded motion-constrained tile set hash SEI message is a suffix SEI message and cannot be contained in a scalable nesting SEI message.

Let a set of pictures associatedPicSet be the pictures from the access unit containing the decoded motion-constrained tile set hash SEI message, inclusive, up to but not including the first of any of the following in decoding order:

– The next access unit, in decoding order, that contains a decoded motion-constrained tile set hash SEI message

– The next IRAP picture with NoRaslOutputFlag equal to 1, in decoding order.

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

When a decoded motion-constrained tile set hash SEI message is present for any picture in associatedPicSet, a temporal motion-constrained tile set SEI message shall be present for the first picture of associatedPicSet in decoding order and may also be present for other pictures of associatedPicSet. The mcts\_hash\_id[] values of the decoded motion-constrained tile sets hash SEI message shall have a corresponding equal mcts\_id[] value in the associated temporal motion-constrained tile set SEI message.

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

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

NOTE 2 – This constraint is similar to the constraint associated with tiles\_fixed\_structure\_flag equal to 1, and it may be desirable for tiles\_fixed\_structure\_flag to be equal to 1 when the decoded motion-constrained tile sets hash SEI message is present (although this is not required).

**num\_mcts\_hash\_minus1** indicates the number of motion-constrained tile set rectangular regions of tiles, minus one, for which a hash is provided.

**mcts\_hash\_id**[ n ] provides a value of an existing mcts\_id, defined in a previous temporal motion-constrained tile set SEI message.

**mcts\_hash\_tile\_rect\_idx**[ n ] provides the index of an existing definition of a rectangular region of tiles in the motion-contrained tile set identified by mcts\_hash\_id. This is the rectangular region of tiles on which the hash is computed.

**crop\_mcts\_hash**[ n ] equal to 0 indicates that the hash is computed on the whole rectangular region of tiles. crop\_mcts\_hash[ n ] equal to 1 indicate that the hash is computed in a cropped area of the rectangular region of tiles that is defined with the following parameters.

**crop\_mcts\_hash\_with\_same\_margins**[ n ] equal to 1 indicates that the cropped region on which to compute the hash is obtained by ignoring the same number of pixels on the top, left, bottom and right boundaries of the rectangular region of tiles. This number of pixels is provided by crop\_mcts\_hash\_margin[ n ]. crop\_mcts\_hash\_with\_same\_margins[ n ] equal to 0 indicate that the margins of pixels to ignore may be different between the top, left, bottom and right boundaries. In such case the number of rows/columns of pixels to ignore are provided by crop\_mcts\_hash\_top\_margin[ n ], crop\_mcts\_hash\_bottom\_margin[ n ], crop\_mcts\_hash\_left\_margin[ n ] and crop\_mcts\_hash\_right\_margin[ n ].

**crop\_mcts\_hash\_margin**[ n ] provides the number of pixels rows/columns of the top, left, right and bottom of the rectangular region of tiles to ignore, when crop\_mcts\_hash[ n ] and crop\_mcts\_hash\_with\_same\_margins[ n ] are both equal to 1.

**crop\_mcts\_hash\_top\_margin**[ n ], **crop\_mcts\_hash\_bottom\_margin**[ n ], **crop\_mcts\_hash\_left\_margin**[ n ] and **crop\_mcts\_hash\_right\_margin**[ n ] provide the number of pixel rows/columns of the top, left, right and bottom of the rectangular region of tiles to ignore, when crop\_mcts\_hash[ n ] is equal to 1 and crop\_mcts\_hash\_with\_same\_margins[ n ] is equal to 0.

crop\_mcts\_hash\_margin[ n ], crop\_mcts\_hash\_top\_margin[ n ], crop\_mcts\_hash\_bottom\_margin[ n ], crop\_mcts\_hash\_left\_margin[ n ] and crop\_mcts\_hash\_right\_margin[ n ] are expressed in number of luma pixels. In order to obtain the number of chroma pixels rows/columns to ignore, these values are divided by SubHeightC for the rows and by SubWidthC for the columns.

Prior to computing the hash, region size and cropping values are initialized as follow:

for( n = 0; n <= num\_mcts\_hash\_minus1; n++ ) {  
 if ( crop\_mcts\_hash[ n ] == 0 )  
 crop\_mcts\_hash\_margin[ n ] = 0  
 if ( crop\_mcts\_hash[ n ] == 0 || crop\_mcts\_hash\_with\_same\_margins[ n ] == 1 ) {  
 crop\_mcts\_hash\_top\_margin[ n ] = crop\_mcts\_hash\_margin[ n ]  
 crop\_mcts\_hash\_bottom\_margin[ n ] = crop\_mcts\_hash\_margin[ n ]  
 crop\_mcts\_hash\_left\_margin[ n ] = crop\_mcts\_hash\_margin[ n ]  
 crop\_mcts\_hash\_right\_margin[ n ] = crop\_mcts\_hash\_margin[ n ]  
 }  
 top\_left\_tile\_row[ n ] = top\_left\_tile\_index[ n ] / (num\_tile\_columns\_minus1 + 1)  
 top\_left\_tile\_column[ n ] =  
 top\_left\_tile\_index[ n ] - top\_left\_tile\_row[ n ] \* (num\_tile\_columns\_minus1 + 1)  
 bottom\_right\_tile\_row[ n ] = bottom\_right\_tile\_index[ n ] / (num\_tile\_columns\_minus1 + 1)  
 bottom\_right\_tile\_column[ n ] =  
 bottom\_right\_tile\_index[ n ] - bottom\_right\_tile\_row[ n ] \* (num\_tile\_columns\_minus1 + 1) (D‑XX)  
 mcts\_rect\_height\_in\_luma\_samples[ n ] = 0  
 mcts\_rect\_width\_in\_luma\_samples[ n ] = 0  
 for ( i = top\_left\_tile\_row[ n ]; i <= bottom\_right\_tile\_row[ n ]; i++ )  
 mcts\_rect\_height\_in\_luma\_samples[ n ] += RowHeightInLumaSamples[ i ]  
 for ( i = top\_left\_tile\_column[ n ]; i <= bottom\_right\_tile\_column[ n ]; i++ )  
 mcts\_rect\_width\_in\_luma\_samples[ n ] += ColumnWidthInLumaSamples[ i ]  
}

Where top\_left\_tile\_index[ n ] and bottom\_right\_tile\_index[ n ] correspond to the values top\_left\_tile\_index[ i ][ mcts\_hash\_tile\_rect\_idx[ n ] ] and top\_right\_tile\_index[ i ][ mcts\_hash\_tile\_rect\_idx[ n ] ] as defined in the temporal motion constrained tile SEI message that has its mcts\_id[ i ] equal to mcts\_hash\_id[ n ].

Prior to computing the hash of the ‘n’-th motion-constrained tile set rectangular region of tiles, the decoded picture data of that motion-constrained tile set rectangular region of tiles is arranged into one or three strings of bytes called mctsData[ n ][ cIdx ] of lengths dataLen[ n ][ cIdx ] as follows:

for( n = 0; n <= num\_mcts\_hash\_minus1; n++ )  
 for( cIdx = 0; cIdx < ( chroma\_format\_idc = = 0 ) ? 1 : 3; cIdx++ ) {  
 if( cIdx = = 0 ) {  
 compWidth[ n ][ cIdx ] = mcts\_rect\_width\_in\_luma\_samples[ n ]  
 compHeight[ n ][ cIdx ] = mcts\_rect\_height\_in\_luma\_samples[ n ]  
 compDepth[ cIdx ] = BitDepthY  
 cropTop[ n ][ cIdx ] = crop\_mcts\_hash\_top\_margin[ n ]  
 cropBottom[ n ][ cIdx ] = crop\_mcts\_hash\_bottom\_margin[ n ]  
 cropLeft[ n ][ cIdx ] = crop\_mcts\_hash\_left\_margin[ n ]  
 cropRight[ n ][ cIdx ] = crop\_mcts\_hash\_right\_margin[ n ]  
 } else {  
 compWidth[ n ][ cIdx ] = mcts\_rect\_width\_in\_luma\_samples[ n ] / SubWidthC  
 compHeight[ n ][ cIdx ] = mcts\_rect\_height\_in\_luma\_samples[ n ] / SubHeightC  
 compDepth[ cIdx ] = BitDepthC (D‑XX)  
 cropTop[ n ][ cIdx ] = crop\_mcts\_hash\_top\_margin[ n ] / SubHeightC  
 cropBottom[ n ][ cIdx ] = crop\_mcts\_hash\_bottom\_margin[ n ] / SubHeightC  
 cropLeft[ n ][ cIdx ] = crop\_mcts\_hash\_left\_margin[ n ] / SubWidthC  
 cropRight[ n ][ cIdx ] = crop\_mcts\_hash\_right\_margin[ n ] / SubWidthC  
 }  
 iLen = 0  
 for( i = cropTop[ n ][ cIdx ]; i < compHeight[ n ][ cIdx ] - cropBottom[ n ][ cIdx ]; i++ )  
 for( j = cropLeft[ n ][ cIdx ]; j < compWidth[ n ][ cIdx ] - cropRight[ n ][ cIdx ]; j++ ) {  
 mctsData[ n ][ cIdx ][ iLen++ ] =  
 mcts\_component[ n ][ cIdx ][ i \* compWidth[ n ][ cIdx ] + j ] & 0xFF  
 if( compDepth[ cIdx ] > 8 )  
 mctsData[ n ][ cIdx ][ iLen++ ] =  
 mcts\_component[ n ][ cIdx ][ i \* compWidth[ n ][ cIdx ] + j ] >> 8  
 }  
 dataLen[ n ][ cIdx ] = iLen  
}

where mcts\_component[ n ][ cIdx ][ i ] is an array in raster scan of decoded sample values of the motion-constrained tile set rectangular region of tiles in two's complement representation.

**mcts\_hash\_type**[ n ] indicates the method used to calculate the checksum according to Table D.XX. Values of mcts\_hash\_type that are not listed in Table D.XX are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore decoded motion-constrained tile set hash SEI messages that contain reserved values of mcts\_hash\_type.

**Table D.XX – Interpretation of mcts\_hash\_type**

|  |  |
| --- | --- |
| **mcts\_hash\_type** | **Method** |
| 0 | MD5 (RFC 1321) |
| 1 | CRC |
| 2 | Checksum |

**mcts\_md5**[ n ][ cIdx ][ i ] is the 16-byte MD5 hash of the cIdx-th colour component of the decoded motion-constrained tile set rectangular region of tiles. The value of mcts\_md5[ n ][ cIdx ][ i ] shall be equal to the value of digestVal[ n ][ cIdx ] obtained as follows, using the MD5 functions defined in RFC 1321:

MD5Init( context )  
MD5Update( context, mctsData[ n ][ cIdx ], dataLen[ n ][ cIdx ] ) (D‑XX)  
MD5Final( digestVal[ n ][ cIdx ], context )

**mcts\_crc**[ n ][ cIdx ] is the cyclic redundancy check (CRC) of the colour component cIdx of the decoded motion-constrained tile set rectangular region of tiles. The value of mcts\_crc[ n ][ cIdx ] shall be equal to the value of crcVal[ n ][ cIdx ] obtained as follows:

crc = 0xFFFF  
mctsData[ n ][ cIdx ][  dataLen[ n ][ cIdx ] ] = 0  
mctsData[ n ][ cIdx ][  dataLen[ n ][ cIdx ] + 1 ] = 0  
for( bitIdx = 0; bitIdx < ( dataLen[ n ][ cIdx ]  + 2 ) \* 8; bitIdx++ ) {  
 dataByte = mctsData[ n ][ cIdx ][ bitIdx >> 3 ]  
 crcMsb = ( crc >> 15 ) & 1  
 bitVal = ( dataByte >> ( 7 − ( bitIdx & 7 ) ) ) & 1  
 crc = ( ( ( crc << 1 ) + bitVal ) & 0xFFFF ) ^ ( crcMsb \* 0x1021 )  
}  
crcVal[ cIdx ] = crc (D‑XX)

NOTE 2 – The same CRC specification is found in Rec. ITU-T H.271.

**mcts\_checksum**[ n ][ cIdx ] is the checksum of the colour component cIdx of the decoded motion constrained tile set rectangular region of tiles. The value of mcts\_checksum[ n ][ cIdx ] shall be equal to the value of checksumVal[ n ][ cIdx ] obtained as follows:

sum = 0  
for( i = cropTop[ n ][ cIdx ]; i < compHeight[ n ][ cIdx ] - cropBottom[ n ][ cIdx ]; i++ ) {  
 y = i - cropTop[ n ][ cIdx ]  
 for( j = cropLeft[ n ][ cIdx ]; j < compWidth[ n ][ cIdx ] – cropRight[ n ][ cIdx ]; j++ ) {  
 x = j - cropLeft[ n ][ cIdx ]  
 xorMask = ( x & 0xFF ) ^ ( y & 0xFF ) ^ ( x >> 8 ) ^ ( y >> 8 )  
 sum = ( sum + ( ( component[ n ][ cIdx ][ i \* compWidth[ n ][ cIdx ] + j ] & 0xFF ) ^ xorMask )  
 ) & 0xFFFFFFFF  
 if( compDepth[ cIdx ] > 8 )  
 sum = ( sum + ( ( component[ n ][ cIdx ][ i \* compWidth[ n ][ cIdx ] + j ] >> 8 ) ^ xorMask )  
 ) & 0xFFFFFFFF  
 }  
checksumVal[ n ][ cIdx ] = sum (D‑XX)

# Patent rights declaration(s)

**Canon Research Centre France may have current or pending patent rights relating to the technology described in this contribution and, conditioned on reciprocity, is prepared to grant licenses under reasonable and non-discriminatory terms as necessary for implementation of the resulting ITU-T Recommendation | ISO/IEC International Standard (per box 2 of the ITU-T/ITU-R/ISO/IEC patent statement and licensing declaration form).**