|  |
| --- |
| **INTERNATIONAL ORGANIZATION FOR STANDARDIZATION ORGANISATION INTERNATIONALE DE NORMALISATION ISO/IEC JTC 1/SC 29/WG 5 MPEG JOINT VIDEO EXPERTS TEAM WITH ITU-T SG 16** |
| **ISO/IEC JTC 1 / SC 29 / WG 5 N 305** |
| **Sapporo, JP – 12–19 July 2024** |
| |  |  | | --- | --- | | **Source:** | **Convenor (Jens-Rainer Ohm)** | | **Title:** | **Technologies under consideration for future extensions of VSEI (version 5)** | | **Type:** | **General** | | **Subtype:** | **Other** | | **Status:** | **Approved** | | **Date:** | **2024-08-16** | | **Expected Action:** | **Info** | | **Action due date:** | **N/A** | | **Pages:** | **94 (not including this cover page)** | | **Email of convenor:** | **ohm @ ient . rwth-aachen . de** | | **Committee URL:** | **https://sd.iso.org/documents/ui/#!/browse/iso/iso-iec-jtc-1/iso-iec-jtc-1-sc-29/iso-iec-jtc-1-sc-29-wg-5** | |

|  |  |
| --- | --- |
| **Joint Video Experts Team (JVET)**  **of ITU-T SG 16 WP 3 and ISO/IEC JTC 1/SC 29**  35th Meeting, Sapporo, Japan, 12–19 July 2024 | Document: JVET-AI2032-v3 |

|  |  |  |  |
| --- | --- | --- | --- |
| *Title:* | **Technologies under consideration for future extensions of VSEI (version 5)** | | |
| *Status:* | Output document approved by JVET | | |
| *Purpose:* | Draft text | | |
| *Author(s) or Contact(s):* | Sean McCarthy Jill Boyce Jie Chen Sachin Deshpande Miska M. Hannuksela Hendry Tan Ye-Kui Wang | Email: | sean.mccarthy@dolby.com  jill.boyce@nokia.com jiechen.cj@alibaba-inc.com [sdeshpande@sharplabs.com](mailto:sdeshpande@sharplabs.com) miska.hannuksela@nokia.com dr.hendry@lge.com yekui.wang@bytedance.com |
| *Source:* | Editors | | |

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

# Abstract

This document contains draft text for changes under consideration for future extensions to the versatile supplemental enhancement information messages for coded video bitstreams (VSEI) standard (Rec. ITU-T H.274 | ISO/IEC 23002-7) to modify existing SEI messages, specify additional SEI messages, or specify other functionalities. This document also contains draft text for changes under consideration for future extensions to the VVC, HEVC, and AVC standards for specifying the use of SEI messages specified in VSEI in VVC, HEVC, and AVC bitstreams, respectively.

The base text for this version of the document is JVET-AH2032.

# Changes integrated for the 35th JVET meeting:

* AI0156 Generative face video SEI bug fixes and editorial updates
* AI0158 Lens optical correction SEI message
* AI0178 Constituent Rectangles SEI minor updates
* AI0181 Display overlays SEI
* AI0182 Bitdepth range info SEI
* AI0184 Generative face video SEI value ranges for ue(v)-coded syntax elements and editorial updates
* AI0189 Generative face video SEI updates on signalling and derivation of translator NN
* AI0190 Generative face video SEI updates on persistence and disallowing pictures from being both a base picture and a fusion picture
* AI0191 Generative face video SEI updates on GFV facial parameters signalling
* AI0192 Generative face video SEI on output order of GFV output pictures
* AI0193 Generative face video SEI editorial suggestions
* AI0194 On chroma key fusion for the generative face video SEI message
* AI0195 Change coding of key point coordinates and decimal part of matrix elements from u(v) to ue(v), and change the key point coordinates that are input to the generative model from the actual coordinates to be the normalized coordinates.
* AI0127:Digitally Signed Content Authentication SEI
* AI0340 AI usage restrictions SEI
* AI0205 Combined image format information SEI to replace separate SEI messages
* AI0206 International Color Consortium (ICC) profiles added to combined image format info SEI

# Changes integrated for the 34th JVET meeting:

Changes noted below correspond to aspects of JVET-AH0049, JVET-AH0161, JVET-AH0162, JVET-AH0164, JVET-AH0325, and JVET-AH0350.

* Add interface text for AVC and HEVC to support modality information SEI as proposed in JVET-AH0049
* Add packed regions SEI as proposed in JVET-AH0161
* Add constituent rectangles SEI as proposed in JVET-AH0162
* Add quality metric SEI as proposed in JVET-AH0164
* Remove post-processing filter gain SEI message (features include in quality metric SEI)
* Remove Copyright SEI and AI marking SEI (features included in text description SEI in JVET-AH2006)
* Add alternative persistence signalling as proposed in JVET-AH0325
* Add modification to generative face video SEI in JVET-AH0053
* Document both the breadth-first and depth-first SEI processing order approach of JVET-AH0350 as TuC text specification to determine whether depth-first has some advantage.
* Semantics fixes for GFV SEI message form JVET-AH0349
* Add generative face video enhancement SEI message from JVET-AH0127
* Added gfv\_low\_confidence\_face\_parameter\_flag to generative face video enhancement SEI message from JVET-AH0054

Summary of JVET-AG0232: changes noted below corresponding to aspects of JVET-AG0044, JVET-AG0045, JVET-AG0051, JVET-AG0086, JVET-AG0087, JVET-AG0203, and JVET-AG0328

* Remove encoder optimization information SEI message (move to JVET-AG2034)
* Add the capability for indicating temporal sublayers where optimization has been applied into the the encoder optimization information SEI message (JVET-AG0086, item 2 option A).
* Remove source picture timing information SEI message (move to JVET-AG2034)
* Remove object mask information SEI message (move to JVET-AG2034)
* Remove modality information in VUI (move to JVET-AG2034 as modality information SEI message (JVET-AG0322))
* Remove NNPFC SEI message application information signaling (move to JVET-AG2034)
* Include copyright SEI message as proposed in JVET-AG0044 proposal 1
* Include AI marking SEI message as proposed in JVET-AG0045 proposal 1
* Include generative face video SEI message
  + as proposed in JVET-AG0203
    - Aspect 1 (gating flag for signalling the translator) (also agreed in context of JVET-AG0087)
    - Aspect 2 (predictive signalling of matrix elements)
    - Aspect 3 (more flexible interface between translator and model in terms of numbers of keypoints and matrix elements)
  + with modifications agreed from JVET-AG0087 as follows.
    - enable the capability of sending parameter translator description only at an IRAP
    - allow multiple GFV SEI messages in a picture unit, e.g., carried within suffix SEI NAL units. Additional ingredients of this item include the following:
      * Each new instance of a GFV SEI message will invoke the generative neural network (NN) inference.
      * To differentiate between repetitions and new instances of GFV SEI messages, add a counter syntax element (gfv\_cnt) in the syntax.
* Include film grain regions characteristics SEI message as proposed in JVET-AG0328
* Include versatile SEI RBSP and versatile SEI message as proposed in JVET-AG0051
  + move to a separate section together with LSEI
  + Proponents of both approaches are asked to work out possible combination

# Changes to the specification text:

*Additions to VSEI subclause 2.3 for support of quality metrics SEI, image format metadata SEI*

– ISO/IEC 23001-10 Carriage of timed metadata metrics of media in ISO base media file format

– ISO/IEC 23001-11 Energy-Efficient Media Consumption (Green Metadata)

* T. 871 : Information technology - Digital compression and coding of continuous-tone still images: JPEG File Interchange Format (JFIF)
* ISO/IEC 10918 – 5 Information technology — Digital compression and coding of continuoustone still images: JPEG File Interchange Format (JFIF)
* Exchangeable image file format for digital still cameras: Exif Version 3.0, Camera & Imaging Products Association, May 2023. [Ed Note: awaiting conclusion of ITU A.5 and A.4 processes]
* ISO 16684-1: Graphic Technology – Extensible metadata platform (XMP) specification– Part 1 Data model, serialization and core properties.
* ISO 16684-3: Graphic Technology – Extensible metadata platform (XMP) specification – Part 3 JSON-LD serialization of XMP.
* ISO 15076-1 Image technology colour management — Architecture, profile format and data structure — Part 1: Based on ICC.1:2010
* ISO DIS 15076-2 Image technology colour management — Architecture, profile format and data structure — Part 2: Based on ICC.1:2022
* ISO 20677 Image technology colour management — Extensions to architecture, profile format and data structure

*Add to clause 3 the following:*

**3. X subpicture:** A rectangular region of one or more *slices* within a *picture*.

*Add to clause 4 the following:*

ICC International Color Consortium

*Changes to VSEI subclause 8.28 for adding the support of grouping of NNPFs*

8.28 Neural-network post-filter SEI messages

**8.28.1 General** **post-processing filtering process using NNPFs**

**8.28.1.1 General**

Input to this process is a bitstream BitstreamToFilter. Output of this process is a list of NNPF output pictures ListNnpfOutputPics.

First, BitstreamToFilter is decoded, and the list CroppedDecodedPictures is set to be the list of the cropped decoded pictures in output order resulted from decoding BitstreamToFilter.

Second, NnpfCand is set to contain any single NNPF or any single NNPF group. When NnpfCand contains an NNPF group with nnpfgc\_grouping\_type equal to 3, the subseequnt specifications of this subclause apply when NnpfCand is set to contain individually each member NNPF, if any, and each member NNPF group, if any, of the NNPF group with nnpfgc\_grouping\_type equal to 3.

Third, the filtering process for one picture, as specified in subclause 8.28.1.2, is repeatedly invoked, in output order, for each cropped decoded picture that is in CroppedDecodedPictures and for which the single NNPF contained in NnpfCand or the single NNPF group contained in NnpfCand, or one or more NNPFs or NNPF groups defined as alternatives or alternating in the NNPF group contained in NnpfCand are activated.

The order of the pictures in ListNnpfOutputPics is in output order.

Within ListNnpfOutputPics there shall be no more than one picture pertaining to any particular output time instance. [Ed. Check phrasing of this. “Pertains” is not used in a similar way anywhere in the standard.] When for any particular picture in CroppedDecodedPictures there are multiple NNPFs activated and only one of the NNPFs is allowed to be chosen to be applied although any of the NNPFs may be chosen, the above constraint shall apply regardless of which NNPF is chosen to be applied to the particular picture.

BitstreamToFilter may be processed multiple times to generate multiple different ListNnpfOutputPics through the second and third steps above.

**8.28.1.2 Filtering process for one picture**

The filtering process specified in this subclause applies to each cropped decoded picture, referred to as the current picture, that is in CroppedDecodedPictures and for which one or more NNPFs or NNPF groups in NnpfCand are activated.

An NNPF or an NNPF group to be applied to the current picture is selected as follows:

– If NnpfCand contains a single NNPF and that NNPF is activated for the current picture according to an NNPFA SEI message, that NNPF is selected to be applied to the current picture.

– Otherwise, if NnpfCand contains an NNPF group with nnpfgc\_grouping\_type equal to 2 and any NNPF of the NNPF group is activated for the current picture according to NNPFA SEI message, that NNPF is selected to be applied to the current picture.

– Otherwise, if NnpfCand contains an NNPF group with nnpfgc\_grouping\_type equal to 0 and that NNPF group is activated for the current picture according to an NNPFGA SEI message, that NNPF group is selected to be applied to the current picture.

– Otherwise (NnpfCand contains an NNPF group with nnpfgc\_grouping\_type equal to 1), the following applies:

– A set of candidate NNPFs or NNPF groups candSet is initially empty and then set to contain the following:

– The NNPFs that are activated for the current picture according to NNPFA SEI messages and are included in the NNPF group contained in NnpfCand.

– The NNPF groups that are activated for the current picture according to NNPFGA SEI messages and are included in the NNPF group contained in NnpfCand.

– For each candidate NNPF or NNPF group candFilter in candSet, the following applies:

– When one or more of the input pictures of candFilter are input pictures to the NNPF or NNPF group prevFilter that was used in any previous invocation of the filtering process specified in this subclause for the same NnpfCand, candFilter is excluded from candSet.

– Any NNPF or NNPF group remaining in candSet is selected to be applied to the current picture.

When applying an NNPF to the current picture, the following applies:

– The filtered and/or interpolated pictures are generated by the NNPF by applying the NNPF process specified in the semantics of the NNPFC SEI message, in a patch-wise manner, to the current picture.

– The order of the pictures generated by the NNPF by applying the NNPF process being stored into the output tensor of the NNPF is in output order.

– The pictures generated by the NNPF and output by the NNPF process are included into ListNnpfOutputPics, in the same order as when the pictures are stored into the output tensor of the NNPF.

When applying an NNPF group to the current picture, the following applies:

– The filtered and/or interpolated pictures are generated by applying the NNPF process specified in the semantics of the NNPFC SEI message, in a patch-wise manner, as specified in the semantics of the NNPFGA SEI message activating the NNPF group.

– The pictures in FinalNnpfgaOutputPicList are included into ListNnpfOutputPics, in the same order as the pictures are stored in FinalNnpfgaOutputPicList.

* + 1. **Neural-network post-filter group characteristics SEI message**
       1. **Neural-network post-filter group characteristics SEI message syntax**

|  |  |
| --- | --- |
| nn\_post\_filter\_group\_characteristics( payloadSize ) { | **Descriptor** |
| **nnpfgc\_id** | ue(v) |
| **nnpfgc\_grouping\_type** | ue(v) |
| if( nnpfgc\_grouping\_type = = 0 | | nnpfgc\_grouping\_type = = 2 ) |  |
| **nnpfgc\_purpose** | u(16) |
| **nnpfgc\_num\_members\_minus2** | ue(v) |
| for( i = 0; i <= nnpfgc\_num\_members\_minus2 + 1; i++ ) { |  |
| **nnpfgc\_member\_id**[ i ] | ue(v) |
| if( nnpfgc\_grouping\_type = = 1 | | nnpfgc\_grouping\_type = = 3 ) |  |
| **nnpfgc\_member\_purpose**[ i ] | u(16) |
| } |  |
| **nnpfgc\_complexity\_info\_present\_flag** | u(1) |
| if( nnpfgc\_complexity\_info\_present\_flag ) { |  |
| **nnpfgc\_parameter\_type\_idc** | u(2) |
| if( nnpfgc\_parameter\_type\_idc != 2 ) |  |
| **nnpfgc\_log2\_parameter\_bit\_length\_minus3** | u(2) |
| **nnpfgc\_num\_parameters\_idc** | u(6) |
| **nnpfgc\_num\_kmac\_operations\_idc** | ue(v) |
| **nnpfgc\_total\_kilobyte\_size** | ue(v) |
| } |  |
| } |  |

* + - 1. **Neural-network post-filter group characteristics SEI message semantics**

The neural-network post-filter group characteristics (NNPFGC) SEI message specifies a neural network post-filter (NNPF) group. It is indicated by the SEI message if the NNPF group defines an NNPF cascade or defines NNPFs or NNPF groups of NNPF cascades that are alternatives to each other. The use of NNPF groups of NNPF cascades for specific pictures is indicated with neural-network post-filter group activation (NNPFGA) SEI messages. An NNPF group may have members that are NNPF or NNPF group. An NNPF is a direct member of an NNPF group when it is included in the list of members in the NNPFGC SEI message of the NNPF group. An NNPF is an indirect member of an NNPF group nnpfg\_1 when it is a member of another NNPF group nnpfg\_2 and nnpfg\_2 is a member of nnpfg\_1.

**nnpfgc\_id** contains an identifying number that may be used to identify an NNPF group. The value of nnpfgc\_id shall be in the range of 0 to 232 − 2, inclusive. Values of nnpfgc\_id from 256 to 511, inclusive, and from 231 to 232 − 2, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders conforming to this edition of this document encountering an NNPFGC SEI message with nnpfgc\_id in the range of 256 to 511, inclusive, or in the range of 231 to 232 − 2, inclusive, shall ignore the SEI message. The value of nnpfgc\_id shall not be equal to any nnpfc\_id value of any NNPFC SEI message present in the same CLVS. When the value of nnpfgc\_id of an NNPFGC SEI message nnpfgcSeiA is equal to the value of nnpfgc\_id of another NNPFGC SEI message nnpfgcSeiB present in the same CLVS, nnpfgcSeiA and nnpfgcSeiB shall be identical.

**nnpfgc\_grouping\_type** equal to 0 indicates that this SEI message specifies a group of cascaded neural-network post-filters.

nnpfgc\_grouping\_type equal to 1 indicates that the NNPFs or NNPF groups identified by the nnpfgc\_member\_id[ i ] are alternatives to each other out of which the post-processor should select only one to be applied.

nnpfgc\_grouping\_type equal to 2 indicates that this SEI message specifies a group of NNPFs that are intended to be used jointly and are activated in an alternating manner so that at most one of these NNPFs is activate for any picture.

nnpfgc\_grouping\_type equal to 3 indicates that the NNPFs or NNPF groups identified by the nnpfgc\_member\_id[ i ] are intended to be used in parallel.

nnpfgc\_grouping\_type equal to 4 indicates that the NNPFs or NNPF groups identified by the nnpfgc\_member\_id[ i ] are optional, i.e., may or may not be applied by the post-processor.

The value of nnpfgc\_grouping\_type shall be in the range of 0 to 255, inclusive. Values of nnpfgc\_grouping\_type in the range of 5 to 255, inclusive, are reserved for future specification by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this edition of this document. Decoders conforming to this edition of this document shall ignore NNPFGC SEI messages with nnpfgc\_grouping\_type in the range of 5 to 255, inclusive.

**nnpfgc\_purpose** has the semantics of nnpfc\_purpose but with the exception that the semantics are specified for the NNPF group defined by this SEI message rather than the NNPF defined by an NNPFC SEI message. When present, the value of nnpfgc\_purpose shall be the union of the values of nnpfc\_purpose of direct and indirect members of the NNPF group.

**nnpfgc\_num\_members\_minus2** plus 2indicates the number of NNPFs or NNPF groups in the NNPF group that this SEI message defines.

**nnpfgc\_member\_id**[ i ] indicates the i-th member in the NNPF group defined by this SEI message as follows:

– If there is an NNPF with nnpfc\_id equal to nnpfgc\_member\_id[ i ] defined in the CLVS, the i-th member in the NNPF group defined by this SEI message is an NNPF that has nnpfc\_id equal to nnpfgc\_member\_id[ i ].

– Otherwise (there is no NNPF with nnpfc\_id equal to nnpfgc\_member\_id[ i ] defined in the CLVS), the i-th member in the NNPF group defined by this SEI message is an NNPF group with nnpfgc\_id equal to nnpfgc\_member\_id[ i ].

When an nnpfgc\_member\_id[ i ] value references an nnpfgc\_id value of an NNPFGC SEI message nnpfgcSei, it is a requirement of bitstream conformance that the NNPFGC SEI message nnpfgcSei shall have nnpfgc\_grouping\_type equal to 0.

NOTE 1 – In other words, when a second NNPF group is a member of a first NNPF group, the type of the second NNPF group cannot be any other than a cascade of NNPFs.

When nnpfgc\_grouping\_type is equal to 0 or 2, it is a requirement of bitstream conformance that there is an NNPF with nnpfc\_id value equal to nnpfgc\_member\_id[ i ] defined in the CLVS.

NOTE 2 – In other words, the members of a cascade or alternating NNPF group are individual NNPFs and cannot be NNPF groups.

When nnpfgc\_grouping\_type is equal to 1, 3, or 4, it is a requirement of bitstream conformance that there is an NNPF with nnpfc\_id value equal to nnpfgc\_member\_id[ i ] or an NNPF group with nnpfgc\_id value equal to nnpfgc\_member\_id[ i ] defined in the CLVS.

When nnpfgc\_grouping\_type is equal to 0, the NNPFs with nnpfc\_id equal to nnpfgc\_member\_id[ i ] are performed in cascade in increasing order of i, as activated by an NNPFGA SEI message with nnpfga\_target\_id equal to nnpfgc\_id.

**nnpfgc\_member\_purpose**[ i ] has the semantics of nnpfc\_purpose but with the exception that the semantics are specified for the NNPFC or NNPFGC associated with nnpfgc\_member[ i ]. When present, the value of nnpfgc\_member\_purpose[ i ] shall be as follows:

– If nnpfgc\_member\_id[ i ] is equal to the value of nnpfc\_id of an NNPF, the value of nnpfgc\_member\_purpose[ i ] shall be equal to the value of nnpfc\_purpose of the NNPF.

– Otherwise (nnpfgc\_member\_id[ i ] is equal to the value of nnpfgc\_id of an NNPF group ), the value of nnpfgc\_member\_purpose[ i ] shall be equal to nnpfgc\_purpose of the associatied NNPF group.

nnpfgc\_complexity\_info\_present\_flag, nnpfgc\_parameter\_type\_idc, nnpfgc\_log2\_parameter\_bit\_length\_minus3, nnpfgc\_num\_parameters\_idc, nnpfgc\_num\_kmac\_operations\_idc, and nnpfgc\_total\_kilobyte\_size have the semantics of nnpfc\_complexity\_info\_present\_flag, nnpfc\_parameter\_type\_idc, nnpfc\_log2\_parameter\_bit\_length\_minus3, nnpfc\_num\_parameters\_idc, nnpfc\_num\_kmac\_operations\_idc, and nnpfc\_total\_kilobyte\_size, respectively, but with the exception that the semantics are specified for the NNPF group defined by this SEI message rather than the NNPF defined by an NNPFC SEI message. When nnpfgc\_grouping\_type is equal to 1, nnpfgc\_complexity\_info\_present\_flag shall be equal to 0.

* + 1. **Neural-network post-filter group activation SEI message**
       1. **Neural-network post-filter group activation SEI message syntax**

|  |  |
| --- | --- |
| nn\_post\_filter\_group\_activation( payloadSize ) { | **Descriptor** |
| **nnpfga\_target\_id** | ue(v) |
| **nnpfga\_cancel\_flag** | u(1) |
| if( !nnpfga\_cancel\_flag ) { |  |
| **nnpfga\_persistence\_flag** | u(1) |
| **nnpfga\_no\_prev\_clvs\_flag** | u(1) |
| if( nnpfga\_persistence\_flag ) |  |
| **nnpfga\_no\_foll\_clvs\_flag** | u(1) |
| **nnpfga\_num\_filters\_minus2** | ue(v) |
| for( i = 0; i <= nnpfga\_num\_filters\_minus2 + 1; i++ ) { |  |
| **nnpfga\_target\_base\_flag**[ i ] | u(1) |
| **nnpfga\_input\_all\_pics\_flag**[ i ] | u(1) |
| if( !nnpfga\_input\_all\_pics\_flag[ i ] ) { |  |
| **nnpfga\_num\_input\_pics\_minus1**[ i ] | ue(v) |
| for( j = 0; j <= nnpfga\_num\_input\_pics\_minus1[ i ]; j++ ) |  |
| **nnpfga\_input\_pic\_skip\_count**[ i ][ j ] | ue(v) |
| } |  |
| **nnpfga\_num\_output\_entries**[ i ] | ue(v) |
| for( j = 0; j < nnpfga\_num\_output\_entries[ i ]; j++ ) |  |
| **nnpfga\_output\_flag**[ i ][ j ] | u(1) |
| } |  |
| **nnpfga\_num\_output\_pic\_update** | ue(v) |
| for( i = 0; i < nnpfga\_num\_output\_pic\_update; i++ ) |  |
| **nnpfga\_output\_pic\_update\_flag**[ i ] | u(1) |
| } |  |
| } |  |

* + - 1. **Neural-network post-filter group activation SEI message semantics**

The neural-network post-filter group activation (NNPFGA) SEI message activates or de-activates the possible use of the target neural-network post-processing filter group (NNPFG) of NNPF groups, identified by nnpfga\_target\_id, for post-processing filtering of a set of pictures.

nnpfgc\_grouping\_type for the identfied NNPF group, which has nnpfgc\_id equal to nnpfga\_target\_id, shall be equal to 0 (cascade) or 1 (alternatives). When nnpfgc\_grouping\_type of the identified NNPF group is equal to 1, each member of the group shall have the same number of input pictures and NNPF output pictures.

For a particular picture for which the NNPFG is activated, the associated NNPFGC SEI message is the last NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id, that precedes the first VCL NAL unit of the current picture in decoding order. If the nnpfgc\_grouping\_type in the associated NNPGC SEI message is equal to 0, the target NNPFG is the NNPFG specified by the associated NNPFGC SEI message. Otherwise (nnpfgc\_grouping\_type in the associated NNPFGC SEI message is equal to 1), the target NNPFG is any NNPFG, if any, that is a member of the NNPFG specified by the associated NNPFGC SEI message. The NNPFs of the target NNPFG are defined by the NNPFC SEI messages that have nnpfc\_id equal to any nnpfgc\_member\_id[ i ] value of the target NNPFG and are present in the current picture unit or precede the current picture in decoding order.

NOTE 1 – The members of the target NNPFG are individual NNPFs and cannot be NNPF groups.

Use of this SEI message requires the definition of the following variables:

– Input picture width and height in units of luma samples, denoted herein by InitCroppedWidth[ idx ] and InitCroppedHeight[ idx ], respectively, of the candidate input pictures with index idx in the range of 0 to numCandInputPics − 1, inclusive, that may be used as input for the NNPFG.

– Luma sample array InitCroppedYPic[ idx ] and chroma sample arrays InitCroppedCbPic[ idx ] and InitCroppedCrPic[ idx ], when present, of the candidate input pictures with index idx in the range of 0 to numCandInputPics − 1, inclusive, that may be used as input for the NNPFG.

– Bit depth BitDepthY for the luma sample array of the candidate input pictures.

– Bit depth BitDepthC for the chroma sample arrays, if any, of the candidate input pictures.

– A chroma format indicator, denoted herein by ChromaFormatIdc, as described in subclause ‎7.3.

– When nnpfc\_auxiliary\_inp\_idc is equal to 1, a filtering strength control value array StrengthControlVal[ idx ] that shall contain real numbers in the range of 0 to 1, inclusive, of the candidate input pictures with index idx in the range of 0 to numCandInputPics − 1, inclusive.

Candidate input picture with index 0 corresponds to the picture for which the NNPFG is activated by this NNPFGA SEI message. Candidate input picture with index i in the range of 1 to numCandInputPics − 1, inclusive, precedes the candidate input picture with index i − 1 in output order. Let candInputPicList[ 0 ] be the list of candidate input pictures in inverse output order.

**nnpfga\_target\_id** indicates the target NNPFG, which is specified by the NNPFGC SEI message that pertains to the current picture and have nnpfgc\_id equal to nnpfga\_target\_id. [Ed. Check phrasing of this. “Pertains” is not used in a similar way anywhere in the standard.]

The value of nnpfga\_target\_id shall be in the range of 0 to 232 − 2, inclusive.

An NNPFGA SEI message with a particular value of nnpfga\_target\_id shall not be present in a current PU unless there is an NNPFGC SEI message with nnpfgc\_id equal to the particular value of nnpfga\_target\_id and nnpfgc\_grouping\_type equal to 0 present in the current PU or in a PU that precedes the current PU in decoding order within the current CLVS.

When a PU contains both an NNPFGC SEI message with a particular value of nnpfgc\_id and an NNPFGA SEI message with nnpfga\_target\_id equal to the particular value of nnpfgc\_id, the NNPFGC SEI message shall precede the NNPFGA SEI message in decoding order.

**nnpfga\_cancel\_flag** equal to 1 indicates that the persistence of the target NNPFG established by any previous NNPFGA SEI message with the same nnpfga\_target\_id as the current SEI message is cancelled, i.e., the target NNPFG is no longer used unless it is activated by another NNPFGA SEI message with the same nnpfga\_target\_id as the current SEI message and nnpfga\_cancel\_flag equal to 0. nnpfga\_cancel\_flag equal to 0 indicates that the target NNPFG is activated for use.

**nnpfga\_persistence\_flag** specifies the persistence of the target NNPFG for the current layer.

nnpfga\_persistence\_flag equal to 0 specifies that the target NNPFG may be used for post-processing filtering for the current picture only.

nnpfga\_persistence\_flag equal to 1 specifies that the target NNPFG may be used for post-processing filtering for the current picture and all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture in the current layer associated with a NNPFGA SEI message with the same nnpfga\_target\_id as the current SEI message that follows the current picture in output order.

NOTE 2 – The target NNPFG is not applied for this subsequent picture in the current layer associated with a NNPFGA SEI message with the same nnpfga\_target\_id as the current SEI message.

Let the nnpfgcTargetPictures be the set of pictures to which the last NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id that precedes the current NNPFGA SEI message in decoding order pertains. [Ed. Check phrasing of this. “Pertains” is not used in a similar way anywhere in the standard.] Let nnpfgaTargetPictures be the set of pictures for which the target NNPFG is activated by the current NNPFGA SEI message. It is a requirement of bitstream conformance that any picture included in nnpfgaTargetPictures shall also be included in nnpfgcTargetPictures.

**nnpfga\_no\_prev\_clvs\_flag** equal to 1 specifies that the input pictures for the NNPF group do not originate from a previous CLVS. nnpfga\_no\_prev\_clvs\_flag equal to 0 specifies that the input pictures for the NNPF group may or may not originate from a previous CLVS.

NOTE 3 – The value of nnpfga\_no\_prev\_clvs\_flag can be changed from 0 to 1, when the current CLVS is spliced from another bitstream next to the previous CLVS and this NNPFGA SEI message would cause one or more input pictures to be selected from one or more previous CLVSs and therefore is likely to impact the output of the target NNPF group negatively.

**nnpfga\_no\_foll\_clvs\_flag** equal to 1 specifies that when this NNPFGA SEI message persists for the last PU of a CLVS in output order, the NNPFGA SEI message is treated like it persisted for the last PU, in output order, of the current layer within the bitstream. When this NNPFGA SEI message does not persist for the last PU, in output order, of a CLVS in output order or nnpfga\_no\_foll\_clvs\_flag is equal to 0, the value of nnpfga\_no\_foll\_clvs\_flag causes no specific impact.

NOTE 4 – The value of nnpfga\_no\_foll\_clvs\_flag can be changed from 0 to 1 for an NNPF group including a picture-rate-upsampling NNPF, when the following CLVS is spliced from a different bitstream next to the current CLVS. Consequently, the NNPF process of the picture-rate-upsampling NNPF interpolates pictures up to the end of the current CLVS using input pictures originating from the current CLVS only.

**nnpfga\_num\_filters\_minus2** plus 2indicates the number of NNPFs in the NNPFG that this SEI message activates. The value of nnpfga\_num\_filters\_minus2 shall be equal to the value of nnpfgc\_num\_members\_minus2 in an NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id.

**nnpfga\_target\_base\_flag**[ i ] equal to 1 specifies that the i-th NNPF in the target NNPFG is the base NNPF with nnpfc\_id equal to nnpfgc\_member\_id[ i ] in an NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id. nnpfga\_target\_base\_flag[ i ] equal to 0 specifies that the i-th NNPF in the target NNPFG is the NNPF specified by the last NNPFC SEI message that has nnpfc\_id equal to nnpfgc\_member\_id[ i ] in an NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id, precedes the first VCL NAL unit of the current picture in decoding order, and is not a repetition of the NNPFC SEI message that contains the base NNPF.

**nnpfga\_input\_all\_pics\_flag**[ i ] equal to 1 specifies that the input pictures to the i-th NNPF are selected from the list of candidate input pictures candInputPicList[ i ] without skipping. nnpfga\_input\_all\_pics\_flag[ i ] equal to 0 specifies that the input pictures to the i-th NNPF are selected from the list of candidate input pictures candInputPicList[ i ] in a manner that some candidate input pictures are skipped.

**nnpfga\_num\_input\_pics\_minus1**[ i ] specifies the number of input pictures for the i-th NNPF in the target NNPFG. When present, nnpfga\_num\_input\_pics\_minus1[ i ] shall be equal to nnpfc\_num\_input\_pics\_minus1 for an NNPF with nnpfc\_id equal to nnpfgc\_member\_id[ i ] of an NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id. When not present, nnpfga\_num\_input\_pics\_minus1[ i ] is inferred to be equal to nnpfc\_num\_input\_pics\_minus1 for an NNPF with nnpfc\_id equal to nnpfgc\_member\_id[ i ] in an NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id.

**nnpfga\_input\_pic\_skip\_count**[ i ][ j ] specifies a j-th picture count that is skipped in the list of candidate input pictures candInputPicList[ i ] when selecting input pictures for the NNPF activated by the i-th loop entry. When nnpfga\_input\_pic\_skip\_count[ i ][ j ] is not present, it is inferred to be equal to 0 for all values of j in the range of 0 to nnpfga\_num\_input\_pics\_minus1[ i ], inclusive. The variable numCandInputPics, which indicates the number of candidate input pictures to the NNPFG, is derived as follows:

numCandInputPics = 0  
for( j = 0; j <= nnpfga\_num\_input\_pics\_minus1[ 0 ]; j++ )  
 numCandInputPics += 1 + nnpfga\_input\_pic\_skip\_count[ 0 ][ j ] (xx)

Let candInputPicList[ m ] for m in the range of 1 to nnpfga\_num\_filters\_minus2 + 1, inclusive, be a list of pictures in inverse output order that is initially empty and formed in decreasing order of n in the range of 0 to m − 1, inclusive, by including each picture that is output by the NNPF process of the n-th loop entry that has no corresponding picture already present in candInputPicList[ m ], and lastly including each picture present in candInputPicList[ 0 ] that has no corresponding picture already present in candInputPicList[ m ].

When a candidate input picture candInputPicList[ m ][ idx ] for any value of m in the range of 1 to nnpfga\_num\_filters\_minus2 + 1, inclusive, is an NNPF output picture of the n-th NNPF process with the value of n being less than the value of m, the width and height of the candidate input picture are respectively equal to nnpfcOutputPicWidth and nnpfcOutputPicHeight of the NNPF output picture.

The list of input pictures inputPicList[ m ] to the NNPF of the m-th loop entry is derived as follows:

candIdx = 0  
for( k = 0; k <= nnpfga\_num\_input\_pics\_minus1[ m ]; k++ ) {  
 candIdx += nnpfga\_input\_pic\_skip\_count[ m ][ k ]  
 inputPicList[ m ][ k ] = candInputPicList[ m ][ candIdx ] (xx)  
 candIdx++  
}

It is a requirement of bitstream conformance that candIdx shall not exceed the number of pictures in candInputPicList[ m ].

It is a requirement of bitstream conformance that the pictures present in inputPicList[ m ], for any value of m in the range of 1 to nnpfga\_num\_filters\_minus2 + 1, inclusive, shall have the same width, height, bit depth, and chroma format.

For purposes of interpretation of the NNPFC SEI message with nnpfc\_id equal to nnpfgc\_member\_id[ i ] in an NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id, the following variables are specified for the i-th loop entry:

– The variables BitDepthY, BitDepthC, and ChromaFormatIdc are used as provided for the interpretation of this SEI message.

– CroppedWidth and CroppedHeight are set equal to the width and height of the pictures in inputPicList[ i ], respectively, in units of luma samples.

– For each input picture k in the range of 0 to nnpfga\_num\_input\_pics\_minus1[ i ], inclusive, the following applies:

– CroppedYPic[ k ], CroppedCbPic[ k ], and CroppedCrPic[ k ], when present, are set equal to respective sample array of inputPicList[ i ][ k ]

– When nnpfc\_auxiliary\_inp\_idc is equal to 1 for the NNPF with nnpfc\_id equal to nnpfgc\_member\_id[ i ] in an NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id, the following applies:

– It is a requirement of bitstream conformance that inputPicList[ i ][ k ] is the same as candInputPicList[ 0 ][ idx ] for any value of idx in the range of 0 to numCandInputPics − 1, inclusive.

– StrengthControlVal[ k ] is set equal to InitStrengthControlVal[ idx ].

**nnpfga\_num\_output\_entries**[ i ] specifies the number of nnpfga\_output\_flag[ i ][ j ] syntax elements present in the NNPFGA SEI message. The value of nnpfga\_num\_output\_entries[ i ] shall be in the range of 0 to NumInpPicsInOutputTensor, inclusive, for an NNPF with nnpfc\_id equal to nnpfgc\_member\_id[ i ] of an NNPFGC SEI message with nnpfgc\_id equal to nnpfga\_target\_id..

**nnpfga\_output\_flag**[ i ][ j ] equal to 1 specifies that the NNPF-generated picture that corresponds to the input picture having index InpIdx[ j ] derived for the i-th NNPF of the target NNPFG is output by the NNPF process activated by this loop entry, where the NNPF process is specified in the semantics of the NNPFC SEI message. nnpfga\_output\_flag[ i ][ j ] equal to 0 specifies that the NNPF-generated picture that corresponds to the input picture having index InpIdx[ j ] derived for the i-th NNPF of the target NNPFG is not output by the NNPF process activated by this loop entry. When nnpfga\_num\_output\_entries[ i ] is less than NumInpPicsInOutputTensor derived for the i-th NNPF of the target NNPFG, nnpfga\_output\_flag[ i ][ j ] is inferred to be equal to 1 for each value of i in the range of nnpfga\_num\_output\_entries[ i ] to NumInpPicsInOutputTensor − 1, inclusive.

Let NnpfgaOutputPicList, which is the list of pictures output by NNPF process of the NNPFG in output order, be initially empty and formed in decreasing order of n in the range of 0 to nnpfga\_num\_filters\_minus2 + 1, inclusive, by including each picture that is output by the NNPF process of the n-th loop entry that has no corresponding picture already present in NnpfgaOutputPicList.

**nnpfga\_num\_output\_pic\_update** specifies the number of output picture update flag that is present. When nnpfga\_num\_output\_pic\_update is greater than 0, nnpfga\_output\_pic\_update\_flag[ i ] for i in the range from 0 to nnpfga\_num\_output\_pic\_update − 1, inclusive, are present. The value of nnpfga\_num\_output\_pic\_update shall be in the range from 0 to number of pictures in NnpfgaOutputPicList, inclusive.

**nnpfga\_output\_pic\_update\_flag**[ i ] equal to 1 specifies that the i-th picture NnpfgaOutputPicList is included in the final output picture list. nnpfga\_output\_pic\_update\_flag[ i ] equal to 0 specifies that the i-th picture NnpfgaOutputPicList is not included in the final output picture list. When the value of nnpfga\_num\_output\_pic\_update is less than the number of pictures in NnpfgaOutputPicList, the value of nnpfga\_output\_pic\_update\_flag [ i ] for i in the range from nnpfga\_num\_output\_pic\_update to the number of pictures in NnpfgaOutputPicList − 1, inclusive, is inferred to be equal to 1.

Let numCandidateOutputPics be the number of pictures in NnpfgaOutputPicList. The list of output picture flag candOutputPicFlag[ i ] for i in the range from 0 to numCandidateOutputPics − 1, inclusive is derived as follows:

for( i = 0; i < numCandidateOutputPics; i++ )  
 candOutputPicFlag[ i ] = 1  
for( i = 0; i < nnpfga\_num\_output\_pic\_update; i++ ) (xx)  
 candOutputPicFlag[ i ] = nnpfga\_output\_pic\_update\_flag[ i ]

Let FinalNnpfgaOutputPicList, which is the list of pictures output by NNPF process of the NNPFG in output order, be derived as follows:

j = 0  
for( i = 0; i < numCandidateOutputPics; i++ )  
 if( candOutputPicFlag[ i ] ) (xx)  
 FinalNnpfgaOutputPicList[ j++ ] = NnpfgaOutputPicList[ i ]

*Alternative changes to VSEI subclause 8.28 and VVC subclause D.12.11 for adding the support of grouping of PFs (both NNPFs and non-NN PFs)*

**8.28 General post-processing filtering process and neural-network post-filter SEI messages**

**8.28.1 General post-processing filtering process using PPFs**

**8.28.1.1 General**

Input to this process is a bitstream BitstreamToFilter. Output of this process is a list of output pictures ListOutputPics.

A post-processing filter (PPF) may be indicated by an SEI message for which the payloadType value is in SeiProcessingOrderSeiList, specified in the semantics of the SEI processing order (SPO) SEI message.

For each picture, there can be multiple PPFs activated and belonging to one or more PPF groups. PPF groups are alternative to each other, i.e., at most one group can be chosen to be applied.

A special PPF cascading case is defined as the case when such two PPFs are both activated for a picture: the two PPFs are both NNPFs (i.e., the payloadType value for the PPFs indicates the NNPFC SEI message), one of the two NNPFs has nnpfc\_purpose equal to 4 and the other has multiple input pictures, and neither of the two NNPFs is associated with an SEI processing order SEI message. In this case, the two NNPFs are implicitly considered as belonging to one PPF group, and the NNPF with nnpfc\_purpose equal to 4 is applied first.

Except for the special PPF cascading case, each PPF group containing multiple PPFs is associated with an SPO SEI message with a particular value of po\_id.

Except for the special PPF cascading case, any PPF not associated with an SPO SEI message is in its own PPF group.

One or more PPFs in the chosen PPF group can be applied. When multiple PPFs (in the chosen PPF group) are applied, they are applied in the cascading manner, meaning that they are applied in the order indicated by the SEI processing order SEI message associated with the chosen PPF group, and for each applied PPF that is not the last applied PPF, the output is used as the input of the next applied PPF.

First, BitstreamToFilter is decoded, the list CroppedDecodedPictures is set to be the list of the cropped decoded pictures in output order resulted from decoding BitstreamToFilter, and the list ListOutputPics is initialized to be the same as CroppedDecodedPictures.

Second, the filtering process for one picture, as specified in subclause 8.28.1.2, is repeatedly invoked, in output order, for each cropped decoded picture that is in CroppedDecodedPictures and for which one or more PPFs of one or more PPF groups are activated and only one of the groups is chosen to be applied.

The order of the pictures in ListOutputPics is in output order.

Within ListOutputPics there shall be no more than one picture pertaining to any particular output time instance. [Ed. Check phrasing of this. “Pertains” is not used in a similar way anywhere in the standard.] When for any particular picture in CroppedDecodedPictures there are PPFs of multiple PPF groups activated, the above constraint shall apply regardless of which group of PPFs is chosen to be applied when the particular picture is the current picture.

For any particular pair of pictures inputPicA and inputPicB consecutive in output order in CroppedDecodedPictures, when there are one or more pictures interpolatedPicSetA in ListOutputPics between inputPicA and inputPicB in output order, the pictures in interpolatedPicSetA shall be among the pictures that were output by applying a particular PPF ppfA when a particular picture currPicA in CroppedDecodedPictures was the current picture. The application of any other PPF that was used in the filtering process for one picture when currPicA was the current picture or the application of any PPF (including ppfA) that was used in the filtering process for one picture when any other picture currPicB in CroppedDecodedPictures was the current picture shall not output any picture between the inputPicA and inputPicB in output order.

NOTE – The intent of the constraints expressed in the above paragraph is to disallow generating PPF output pictures between any particular pair of consecutive input pictures more than once.

**8.28.1.2 Filtering process for one picture**

The filtering process specified in this subclause applies to each cropped decoded picture, referred to as the current picture, that is in CroppedDecodedPictures and for which one or more groups of PPFs are activated, only one of the PPF groups is chosen to be applied, and the number of PPFs (in the chosen PPF group) to be applied is greater than 0.

The filtering process for one picture using one PPF, as specified in subclause 8.28.1.3, is repeatedly invoked for each of the PPFs to be applied. When the number of PPFs to be applied is greater than 1, the following applies:

– If the special PPF cascading case applies for the chosen PPF group, the NNPF with nnpfc\_purpose equal to 4 is applied first, followed by the NNPF with multiple input pictures.

– Otherwise (the special PPF cascading case does not apply for the chosen PPF group), the PPFs are applied in the preferred order indicated by the SEI processing order SEI message associated with the chosen group of PPFs.

**8.28.1.3 Filtering process for one picture using one PPF**

The filtering process specified in this subclause applies when a particular PPF is applied when a particular picture is the current picture.

Before the PPF is applied, when the PPF is the first PPF to be applied, the list CandInputPicList is set to be identical to CroppedDecodedPictures.

When applying a PPF to the current picture, the input pictures for the PPF are selected from the list CandInputPicList, and the order of the pictures generated and output by the PPF are in output order.

When applying a PPF that is an NNPF to the current picture, the following applies:

– The filtered and/or interpolated pictures are generated by the NNPF by applying the NNPF process specified in the semantics of the NNPFC SEI message, in a patch-wise manner, to the current picture.

– The order of the pictures generated by the NNPF by applying the NNPF process being stored into the output tensor of the NNPF is in output order.

After the PPF is applied, the lists CandInputPicList and ListOutputPics are both updated, in the same manner, by 1) replacing each of those pictures in the list having a corresponding PPF output picture of the PPF with the corresponding PPF output picture, and 2) inserting those interpolated pictures, if any, into the list and placing them such that all pictures in the updated list are in output order.

**8.28.2 Neural-network post-filter characteristics SEI message**

**8.28.2.1 Neural-network post-filter characteristics SEI message syntax**

(no change)

**8.28.2.2 Neural-network post-filter characteristics SEI message semantics**

The neural-network post-filter characteristics (NNPFC) SEI message specifies a neural network that may be used as a post-processing filter. The use of specified neural-network post-processing filters (NNPFs) for specific pictures is indicated with neural-network post-filter activation (NNPFA) SEI messages.

Use of this SEI message requires the definition of the following variables:

– The list CandInputPicList that contains a list of pictures in output order from which the input pictures for the NNPF are selected.

NOTE 1 – This list is updated by the PPF filtering process each time when a PPF is applied for the current picture unless the PPF is the last PPF that is applied for the current picture.

– Input picture width and height in units of luma samples, denoted herein by CroppedWidth and CroppedHeight, respectively.

– Luma sample array CroppedYPic[ idx ] and chroma sample arrays CroppedCbPic[ idx ] and CroppedCrPic[ idx ], when present, of the input pictures with index idx in the range of 0 to numInputPics − 1, inclusive, that are used as input for the NNPF.

– Bit depth BitDepthY for the luma sample array of the input pictures.

– Bit depth BitDepthC for the chroma sample arrays, if any, of the input pictures.

– Chroma format indicator ChromaFormatIdc, as described in subclause 7.3, of the input pictures.

– When nnpfc\_auxiliary\_inp\_idc is equal to 1, a filtering strength control value array StrengthControlVal[ idx ] that shall contain real numbers in the range of 0 to 1, inclusive, of the input pictures with index idx in the range of 0 to numInputPics − 1, inclusive.

Input picture with index 0 is the picture in CandInputPicList corresponding to the picture for which the NNPF defined by this NNPFC SEI message is activated by an NNPFA SEI message. Input picture with index i in the range of 1 to numInputPics − 1, inclusive, precedes the input picture with index i − 1 in output order.

NOTE 2 – The picture in CandInputPicList corresponding to a picture is either the cropped decoded output picture of that picture or a filtered version of the cropped decoded output picture that was an output picture of a previously applied PPF.

The variables SubWidthC and SubHeightC are derived from ChromaFormatIdc as specified by Table 2.

…

**nnpfc\_absent\_input\_pic\_zero\_flag** equal to 1 indicates that the NNPF expects an input picture corresponding to a picture that is not present in the bitstream to be represented by sample arrays with sample values equal to 0. nnpfc\_absent\_input\_pic\_zero\_flag equal to 0 indicates that the NNPF expects an input picture inputPicA corresponding to a picture that is not present in the bitstream to be represented by the input picture inputPicB that is the closest to inputPicA in output order and the picture corresponding to inputPicB is present in the bitstream.

…

**nnpfa\_no\_prev\_clvs\_flag** equal to 1 specifies that the pictures corresponding to the input pictures for the NNPF do not originate from a previous CLVS. nnpfa\_no\_prev\_clvs\_flag equal to 0 specifies that the pictures corresponding to the input pictures for the NNPF may or may not originate from a previous CLVS.

NOTE 4 – The value of nnpfa\_no\_prev\_clvs\_flag can be changed from 0 to 1, when the current CLVS is spliced from another bitstream next to the previous CLVS and this NNPFA SEI message would cause one or more input pictures to be selected that have corresponding pictures from one or more previous CLVSs and therefore is likely to impact the output of the target NNPF negatively.

**nnpfa\_no\_foll\_clvs\_flag** equal to 1 specifies that when this NNPFA SEI message persists for the last PU of a CLVS in output order, the NNPFA SEI message is treated like it persisted for the last PU, in output order, of the current layer within the bitstream. When this NNPFA SEI message does not persist for the last PU, in output order, of a CLVS in output order or nnpfa\_no\_foll\_clvs\_flag is equal to 0, the value of nnpfa\_no\_foll\_clvs\_flag causes no specific impact.

NOTE 5 – The value of nnpfa\_no\_foll\_clvs\_flag can be changed from 0 to 1 for a picture-rate-upsampling NNPF, when the following CLVS is spliced from a different bitstream next to the current CLVS. Consequently, the NNPF process interpolates pictures up to the end of the current CLVS using input pictures corresponding to pictures originating from the current CLVS only.

…

**8.28.3 Neural-network post-filter activation SEI message**

(no change)

**D.12.11 Use of the post-processing filter SEI messages, including the neural network post-filter characteristics SEI message and the neural network post-filter activation SEI message**

A post-processing filter (PPF) may be indicated by an SEI message for which the payloadType value is in SeiProcessingOrderSeiList, which is specified in the semantics of the SEI processing order SEI message.

Let currPic be the cropped decoded output picture for which the post-processing filter (PPF), e.g., neural-network post-processing filter (NNPF) defined by the neural-network post-filter characteristics (NNPFC) SEI message, is activated, e.g., by a neural-network post-filter activation (NNPFA) SEI message, and currLayerId be the nuh\_layer\_id value of currPic.

The list candInputPicList contains a list of pictures in output order from which the input pictures for the PPF are selected.

NOTE 1 – This list is updated by the PPF filtering process each time when a PPF is applied for the current picture unless the PPF is the last PPF that is applied for the current picture.

When the PPF is not an NNPF, the PPF is considered to have only one input picture that is the picture in candInputPicList corresponding to the current picture.

NOTE 2 – The picture in candInputPicList corresponding to a cropped decoded output picture is either the cropped decoded output picture itself or a filtered version of the cropped decoded output picture that was an output picture of a previously applied PPF.

If the PPF is the first PPF that is applied for the current picture, the following applies:

– CroppedWidth is set equal to the value of pps\_pic\_width\_in\_luma\_samples − ‌SubWidthC \* ‌ ( pps\_conf\_win\_left\_offset + pps\_conf\_win\_right\_offset ) for currPic.

– CroppedHeight is set equal to the value of pps\_pic\_height\_in\_luma\_samples − ‌SubHeightC \* ‌ ( pps\_conf\_win\_top\_offset + pps\_conf\_win\_bottom\_offset ) for currPic.

– BitDepthY and BitDepthC are both set equal to BitDepth.

– ChromaFormatIdc is set equal to sps\_chroma\_format\_idc.

Otherwise (the PPF is not the first PPF that is applied for the current picture), the following applies:

– CroppedWidth is set equal to the picture width in units of luma samples of the picture in candInputPicList corresponding to the current picture.

– CroppedHeight is set equal to the picture height in units of luma samples of the picture in candInputPicList corresponding to the current picture.

– BitDepthY is set equal to the bit depth BitDepthY for the luma sample array of the picture in candInputPicList corresponding to the current picture.

– BitDepthC is set equal to the bit depth BitDepthC for the chroma sample arrays, if any, of the picture in candInputPicList corresponding to the current picture.

– ChromaFormatIdc is set equal to the chroma format indicator ChromaFormatIdc of the the picture in candInputPicList corresponding to the current picture.

The remainder of this subclause applies when the PPF is an NNPF.

It is a requirement of bitstream conformance that when a picture unit contains an NNPFA SEI message, the value of ph\_pic\_output\_flag in the picture header contained in that picture unit shall be equal to 1.

NOTE 3 – Since when the NNPF is the first PPF that is applied for currPic only cropped decoded output pictures are used as input pictures of the NNPF, the value of ph\_pic\_output\_flag in the picture header of the coded picture corresponding to each input picture of the NNPF is equal to 1.

The variable pictureRateUpsamplingFlag is set equal to ( ( nnpfc\_purpose & 0x08 ) > 0 ) ? 1 : 0.

The variable numInputPics is set equal to nnpfc\_num\_input\_pics\_minus1 + 1.

The variable numInferences is derived as follows:

– If all of the following conditions are true, the variable numPostRoll is set equal to the value of i such that nnpfc\_interpolated\_pics[ i ] is greater than 0 and the variable numInferences is set equal to 1 + numPostRoll:

– nnpfc\_purpose is equal to 8 (i.e., the only purpose for the NNPF is picture rate upsampling).

– nnpfa\_persistence\_flag is equal to 1.

– nnpfc\_interpolated\_pics[ i ] is greater than 0 only for a single value of i that is greater than 0.

– Either of the following conditions is true:

– currPic is the last picture of the bitstream in output order that has nuh\_layer\_id equal to currLayerId.

– currPic is the last picture in the CLVS in output order and nnpfa\_no\_foll\_clvs\_flag is equal to 1.

– Otherwise, if all of the following conditions are true, the variable numPostRoll is set equal to InpIdx[ i ] for the value of i such that nnpfa\_output\_flag[ i ] is equal to 1, and the variable numInferences is set equal to 1 + numPostRoll:

– pictureRateUpsamplingFlag is equal to 0.

– numInputPics is greater than 1.

– nnpfa\_persistence\_flag is equal to 1.

– nnpfa\_output\_flag[ idx ] is equal to 1 for a single value of idx in the range of 0 to NumInpPicsInOutputTensor − 1, inclusive, and for that single value of idx, InpIdx[ idx ] is greater than 0.

– Either of the following conditions is true:

– currPic is the last picture of the bitstream in output order that has nuh\_layer\_id equal to currLayerId.

– currPic is the last picture in the CLVS in output order and nnpfa\_no\_foll\_clvs\_flag is equal to 1.

– Otherwise, the variable numInferences is set equal to 1.

For each value of j in the range of 0 to numInferences − 1, inclusive, the following applies for the derivation of the input pictures for the NNPF such that each input picture is a picture in candInputPicList:

– The arrays inputPic[ i ] and inputPresentFlag[ i ] for i in the range of 0 to numInputPics − 1, inclusive, representing all the input pictures and the presence of input pictures, respectively, are specified as follows:

– When j is greater than 0, for each value of k in the range of 0 to j − 1, inclusive, inputPic[ k ] is set to be the picture in candInputPicList corresponding to currPic and inputPresentFlag[ k ] is set equal to 0.

– The j-th input picture, inputPic[ j ], is set to be the picture in candInputPicList corresponding to currPic and inputPresentFlag[ j ] is set equal to 1.

– When numInputPics is greater than 1, the following applies for each value of i in the range of j + 1 to numInputPics − 1, inclusive, in increasing order of i:

– If both of the following conditions are true, inputPic[ i ] is set to be the picture in candInputPicList corresponding to prevPic and inputPresentFlag[ i ] is set equal to 1:

– Either of the following conditions is true:

– pictureRateUpsamplingFlag is equal to 1 and currPic is associated with a frame packing arrangement SEI message with frame\_packing\_arrangement\_type equal to 5 and a particular value of fp\_current\_frame\_is\_frame0\_flag, and there is a cropped decoded output picture prevPic that is the last picture in output order among all cropped decoded output pictures that have nuh\_layer\_id equal to currLayerId, precede inputPic[ i − 1 ] in output order, and are associated with a frame packing arrangement SEI message with frame\_packing\_arrangement\_type equal to 5 and the same value of fp\_current\_frame\_is\_frame0\_flag.

– pictureRateUpsamplingFlag is equal to 0 or currPic is not associated with a frame packing arrangement SEI message with frame\_packing\_arrangement\_type equal to 5, and there is a cropped decoded output picture prevPic that is the last picture in output order among all cropped decoded output pictures that have nuh\_layer\_id equal to currLayerId and precede inputPic[ i − 1 ] in output order.

– nnpfa\_no\_prev\_clvs\_flag is equal to 0 or the coded picture corresponding to prevPic and the current picture are present in the same CLVS.

– Otherwise, the following applies:

– inputPic[ i ] is set to be the same picture as inputPic[ i − 1 ] and inputPresentFlag[ i ] is set equal to 0.

– It is a requirement of bitstream conformance that, when pictureRateUpsamplingFlag is equal to 1, nnpfc\_interpolated\_pics[ i − 1 ] shall be equal to 0.

– It is a requirement of bitstream conformance that when inputPresentFlag[ i ] is equal to 0 and nnpfc\_input\_pic\_output\_flag[ i ] is equal to 1, the value of nnpfa\_output\_flag[ idx ] shall be equal to 0 for the value of idx such that InpIdx[ idx ] is equal to i.

– For purposes of interpretation of the NNPFC SEI message, the following variables are specified:

– The luma sample arrays CroppedYPic[ i ] and the chroma sample arrays CroppedCbPic[ i ] and CroppedCrPic[ i ], when present, are derived as follows for each value of i in the range of 0 to numInputPics − 1, inclusive:

– The variable sourcePic is derived as follows:

– If inputPresentFlag[ i ] is equal to 1 or nnpfc\_absent\_input\_pic\_zero\_flag is equal to 0, sourcePic is set to be inputPic[ i ].

– Otherwise (inputPresentFlag[ i ] is equal to 0 and nnpfc\_absent\_input\_pic\_zero\_flag is equal to 1), sourcePic is set to be a picture with a luma sample array of CroppedWidth × CroppedHeight samples equal to 0 and Cb and Cr sample arrays of ( CroppedWidth / SubWidthC ) × ( CroppedHeight / SubHeightC ) samples equal to 0.

– The luma sample array CroppedYPic[ i ] and the chroma sample arrays CroppedCbPic[ i ] and CroppedCrPic[ i ], when present, are set to be the 2-dimensional arrays of decoded sample values of the Y, Cb and Cr components, respectively, of sourcePic.

– The array StrengthControlVal[ i ] for all values of i in the range of 0 to numInputPics − 1, inclusive, specifying the filtering strength control value for the input pictures for the NNPF, is derived as follows:

– StrengthControlVal[ i ] is set equal to the value of ( firstSliceQpY + QpBdOffset ) ÷ ( 63 + QpBdOffset ), where firstSliceQpY is equal to SliceQpY of the first slice of the cropped decoded output picture corresponding to inputPic[ i ].

There shall not be more than two NNPFC SEI messages present in a picture unit with the same value of nnpfc\_id. When there are two NNPFC SEI messages present in a picture unit with the same value of nnpfc\_id, these SEI messages shall have different content. When two NNPFC SEI messages with the same nnpfc\_id and different content are present in the same picture unit, both of these NNPFC SEI messages shall be in the same SEI NAL unit.

*Modify clause 8.30 as indicated with the highlighting:*

* 1. **Encoder optimization information SEI message**
     1. **Encoder optimization information SEI message syntax**

|  |  |
| --- | --- |
| encoder\_optimization\_info(payloadSize ) { | **Descriptor** |
| **eoi\_cancel\_flag** | u(1) |
| if( !eoi\_cancel\_flag ) { |  |
| **eoi\_persistence\_flag** | u(1) |
| **eoi\_for\_human\_viewing\_idc** | u(2) |
| **eoi\_for\_machine\_analysis\_idc** | u(2) |
| **eoi\_type** | u(16) |
| if( eoi\_persistence\_flag ) { |  |
| **eoi\_num\_sublayers** | u(3) |
| if( eoi\_num\_sublayers < 7 ) |  |
| **eoi\_min\_tid** | u(3) |
| } |  |
| if( EoiObjectBasedFlag ) |  |
| … |  |

* + 1. **Encoder optimization information SEI message semantics**

…

**eoi\_persistence\_flag** specifies the persistence of the optimization information provided in this SEI message. eoi\_persistence\_flag equal to 0 specifies that the optimization information applies for the current picture only. eoi\_persistence\_flag equal to 1 specifies that the optimization information applies for the pictures in the temporal sublayers indicated by eoi\_num\_sublayers and eoi\_min\_tid, when present, among the current picture and all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

…

**eoi\_num\_sublayers** equal to 7 specifies that this SEI message applies to all temporal sublayers. eoi\_num\_sublayers less than 7 specifies the number of temporal sublayers to which this SEI message applies. eoi\_num\_sublayers equal to 0 is reserved.

**eoi\_min\_tid** specifies the smallest temporal sublayer identifier value to which this SEI message applies. The value of eoi\_min\_tid shall be in the range of 0 to 6 − eoi\_num\_sublayers, inclusive. When eoi\_persistence\_flag is equal to 1 and eoi\_min\_tid is not present, eoi\_min\_tid is inferred to be equal to 0.

When eoi\_persistence\_flag is equal to 1, the variable eoiMaxSublayer, which is the greatest sublayer identifier value to which this SEI message applies, is derived as follows:

eoiMaxSublayer = eoi\_min\_tid + eoi\_num\_sublayers − 1

When eoi\_persistence\_flag is equal to 1, the semantics of eoi\_for\_human\_viewing\_flag and eoi\_for\_machine\_analysis\_flag apply to the sublayer representation of eoiMaxSubLayer and the semantics of eoi\_type, eoi\_object\_based\_idc (when present), eoi\_temporal\_resampling\_type\_flag (when present), and eoi\_num\_int\_pics (when present) apply to the sublayers with a sublayer identifier in the range of eoi\_min\_tid to eoiMaxSublayer, inclusive.

…

When multiple encoder optimization information SEI messages are present in a particular PU, the following constraints apply for these SEI messages:

– All of the SEI messages shall have the same value of eoi\_cancel\_flag.

– When eoi\_persistence\_flag is present in the SEI messages, all the SEI messages shall have the same value of eoi\_persistence\_flag.

– When any of the SEI messages has eoi\_num\_sublayers equal to 7, all the SEI messages shall have the same SEI payload content.

– When an SEI message among the SEI messages has eoi\_num\_sublayers less than 7, the following constraints apply:

– All the SEI messages with the same values of eoi\_num\_sublayers and eoi\_min\_tid, which are assigned to numSubLayersA and minTidA, respectively, shall have the same SEI payload content.

– There shall be no such SEI message among the SEI messages for which both of the following are true:

– eoi\_num\_sublayers is not equal to numSubLayersA or eoi\_min\_tid is not equal to minTidA.

– eoi\_min\_tid is greater than minTidA and less than minTidA + numSubLayersA.

*Add VSEI subclauses 8.34 to 8.45*

8.34 Image format metadata SEI messages

8.34.1 Image format metadata SEI message syntax

|  |  |
| --- | --- |
| image\_format\_metadata( payloadSize ) { | **Descriptor** |
| **ifm\_cancel\_flag** | u(1) |
| if ( !ifm\_cancel\_flag ) { |  |
| **ifm\_persistence\_flag** | u(1) |
| **ifm\_num\_metadata\_payloads** | ue(v) |
| for ( i=0; i<ifm\_num\_metadata\_payloads; i++ ) { |  |
| **ifm\_type\_id**[ i ] | ue(v) |
| **ifm\_uri\_present\_flag**[ i ] | u(1) |
| if( !ifm\_uri\_present\_flag[ i ] ) { |  |
| **ifm\_payload\_len\_minus1**[ i ] | ue(v) |
| for( j = 0; j <= ifm\_payload\_len[ i ]; j++ ) |  |
| **ifm\_data\_payload\_byte**[ i ][ j ] | b(8) |
| } |  |
| else { |  |
| while( !byte\_aligned( ) ) |  |
| **ifm\_bit\_equal\_to\_zero** | f(1) |
| **ifm\_data\_uri**[ i ] | st(v) |
| } |  |
| } |  |
| } |  |
| } |  |

8.34.2 Image format metadata SEI message semantics

The image format metadata SEI message specifies an SEI message in which the payload of EXIF, JFIF, XMP, and ICC profile image format metadata can be carried in the video bitstream.

**ifm\_cancel\_flag** equal to 1 indicates that the SEI message cancels the persistence of any previous image format metadata SEI message in output order. ifm\_cancel\_flag equal to 0 indicates that image format information follows.

**ifm\_persistence\_flag** specifies the persistence of the image format metadata SEI message for the current layer.

ifm\_persistence\_flag equal to 0 specifies that the image format metadata SEI message applies to the current decoded picture only.

ifm\_persistence\_flag equal to 1 specifies that the image format metadata SEI message applies to the current decoded picture and persists for all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture in the current layer in an AU associated with an Exif metadata SEI message is output that follows the current picture in output order.

**ifm\_num\_metadata\_payloads** indicates the number of metadata payloads that follow.

**ifm\_bit\_equal\_to\_zero** shall be equal to zero.

**ifm\_type\_id**[ i ]indicates the type of the metadata payload as defined in Table xx.

**Table xx – Definition of ifm\_type\_id[ i ]**

|  |  |
| --- | --- |
| **Value** | **Interpretation** |
| 0 | Exif metadata as specified by (Ed Note: awaiting conclusion of ITU A.5 and A.4 processes) |
| 1 | JFIF metadata from one or more concatenated JFIF segments as specified in ITU-T Recommendation T.871 | International Standard ISO/IEC 10918-5 |
| 2 | one or more JFIF extension markers as specified in ITU-T Recommendation T.871 | International Standard ISO/IEC 10918-5 |
| 3 | first JFIF marker found following the start of the image as specified in ITU-T Recommendation T.871 | International Standard ISO/IEC 10918-5 |
| 4 | XMP metadata as specified in ISO 16684-1: Graphic Technology – Extensible metadata platform (XMP) specification– Part 1 Data model, serialization and core properties. Or ISO 16684-3: Graphic Technology – Extensible metadata platform (XMP) specification – Part 3 JSON-LD serialization of XMP. |
| 5 | ICC profile as specified in 15076-1, 15076-2, or 20677 |
| 6-255 | Reserved |

When ifm\_type\_id[ i ] is equal to 0, the following image metadata payload is Exchangeable Image File (Exif) metadata. The Exif Format for digital still cameras includes a set of metadata that captures information regarding the digital photography process that was used to record an image. Such metadata includes: the acceleration vector of the camera at the time the image was captured, camera lens information, GPS data, the color space used, spectral sensitivity, maximum lens aperture, and more.

Note: Exif metadata is specified by any of the Exchangeable image file (Exif) format for digital still cameras standards developed jointly by the Camera & Imaging Products Association (CIPA) and the Japan Electronics and Information Technology Industries Association (JEITA), for the associated video source pictures prior to encoding, e.g., for camera-captured content. To date, there are multple versions of Exif deployed, each with its own indicator to signal the version number in use.

[Ed. Neither CIPA nor JEITA are listed in the organizations for which normative references may be made to documents published by CIPA or JEITA. The list of such organizations is available at https://www.itu.int/net4/ITU-T/lists/sdo.aspx. To add either CIPA or JEITA to the list, the A.4 process needs to be executed for one or both organizations, and their status as A.5 organizations needs to be approved by SG16. There would probably also need to be a specific list of documents referenced rather than the concept of “any of the” standards produced by those organizations.]

When ifi\_type\_id[ i ] is equal to 1, 2, or 3, the following metadata payload is JFIF metadata, as specified by ITU-T Recommendation T.871 | ISO/IEC International Standard 10918-5.

NOTE 1 – JFIF metadata and its semantics are specified by ITU-T Recommendation T.871 | ISO/IEC International Standard 10918-5 (hereafter, the JFIF standard). Of particular importance is the distinction by JFIF between a JFIF marker that contains information describing the organization of the image data, and a JFIF marker that contains “extension” information.

When ifm\_type\_id[ i ] is equal to 1, the following image format information payload is JFIF metadata from one or more concatenated JFIF marker segments, without the APP0 marker segment identifiers themselves, as specified in ITU-T Recommendation T.871 | International Standard ISO/IEC 10918-5.

When ifm\_type\_id[ i ] is equal to 2, the following image format information payload contains bytes from one or more JFIF extension markers, without the APP0 marker segment identifiers themselves, as specified in ITU-T Recommendation T.871 | International Standard ISO/IEC 10918-5.

When ifm\_type\_id[ i ] is equal to 3, the following image format information payload contains bytes obtained from the first JFIF marker found following the start of the image as specified in ITU-T Recommendation T.871 | International Standard ISO/IEC 10918-5.

When ifm\_type\_id[ i ] is equal to 4, the following image information payload contains Extensible Metadata Platform (XMP) metadata as specified by ISO 16684-1 in XML or ISO 16684-3 in JSON.

NOTE 2 – Extensible Metadata Platform (XMP) metadata is specified in ISO 16684-1: Graphic Technology – Extensible metadata platform (XMP) specification or ISO 16684-3: Graphic Technology – Extensible metadata platform (XMP) specification – Part 3 JSON-LD serialization of XMP. XMP is widely deployed by digital cameras and digital image editing packages to record provenance and editing history with an image. The most common set of metadata included in XMP refers to a vocabulary to store information such as digital rights ownership and names of software packages used to modify the image.

When ifm\_type\_id[ i ] is equal to 5, the following image information payload provides International Color Consortium (ICC) profile metadata for receivers that use a colour management workflow as described by the ICC. ICC profile metadata is specified in ISO 15076-1, ISO 15076-2 (Ed Note: currently at DIS), or ISO 20677.

When ifm\_type\_id[ i ] is equal to 5, ifm\_payload\_len\_minus1[ i ] plus 1 shall not be less than 11, and the data payload corresponding to ifm\_type\_id [ i ] equal to 5 shall contain a complete ICC profile conforming to the referenced ICC profile standard.

When ifm\_type\_id[ i ] is equal to 5, ifm\_data\_payload\_byte[ i ][ j ] is a byte containing data having syntax and semantics that shall be as specified by ISO 15076-1, ISO 15076-2 (ED NOTE: currently at DIS), or ISO 20677.

The variable ICCmajorVer is set equal to ifm\_data\_payload\_byte[ i ][ 8 ] >> 4 and ICCminorVer is set equal to ifm\_data\_payload\_byte[ i ][ 9 ] >> 4 and are interpreted as the ICC profile major and minor versions, respectively, as specified in Tables yy and zz.

Values of ICCmajorVer and ICCminorVer that are not listed in Tables yy and zz are reserved for future use by ITU-T | ISO/IEC. Such values shall not be present in bitstreams conforming to this version of this Specification. Decoders shall ignore ICC profile SEI messages in which such values are present.

**Table yy – ICC major version of profile identifier**

| **ICCmajorVer** | **Interpretation** |
| --- | --- |
| 4 | Major profile version 4 as specified in ISO 15076-1 or ISO 15076-2 |
| 2 | Major profile version 2 as specified in ISO 20677 |

**Table zz – ICC minor version of profile identifier**

| **ICCminorVer** | **Interpretation** |
| --- | --- |
| 0 | Minor profile version 0 as specified in ISO 20677 |
| 2 | Minor profile version 2 as specified in ISO 15076-1 |
| 3 | Minor profile version 3 as specified in ISO 15076-1 |
| 4 | Minor profile version 4 as specified in ISO 15076-2 (currently at DIS) |

**ifm\_uri\_present\_flag**[ i ] signals if the metadata is obtained via a URI with syntax and semantics as specified in IETF Internet Standard 66.

– If ifm\_uri\_present\_flag [ i ] is equal to 0, the image information payload is obtained directly from the payload of the SEI message.

– Otherwise if ifm\_uri\_present\_flag [ i ] is equal to 1, the image information payload is obtained from a URI.

**ifm\_payload\_len\_minus1**[ i ]plus 1 indicates the length of the image information payload according to the ifm\_type\_id[ i ] as described in Table xx.

**ifm\_data\_payload\_byte**[ i ][ j ] shall be a byte containing data having syntax and semantics according to ifm\_type\_id[ i ] as described in Table xx.

**ifm\_data\_uri**[ i ] shall contain a URI with syntax and semantics as specified in IETF Internet Standard 66 identifying the image information payload according to the ifm\_type\_id [ i ] as described in Table xx.

8.35 Multiplane image information SEI message

8.35.1 Multiplane image information SEI message syntax

|  |  |
| --- | --- |
| multiplane\_image\_information( payloadSize ) { | **Descriptor** |
| **mpii\_num\_layers\_minus1** | ue(v) |
| **mpii\_layer\_depth\_equal\_distance\_flag** | u(1) |
| if( mpii\_layer\_depth\_equal\_distance\_flag ) { |  |
| depth\_rep\_info\_element( ZNearSign, ZNearExp, ZNearMantissa, ZNearManLen ) |  |
| depth\_rep\_info\_element( ZFarSign, ZFarExp, ZFarMantissa, ZFarManLen ) |  |
| } else |  |
| for( i = 0; i <= mpii\_num\_layer\_minus1; i++ ) |  |
| depth\_rep\_info\_element( ZSign[ i ], ZExp[ i ], ZMantissa[ i ], ZManLen[ i ] ) |  |
| **mpii\_texture\_opacity\_interleave\_flag** | u(1) |
| if( mpii\_texture\_opacity\_interleave\_flag = = 0 ) |  |
| **mpii\_texture\_opacity\_arrangement\_flag** /\* 0: Top-and-Bottom, 1: Side-by-Side \*/ | u(1) |
| **mpii\_picture\_num\_layers\_in\_height\_minus1** | ue(v) |
| } |  |

8.35.2 Muliplane image information SEI message semantics

The multiplane image information (MPII) SEI message specifies multiplane image (MPI) scene representation information that may be used for view synthesis.

When an MPII SEI message is present in any AU of a CLVS, an MPII SEI message shall be present in the first AU of the CLVS and persists for all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture in the current layer in an AU associated with an MPII SEI message is output that follows the current picture in output order.

NOTE 1 – This SEI message can work together with the multiview acquisition information SEI message for view synthesis. The multiview acquisition information SEI message specifies the intrinsic and extrinsic parameters for current camera view. When multiple views are available, the reconstructed novel views can be rendered from multiplane images of nearby views.

Use of this SEI message requires the definition of the following variables:

– Cropped decoded output picture width and height in units of luma samples, denoted herein by CroppedWidth and CroppedHeight, respectively.

– A chroma format indicator, denoted herein by ChromaFormatIdc, as described in subclause ‎7.3.

– A cropped decoded picture array decPicCurr0[ cIdx ][ x ][ y ], with cIdx = 0..(ChromaFormatIdc = = 0 ) ? 0 : 2, x = 0..( cIdx = = 0 ) ? CroppedWidth : CroppedWidth / SubWidthC − 1, y = 0..( cIdx = = 0 ) ? CroppedHeight : CroppedHeight / SubHeightC − 1.

– In output order a temporally following cropped decoded picture array decPicCurr1[ cIdx ][ x ][ y ], with cIdx = 0..(ChromaFormatIdc = = 0 ) ? 0 : 2, x = 0..( cIdx = = 0 ) ? CroppedWidth : CroppedWidth / SubWidthC − 1, y = 0..( cIdx = = 0 ) ? CroppedHeight : CroppedHeight / SubHeightC − 1.

The variables SubWidthC and SubHeightC are derived from ChromaFormatIdc as specified by Table 2.

**mpii\_num\_layers\_minus1** plus 1 specifies the number of texture and opacity layers for the MPI representation.

**mpii\_layer\_depth\_equal\_distance\_flag** equal to 1 indicates that equal distances are used to generate MPI layers and depth parameters for each layer. In this case, Z[ i ] can be derived using the nearest depth value ZNear and the farthest depth value ZFar.

The depth value for i-th MPI layer, Z[ i ], is derived as follows:

Z[ i ] = i \* ( ZFar − Znear ) ÷ (mpi\_num\_layers\_minus1 ) + ZNear (xx)

mpii\_layer\_depth\_equal\_distance\_flag equal to 0 indicates that the depth information for each layer follows next in the SEI message.

The variables in the x column of Table xx are derived from the respective variables in the s, e, n and v columns of Table xx as follows:

– If the value of e is in the range of 0 to 127, exclusive, x is set equal to ( − 1)s \* 2( e − 31 ) \* ( 1 + n ÷ 2v ).

– Otherwise (e is equal to 0), x is set equal to ( − 1)s \* 2−( 30 + v ) \* n.

NOTE 2 – The above specification is similar to that found in IEC 60559:1989.

**Table xx – Association between depth parameter variables and syntax elements**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **x** | **s** | **e** | **n** | **v** |
| ZNear | ZNearSign | ZNearExp | ZNearMantissa | ZNearManLen |
| ZFar | ZFarSign | ZFarExp | ZFarMantissa | ZFarManLen |
| Z[ i ] | ZSign[ i ] | ZExp | ZMantissa | ZManLen |

NOTE 3 – In some applications, disparity is used instead of depth (the disparity value D and depth value Z relationship is D = 1 ÷ Z). Correspoding to Equation (xx), the disparity value for the i-th MPI layer is D[ i ] = i \* ( DFar − Dnear ) ÷ (mpi\_num\_layers\_minus1 ) + DNear.

**mpii\_texture\_opacity\_interleave\_flag** equal to 1 indicates decoded output pictures correspond to temporally interleaved texture and opacity constituent pictures in output order as illustrated in Figure XX. mpii\_texture\_opacity\_interleave\_flag equal to 0 indicates decoded output pictures correspond to spatially packed texture and opacity constituent pictures as illustrated in Figures XX and XX.

**mpii\_texture\_opacity\_arrangement\_flag** equal to 0 indicates decoded output pictures represent texture and opacity constituent pictures in a top-bottom packing arrangement as illustrated in Figure XX. mpii\_texture\_opacity\_arrangement\_flagequal to 1 indicates decoded output pictures represent texture and opacity constituent pictures in a side-by-side packing arrangement as illustrated in Figure XX.

For each specified picture packing arrangement scheme, there are two constituent pictures that are referred to as picture 0 and picture 1. When mpii\_texture\_opacity\_interleave\_flag is equal to 0, the constituent picture associated with the upper-left sample of the decoded picture is considered to be constituent picture 0 and the other constituent picture is considered to be constituent picture 1. When mpii\_texture\_opacity\_interleave\_flag is equal to 1, the first decoded picture in the current CLVS is constituent picture 0 and the next decoded picture in output order is constituent picture 1 and the display time of the constituent picture 0 should be delayed to coincide with the display time of constituent picture 1. The two constituent pictures form the spatially packed texture and opacity picture of a MPI, with picture 0 being associated with the spatially packed texture picture and picture 1 being associated with the spatially packed opacity picture.

**mpii\_picture\_num\_layers\_in\_height\_minus1** plus 1 specifies the number of spatially packed layers in height for picture 0 and picture 1. The variable hLayers is set equal to mpii\_picture\_num\_layers\_in\_height\_minus1 + 1 and the variable wLayers is set equal to (mpii\_num\_layers\_minus1 + 1) / hLayers.

Let variable fWidth and fHeight specify the width and height of picture 0 and picture 1, respectively, and are derived as follows:

– If mpii\_texture\_opacity\_interleave\_flag is equal to 1, the following applies:

fWidth = CroppedWidth

fHeight = CroppedHeight

– Otherwise (mpii\_texture\_opacity\_interleave\_flag is equal to 0)

– If mpii\_texture\_opacity\_arrangement\_flag is equal to 0, the following applies:

fWidth = CroppedWidth , fHeight = CroppedHeight / 2

– Otherwise (mpii\_texture\_opacity\_arrangement\_flag is equal to 1), the following applies:

fWidth = CroppedWidth / 2 , fHeight = CroppedHeight

Let variable cWidth = fWidth / subWidthC and variable cHeight = fHeight / subHeightC.

Let array picture0[ cIdx ][ x ][ y ] specify samples in picture 0 and array picture1[ cIdx ][ x ][ y ] specify samples in picture 1, with cIdx = 0..(ChromaFormatIdc = = 0 ) ? 0 : 2, x = 0..( cIdx = = 0 ) ? fWidth: cWidth − 1, y = 0..( cIdx = = 0 ) ? fHeight : cHeight − 1 and are derived as follows:

– If mpii\_texture\_opacity\_interleave\_flag is equal to 1, the following applies:

picture0[ cIdx ][ x ][ y ] = decPicCurr0[ cIdx ][ x ][ y ]

picture1[ cIdx ][ x ][ y ] = decPicCurr1[ cIdx ][ x ][ y ]

– Otherwise (mpii\_texture\_opacity\_interleave\_flag is equal to 0)

– Let variable cW = ( cIdx = = 0 )? fWidth : cWidth

– Let variable cH = ( cIdx = = 0 )? fHeight : cHeight

– If mpii\_texture\_opacity\_arrangement\_flag is equal to 0, the following applies:

picture0[ cIdx ][ x ][ y ] = decPicCurr0[ cIdx ][ x ][ y ]

picture1[ cIdx ][ x ][ y ] = decPicCurr0[ cIdx ][ x ][ y + cH ]

– Otherwise (mpii\_texture\_opacity\_arrangement\_flag is equal to 1), the following applies:

picture0[ cIdx ][ x ][ y ] = decPicCurr0[ cIdx ][ x ][ y ]

picture1[ cIdx ][ x ][ y ] = decPicCurr0[ cIdx ][ x + cW ][ y ]

Let variable layerWidth and layerHeight specify the width and height for decoded MPI layer, respectively. The variables are derived as follows:

layerWidth = fWidth / wLayers

layerHeight = fHeight / hLayers

The reconstruction of MPI process is described as follows:

The outputs of this process are:

– a 4D MPI texture layer array recTextureLayer[ i ][ cIdx ][ w ][ h ] with i = 0..mpii\_num\_layers\_minus1, cIdx = 0..(ChromaFormatIdc = = 0 ) ? 0 : 2, w = 0..( cIdx = = 0 ) ? layerWidth : layerWidth / SubWidthC − 1, and h = 0..( cIdx = = 0 ) ? layerHeight : layerHeight / SubHeightC − 1.

– a 3D MPI opacity layer array recOpacityLayer[ i ][ w ][ h ] with i = 0..mpii\_num\_layers\_minus1, x = 0..layerWidth − 1, and y = 0..layerHeight − 1.

The array recTextureLayer and array recOpacityLayer are derived as follows:

for( i = 0; i  <=  mpii\_num\_layers\_minus1; i++ ) {  
 k = i % wLayers  
 m = ( i − k ) / hLayers  
 for( cIdx = 0; cIdx < ChromaFormatIdc = = 0 ) ? 1 : 3; cIdx++ )  
 for( h = 0; h < ( cIdx = = 0 ) ? layerHeight : layerHeight / SubHeightC ; h++ )  
 for( w = 0; w < ( cIdx = = 0 ) ? layerWidth : layerWidth / SubWidthC ; w++ ) {  
 u = k \* ( cIdx  = =  0 ) ? layerWidth : layerWidth / SubWidthC + w v = m \* ( cIdx  = =  0 ) ? layerHeight : layerHeight / SubHeightC + h  
 recTextureLayer[ i ][ cIdx ][ w ][ h ] = picture0[ cIdx ][ u ][ v ]  
 }  
 for( h = 0; h < layerHeight; h++ )  
 for( w = 0; w < layerWidth; w++ )  
 recOpacityLayer[ i ][ w ][ h ] = picture1[ 0 ][ k \* layerWidth + w ][ m \* layerHeight + h ]  
}

8.36 Generative face video SEI message

8.36.1 Generative face video SEI message syntax

|  |  |
| --- | --- |
| generative\_face\_video ( payloadSize ) { | **Descriptor** |
| **gfv\_id** | ue(v) |
| **gfv\_cnt** | ue(v) |
| if( gfv\_cnt  = =  0 ) |  |
| **gfv\_base\_pic\_flag** /\* indicate if the current decoded output picture is a base picture \*/ | u(1) |
| if( gfv\_base\_pic\_flag ) { /\* specify TranslatorNN( ) \*/ |  |
| **gfv\_nn\_present\_flag** | u(1) |
| if( gfv\_nn\_present\_flag ) { |  |
| **gfv\_nn\_base\_flag** | u(1) |
| **gfv\_nn\_mode\_idc** | ue(v) |
| if( gfv\_nn\_mode\_idc  = =  1 ) { |  |
| while( !byte\_aligned( ) ) |  |
| **gfv\_nn\_alignment\_zero\_bit\_a** | u(1) |
| **gfv\_nn\_tag\_uri** | st(v) |
| **gfv\_nn\_uri** | st(v) |
| } |  |
| } |  |
| **gfv\_chroma\_key\_info\_present\_flag** | u(1) |
| if( gfv\_chroma\_key\_info\_present\_flag ) { |  |
| for( c = 0; c < 3; c++ ) { |  |
| **gfv\_chroma\_key\_value\_present\_flag**[ c ] | u(1) |
| if( gfv\_chroma\_key\_value\_present\_flag[ c ] ) |  |
| **gfv\_chroma\_key\_value**[ c ] | u(8) |
| } |  |
| for( i = 0; i < 2; i++ ) { |  |
| **gfv\_chroma\_key\_thr\_present\_flag**[ i ] | u(1) |
| if( gfv\_chroma\_key\_thr\_present\_flag[ i ] ) |  |
| **gfv\_chroma\_key\_thr\_value**[ i ] | ue(v) |
| } |  |
| } |  |
| } else |  |
| **gfv\_drive\_pic\_fusion\_flag** /\* indicate if DrivePicture is input to GenerativeNN( ) \*/ | u(1) |
| **gfv\_low\_confidence\_face\_parameter\_flag** | u(1) |
| **gfv\_coordinate\_present\_flag** | u(1) |
| if( gfv\_coordinate\_present\_flag ) { |  |
| **gfv\_kps\_pred\_flag** | u(1) |
| if( gfv\_base\_pic\_flag  | |  !gfv\_kps\_pred\_flag ) { |  |
| **gfv\_coordinate\_precision\_factor\_minus1** | ue(v) |
| **gfv\_num\_kps\_minus1** | ue(v) |
| **gfv\_coordinate\_z\_present\_flag** | u(1) |
| if(gfv\_coordinate\_z\_present\_flag ) |  |
| **gfv\_coordinate\_z\_max\_value\_minus1** | ue(v) |
| } |  |
| for( i = 0; i  <=  gfv\_num\_kps\_minus1; i++ ) { |  |
| if( !gfv\_kps\_pred\_flag ) { |  |
| **gfv\_coordinate\_x\_abs**[ i ] | ue(v) |
| if( gfv\_coordinate\_x\_abs[ i ] > 0 ) |  |
| **gfv\_coordinate\_x\_sign\_flag**[ i ] | u(1) |
| **gfv\_coordinate\_y\_abs**[ i ] | ue(v) |
| if( gfv\_coordinate\_y\_abs[ i ] > 0 ) |  |
| **gfv\_coordinate\_y\_sign\_flag**[ i ] | u(1) |
| if( gfv\_coordinate\_z\_present\_flag > 0 ) { |  |
| **gfv\_coordinate\_z\_abs**[ i ] | ue(v) |
| if( gfv\_coordinate\_z\_abs[ i ] > 0 ) |  |
| **gfv\_coordinate\_z\_sign\_flag**[ i ] | u(1) |
| } |  |
| } else { |  |
| **gfv\_coordinate\_dx\_abs**[ i ] | ue(v) |
| if( gfv\_coordinate\_dx\_abs[ i ] > 0 ) |  |
| **gfv\_coordinate\_dx\_sign\_flag**[ i ] | u(1) |
| **gfv\_coordinate\_dy\_abs**[ i ] | ue(v) |
| if( gfv\_coordinate\_dy\_abs[ i ] > 0 ) |  |
| **gfv\_coordinate\_dy\_sign\_flag**[ i ] | u(1) |
| if( gfv\_coordinate\_z\_present\_flag ) { |  |
| **gfv\_coordinate\_dz\_abs**[ i ] | ue(v) |
| if( gfv\_coordinate\_dz\_abs[ i ] > 0 ) |  |
| **gfv\_coordinate\_dz\_sign\_flag**[ i ] | u(1) |
| } |  |
| } |  |
| } |  |
| } |  |
| **gfv\_matrix\_present\_flag** | u(1) |
| if(gfv\_matrix\_present\_flag ) { |  |
| if( !gfv\_base\_pic\_flag ) |  |
| **gfv\_matrix\_pred\_flag** | u(1) |
| if( gfv\_base\_pic\_flag  | |  !gfv\_matrix\_pred\_flag ) { |  |
| **gfv\_matrix\_element\_precision\_factor\_minus1** | ue(v) |
| **gfv\_num\_matrix\_types\_minus1** | ue(v) |
| for( i = 0; i  <=  num\_matrix\_types\_minus1; i++ ) { |  |
| **gfv\_matrix\_type\_idx**[ i ] | u(6) |
| if( gfv\_matrix\_type\_idx[ i ]  = =  0  | |  gfv\_matrix\_type\_idx[ i ]  = =  1 ) { |  |
| if( gfv\_coordinate\_present\_flag ) |  |
| **gfv\_num\_matrices\_equal\_to\_num\_kps\_flag**[ i ] | u(1) |
| if(!gfv\_num\_matrices\_equal\_to\_num\_kps\_flag[ i ] ) |  |
| **gfv\_num\_matrices\_info**[ i ] | ue(v) |
| } else if( gfv\_matrix\_type\_idx[ i ]  = = 2  | |  gfv\_matrix\_type\_idx[ i ]  = =  3  | |  gfv\_matrix\_type\_idx[ i ]  >=  7 ) { |  |
| if( gfv\_matrix\_type\_idx[ i ]  >=  7 ) |  |
| **gfv\_num\_matrices\_minus1**[ i ] | ue(v) |
| **gfv\_matrix\_width\_minus1**[ i ] | ue(v) |
| **gfv\_matrix\_height\_minus1**[ i ] | ue(v) |
| } else if( gfv\_matrix\_type\_idx[ i ]  >=  4  &&  gfv\_matrix\_type\_idx[ i ]  <=  6  &&  !gfv\_coordinate\_present\_flag ) |  |
| **gfv\_matrix\_for\_3D\_space\_flag**[ i ] | u(1) |
| } |  |
| } |  |
| for( i = 0; i  <=  num\_matrix\_types\_minus1; i++ ) { |  |
| for( j = 0; j < numMatrices[ i ]; j++ ) |  |
| for( k = 0; k < matrixHeight[ i ]; k++ ) |  |
| for( m = 0; m <matrixWidth[ i ]; m++ ) { |  |
| if( !gfv\_matrix\_pred\_flag ) { |  |
| **gfv\_matrix\_element\_int**[ i ][ j ][ k ][ m ] | ue(v) |
| **gfv\_matrix\_element\_dec**[ i ][ j ][ k ][ m ] | u(v) |
| if( gfv\_matrix\_element\_int[ i][ j ][ k ][ m ]  | |  gfv\_matrix\_element\_dec[ i ][ j ][ k ][ m ] ) |  |
| **gfv\_matrix\_element\_sign\_flag**[ i ][ j ][ k ][ m ] | u(1) |
| } else { |  |
| **gfv\_matrix\_delta\_element\_int**[ i ][ j ][ k ][ m ] | ue(v) |
| **gfv\_matrix\_delta\_element\_dec**[ i ][ j ][ k ][ m ] | ue(v) |
| if( gfv\_matrix\_delta\_element\_int[ i][ j ][ k ][ m ]  | |  gfv\_matrix\_delta\_element\_dec[ i ][ j ][ k ][ m ] ) |  |
| **gfv\_matrix\_delta\_element\_sign\_flag**[ i ][ j ][ k ][ m ] | u(1) |
| } |  |
| } |  |
| } |  |
| } |  |
| if( gfv\_nn\_present\_flag ) |  |
| if( gfv\_nn\_mode\_idc  = =  0 ) { |  |
| while( !byte\_aligned( ) ) |  |
| **gfv\_nn\_alignment\_zero\_bit\_b** | u(1) |
| for( i = 0; more\_data\_in\_payload( ); i++ ) |  |
| **gfv\_nn\_payload\_byte**[ i ] | b(8) |
| } |  |
| } |  |

8.36.2 Generative face video SEI message semantics

The generative face video (GFV) SEI message carries facial parameters and indicates a facial parameter translator network, denoted as TranslatorNN( ), that may be used to convert various formats of facial parameters signalled in the SEI message into a particular facial parameter format supported by the decoding system. A face picture generator neural network, denoted as GenerativeNN( ), may be used to generate output pictures using the facial parameters translated into the particular format and previously decoded output pictures.

When a picture unit contains a GFV SEI message with a particular gfv\_id value and gfv\_base\_pic\_flag equal to 1, the picture in the picture unit is referred to as a base picture for that particular gfv\_id value.

When a picture unit contains a GFV SEI message with a particular gfv\_id value and gfv\_base\_pic\_flag equal to 0, and the picture unit does not contain a GFV SEI message with that particular gfv\_id value and gfv\_base\_pic\_flag equal to 1, the picture in the picture unit is referred to as a driving picture for that particular gfv\_id value.

When a picture unit contains a GFV SEI message with a particular gfv\_id value, gfv\_base\_pic\_flag equal to 0, and gfv\_drive\_pic\_fusion\_flag equal to 1, and the picture unit does not contain a GFV SEI message with that particular gfv\_id value and gfv\_base\_pic\_flag equal to 1, the picture in the picture unit is referred to as a fusion picture for that particular gfv\_id value.

NOTE 1 – Facial parameters could be determined from source pictures prior to encoding.

NOTE 2 – Previously decoded output pictures input to GenerativeNN( ) may be a base picture (a decoded output picture that provides the reference texture from which the face pictures may be generated) and, optionally, a picture that can be fused by GenerativeNN( ) to improve background texture and facial details. When the current picture is not a base picture, the GFV SEI message may be used to generate a face picture based on the previously decoded base picture, the facial parameters conveyed by the GFV SEI message, and, optionally, the current decoded picture for fusion purpose.

Use of this SEI message requires the definition of the following variables:

* Input and output picture width and height in units of luma samples, denoted herein by CroppedWidth and CroppedHeight, respectively.
* Luma sample array baseCroppedYPic and chroma sample arrays baseCroppedCbPic and baseCroppedCrPic for a decoded output picture, denoted as BasePicture, corresponding to a source base picture.
* Luma sample array driveCroppedYPic and chroma sample arrays driveCroppedCbPic and driveCroppedCrPic for a decoded output picture, denoted as DrivePicture, corresponding to a source driving picture.
* Bit depth BitDepthY for the luma sample array of the input and output pictures.
* Bit depth BitDepthC for the chroma sample arrays, if any, of the input and output pictures.
* A chroma format indicator, denoted herein by ChromaFormatIdc, as described in subclause ‎7.3.

The variables SubWidthC and SubHeightC are derived from ChromaFormatIdc as specified by Table 2.

**gfv\_id** contains an identifying number that may be used to identify face feature information and specify a neural network that may be used as TranslatorNN( ). The value of gfv\_id shall be in the range of 0 to 232 − 2, inclusive. Values of gfv\_id from 256 to 511, inclusive, and from 231 to 232 − 2, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders conforming to this edition of this document encountering a GFV SEI message with gfv\_id in the range of 256 to 511, inclusive, or in the range of 231 to 232 − 2, inclusive, shall ignore the SEI message.

**gfv\_cnt** specifies a GFV SEI message instance count value for this gfv\_id value within a picture unit.

The gfv\_cnt of the first GFV SEI message, in decoding order, with a particular value of gfv\_id within a picture unit shall be equal to 0. When gfv\_cnt assigned to currGfvCnt is greater than 0, a GFV SEI message with the same gfv\_id value and gfv\_cnt equal to currGfvCnt − 1 shall be present in the same picture unit and precede the current GFV SEI message in decoding order.

The value of gfv\_cnt shall be in the range of 0 to 65 535, inclusive.

**gfv\_base\_pic\_flag** equal to 1 indicates that the current decoded output picture corresponds to a base picture. gfv\_base\_pic\_flag equal to 0 indicates that the current decoded output picture does not correspond to a base picture or this SEI message does not specify syntax elements for a base picture. When gfv\_base\_pic\_flag is not present, it is inferred to be equal to 0.

When a GFV SEI message is the first GFV SEI message, in decoding order, that has a particular gfv\_id value within the current CLVS, the value of gfv\_base\_pic\_flag shall be equal to 1.

When a GFV SEI message with a particular gfv\_id value has gfv\_base\_pic\_flag equal to 1, the base picture for that particular gfv\_id value, which is the current cropped decoded picture, remains valid for the current decoded picture and all subsequent decoded pictures of the current layer, in output order, until the end of the current CLVS or up to but excluding the decoded picture that is within the current CLVS, follows the current decoded picture in output order, and is associated with a GFV SEI message having that particular gfv\_id value and gfv\_base\_pic\_flag equal to 1, whichever is earlier.

**gfv\_nn\_present\_flag** equal to 1 indicates that a neural network that may be used as a TranslatorNN( ) is contained or indicated by the SEI message. gfv\_nn\_present\_flag equal to 0 indicates that a neural network that may be used as a TranslatorNN( ) is not contained or indicated by the SEI message. When gfv\_nn\_present\_flag is not present, it is inferred to be 0.

When a GFV SEI message with gfv\_cnt equal to 0 is present in the first picture unit in a CLVS in decoding order, gfv\_nn\_present\_flag shall be present and equal to 1.

When gfv\_nn\_present\_flag is equal to 0 and TranslatorNN is referenced in the semantics of the GFV SEI message, the following constraint applies:

– If gfv\_cnt is equal to 0, there shall be at least one GFV SEI message present in a preceding picture unit in output order in the current CLVS and having the same value of gfv\_id as that in the current GFV SEI message and gfv\_nn\_present\_flag equal to 1.

– Otherwise (gfv\_cnt is greater than 0), there shall be at least one GFV SEI message that is present in either the current picture unit or a preceding picture unit in output order in the current CLVS and has the same value of gfv\_id as that in the current GFV SEI message and gfv\_nn\_present\_flag equal to 1.

When gfv\_nn\_present\_flag is equal to 0 and TranslatorNN is referenced in the semantics of this SEI message, the following applies for deriving the applicable TranslatorNN:

– If gfv\_cnt is greater than 0 and there exists one or more preceding GFV SEI messages in decoding order in the current picture unit that has the same value of gfv\_id as that in the current GFV SEI message and gfv\_nn\_present\_flag equal to 1, the applicable TranslatorNN is defined by the last preceding GFV SEI message in decoding order in the current picture unit that has the same value of gfv\_id as that in the current GFV SEI message and gfv\_nn\_present\_flag equal to 1.

– Otherwise, the applicable TranslatorNN is defined by a GFV SEI message that is present in the last preceding picture unit puB in output order in the current CLVS that has the same value of gfv\_id as the current GFV SEI message and gfv\_nn\_present\_flag equal to 1. When there are multiple such GFV SEI messages present in the picture unit puB that have the same value of gfv\_id as the current GFV SEI message and gfv\_nn\_present\_flag equal to 1, the applicable TranslatorNN is defined by the last of such GFV SEI messages in decoding order.

**gfv\_nn\_base\_flag**, **gfv\_nn\_mode\_idc**, **gfv\_nn\_alignment\_zero\_bit\_a**, **gfv\_nn\_tag\_uri**, **gfv\_nn\_uri**, **gfv\_nn\_alignment\_zero\_bit\_b**, and **gfv\_nn\_payload\_byte**[ i ] specify a neural network that may be used as a TranslatorNN( ). gfv\_nn\_base\_flag, gfv\_nn\_mode\_idc, gfv\_nn\_alignment\_zero\_bit\_a, gfv\_nn\_tag\_uri, gfv\_nn\_uri, gfv\_nn\_alignment\_zero\_bit\_b, and gfv\_nn\_payload\_byte[ i ] have the same syntax and semantics as nnpfc\_base\_flag, nnpfc\_mode\_idc, nnpfc\_alignment\_zero\_bit\_a, nnpfc\_tag\_uri, nnpfc\_uri, nnpfc\_alignment\_zero\_bit\_b, and nnpfc\_payload\_byte[ i ], respectively.

When either of the following conditions is true, GFV SEI messages shall have the same SEI payload content:

– The GFV SEI messages are present in the same picture unit, have gfv\_cnt equal to 0, have gfv\_nn\_base\_flag present, and have the same value of gfv\_id and gfv\_nn\_base\_flag.

– The GFV SEI messages are present in the same picture unit, have the same value of gfv\_cnt that is greater than 0, and have the same value of gfv\_id.

**gfv\_chroma\_key\_info\_present\_flag** equal to 1 indicates that the syntax elements gfv\_chroma\_key\_value\_present\_flag[ c ] and gfv\_chroma\_key\_thr\_present\_flag[ i ] are present and the syntax elements and gfv\_chroma\_key\_value[ c ] and gfv\_chroma\_key\_thr\_value[ i ] might be present. gfv\_chroma\_key\_info\_present\_flag equal to 0 specifies that the syntax elements gfv\_chroma\_key\_value\_present\_flag[ c ], gfv\_chroma\_key\_thr\_present\_flag[ i ], gfv\_chroma\_key\_value[ c ], and gfv\_chroma\_key\_thr\_value[ i ] are not present.

**gfv\_chroma\_key\_value\_present\_flag**[ c ] equal to 1 indicates that the syntax element gfv\_chroma\_key\_value[ c ] is present. gfv\_chroma\_key\_present\_flag[ c ] equal to 0 indicates that the syntax element gfv\_chroma\_key\_value[ c ] is not present.

The variable ChromaKeyDefaultValueFlag is set equal to !( gfv\_chroma\_key\_value\_present\_flag[ 0 ] | | gfv\_chroma\_key\_value\_present\_flag[ 1 ] | | gfv\_chroma\_key\_value\_present\_flag[ 2 ] ).

**gfv\_chroma\_key\_value**[ c ] specifies the chroma key value corresponding to the c-th colour component as follows:

– If ChromaKeyDefaultValueFlag is equal to 1, the variables GfvChromaKeyValue[ c ] are specified as follows:

– GfvChromaKeyValue[ 0 ] is set equal to 50.

– GfvChromaKeyValue[ 1 ] is set equal to 220.

– GfvChromaKeyValue[ 2 ] is set equal to 100.

– Otherwise, ChromaKeyDefaultValueFlag is equal to 0, the following applies:

– If gfv\_chroma\_key\_value\_present\_flag[ c ] is equal to 1, GfvChromaKeyValue[ c ] is set equal to the value of gfv\_chroma\_key\_value[ c ].

– Otherwise, gfv\_chroma\_key\_value\_present\_flag[ c ] is equal to 0, GfvChromaKeyValue[ c ] is not specified by this Specification.

**gfv\_chroma\_key\_thr\_present\_flag**[ i ] equal to 1 indicates that the syntax element gfv\_chroma\_thr\_value[ i ] is present. gfv\_chroma\_key\_thr\_present\_flag[ i ] equal to 0 indicates gfv\_chroma\_key\_thr\_value[ i ] is not present.

**gfv\_chroma\_key\_thr\_value**[ i ], when present, specifies the i-th chroma key threshold value. When not present, the value of gfv\_chroma\_key\_thr\_value[ i ] is inferred as follows:

If i is equal to 0, gfv\_chroma\_key\_thr\_value[ 0 ] is set equal to 48.

Otherwise, i is equal to 1, gfv\_chroma\_key\_thr\_value[ 1 ] is set equal to 75.

NOTE 3 – The syntax elements gfv\_chroma\_key\_value\_present[ c ], gfv\_chroma\_key\_value[ c ], and gfv\_chroma\_key\_thr\_value[ i ] could be used to determine a transparency indicator for fusion of the generated face picture and background picture. For example, a transparency indicator, denoted as alpha[ x ][ y ] , for picture sample value, denoted as I[ c ][ x ][ y ] with bitDepth[ c ], where bitDepth[ 0 ] is equal to BitDepthY, bitDepth[ 1 ] and bitDepth[ 2 ] are equal to BitDepthC, and GfvChromaKey[ c ] values for sample coordinates x, y, and colour components c could be determined as follows:

d[ x ][ y ]= 0  
for( c = 0; c < 3; c++ )  
 if( gfv\_chroma\_key\_value\_present[ c ]  | |  ChromaKeyDefaultValueFlag )  
 d[ x ][ y ]  +=  ( I[ c ][ x ][ y ] / ( 1  <<  (bitDepth[ c ] − 8) ) − GfvChromaKeyValue[ c ] )2  
if( ( d[ x ][ y ] < gfv\_chroma\_key\_thr\_value[ 0 ] )  
 alpha[ x ][ y ] = 0  
else if ( ( d[ x ][ y ] > gfv\_chroma\_key\_thr\_value[ 1 ] )  
 alpha[ x ][ y ] = 1  
else  
 alpha[ x ][ y ] = ( d[ x ][ y ] − gfv\_chroma\_key\_thr\_value[ 0 ] ) ÷  
 ( gfv\_chroma\_key\_thr\_value[ 1 ] − gfv\_chroma\_key\_thr\_value[ 0 ] )

A value of alpha[ x ][ y ] equal to 0 could indicate transparency. A value of alpha[ x ][ y ] equal to 1 could indicate opacity. Intermediate values of alpha[ x ][ y ] could indicate semitransparency.

**gfv\_drive\_pic\_fusion\_flag**, when present, equal to 1 indicates that the current decoded picture, which corresponds to a driving picture that may be used for fusion, may be input to GenerativeNN( ). gfv\_drive\_pic\_fusion\_flag equal to 0 indicates that the current decoded picture should not be input to GenerativeNN( ).

NOTE 4 – A gfv\_drive\_pic\_fusion\_flag value of 1 can be used, for example, to indicate that the current decoded picture can be used to improve face details or handle background changes.

NOTE 5 – When gfv\_base\_pic\_flag is equal to 0 and gfv\_drive\_pic\_fusion\_flag is equal to 1, the GFV process takes three inputs: the base picture, features from keypoints and/or matrices carried in the GFV SEI message, and the current decoded picture that is a fusion picture, and outputs a picture that is generated by the GenerativeNN( ).

NOTE 6 – When gfv\_base\_pic\_flag is equal to 0 and gfv\_drive\_pic\_fusion\_flag is equal to 0, the GFV process takes twoinputs: the base picture and features from keypoints and/or matrices carried in the GFV SEI message, and outputs a picture that is generated by the GenerativeNN( ).

NOTE 7 – When gfv\_base\_pic\_flag is equal to 1, the GFV process directly outputs the cropped decoded picture.

When a GFV SEI message has gfv\_base\_pic\_flag equal to 0 and gfv\_drive\_pic\_fusion\_flag equal to 0, the GFV SEI message pertains to the current decoded picture only.

When a GFV SEI message with a particular gfv\_id value has gfv\_base\_pic\_flag equal to 0 and gfv\_drive\_pic\_fusion\_flag equal to 1, the fusion picture for that particular gfv\_id value, which is the current cropped decoded picture, remains valid for the current decoded picture and all subsequent decoded pictures of the current layer, in output order, until the end of the current CLVS or up to but excluding the decoded picture that is within the current CLVS, follows the current decoded picture in output order, and is associated with a GFV SEI message having that particular gfv\_id value, whichever is earlier.

When a GFV SEI message gfvSeiA with a particular gfv\_id value has gfv\_cnt greater than 0 and a GFV SEI message gfvSeiB with the same gfv\_id value in the same picture unit has gfv\_base\_pic\_flag equal to 1 (i.e., the current decoded picture is a base picture), the GFV SEI message gfvSeiA shall have gfv\_drive\_pic\_fusion\_flag equal to 0.

**gfv\_low\_confidence\_face\_parameter\_flag** equal to 1 indicates the facial parameters have been derived with low confidence. gfv\_low\_confidence\_face\_parameter\_flag equal to 0 indicates the confidence information of the facial parameters is not specified.

**gfv\_coordinate\_present\_flag** equal to 1 indicates that coordinate information of keypoints is present. gfv\_coordinate\_present\_flag equal to 0 indicates that coordinate information of keypoints is not present.

It is a requirement of bitstream conformance that when gfv\_matrix\_type\_idx[ i ] for any i from 0 to gfv\_num\_matrix\_types\_minus1 is equal to 0 or 1, the value of gfv\_coordinate\_present\_flag shall be equal to 1.

**gfv\_kps\_pred\_flag** equal to 1 indicates that the syntax elements gfv\_coordinate\_dx\_abs[ i ] ,gfv\_coordinate\_dy\_abs[ i ], and gfv\_coordinate\_dz\_abs[ i ] are present and the syntax elements gfv\_coordinate\_dx\_sign\_flag[ i ], gfv\_coordinate\_dy\_sign\_flag[ i ] and gfv\_coordinate\_dz\_sign\_flag[ i ] may be present. gfv\_kps\_pred\_flag equal to 0 indicates that the syntax elements gfv\_coordinate\_x\_abs[ i ], gfv\_coordinate\_y\_abs[ i ], and gfv\_coordinate\_z\_abs[ i ] are present and the syntax elements gfv\_coordinate\_x\_sign\_flag[ i ], gfv\_coordinate\_y\_sign\_flag[ i ] and gfv\_coordinate\_z\_sign\_flag[ i ] may be present.

When gfv\_coordinate\_present\_flag is equal to 1, gfv\_base\_pic\_flag is equal to 0, and gfv\_kps\_pred\_flag is equal to 1, there shall be a previous GFV SEI message in decoding order with the same gfv\_id as the current GFV SEI message and gfv\_base\_pic\_flag equal to 1 in the current CLVS.

**gfv\_coordinate\_precision\_factor\_minus1** plus 1 indicates the precision of key point coordinates signgalled in the SEI message. The value of gfv\_coordinate\_precision\_factor\_minus1 shall be in the range of 0 to 31, inclusive. When gfv\_coordinate\_present\_flag is equal to 1, gfv\_base\_pic\_flag is equal to 0, and gfv\_kps\_pred\_flag is equal to 1, the value of gfv\_coordinate\_precision\_factor\_minus1 is inferred to be equal to the gfv\_coordinate\_precision\_factor\_minus1 of the previous GFV SEI message in decoding order with the same gfv\_id as the current GFV SEI message and gfv\_base\_pic\_flag equal to 1.

**gfv\_num\_kps\_minus1** plus 1 indicates the number of keypoints. The value of gfv\_num\_kps\_minus1 shall be in the range of 0 to 210 – 1, inclusive. When gfv\_coordinate\_present\_flag is equal to 1, gfv\_base\_pic\_flag is equal to 0, and gfv\_kps\_pred\_flag is equal to 1, the value of gfv\_num\_kps\_minus1 is inferred to be equal to the gfv\_num\_kps\_minus1 of the previous GFV SEI message in decoding order with the same gfv\_id as the current GFV SEI message and gfv\_base\_pic\_flag equal to 1.

**gfv\_coordinate\_z\_present\_flag** equal to 1 indicates that z-axis coordinate information of the keypoints is present. gfv\_coordinate\_z\_present\_flagequal to 0 indicates that the z-axis coordinate information of the keypoints is not present. When gfv\_coordinate\_present\_flag is equal to 1, gfv\_base\_pic\_flag is equal to 0, and gfv\_kps\_pred\_flag is equal to 1, the value of coordinate\_z\_present\_flag is inferred to be equal to the coordinate\_z\_present\_flag of the previous GFV SEI message in decoding order with the same gfv\_id as the current GFV SEI message and gfv\_base\_pic\_flag equal to 1.

**gfv\_coordinate\_z\_max\_value\_minus1** plus 1 indicates the maximum absolute value of z-axis coordinates of keypoints. The value of gfv\_coordinate\_z\_max\_value\_minus1 shall be in the range of 0 to 216 − 1, inclusive. When gfv\_coordinate\_present\_flag is equal to 1, gfv\_base\_pic\_flag is equal to 0, and gfv\_kps\_pred\_flag is equal to 1, the value of gfv\_coordinate\_z\_max\_value\_minus1 is inferred to be equal to the gfv\_coordinate\_z\_max\_value\_minus1, when present, in the previous GFV SEI message in decoding order with the same gfv\_id as the current GFV SEI message and gfv\_base\_pic\_flag equal to 1.

**gfv\_coordinate\_x\_abs**[ i ] is used to derive the x-axis coordinate of the i-th keypoint.

**gfv\_coordinate\_x\_sign\_flag**[ i ] specifies the sign of the x-axis coordinate of the i-th keypoint. When gfv\_coordinate\_x\_sign\_flag[ i ] is not present, it is inferred to be equal to 0.

**gfv\_coordinate\_y\_abs**[ i ] is used to derive y-axis coordinate of i-th keypoint.

**gfv\_coordinate\_y\_sign\_flag**[ i ] specifies the sign of the y-axis coordinate of the i-th keypoint. When gfv\_coordinate\_y\_sign\_flag[i] is not present, it is inferred to be equal to 0.

**gfv\_coordinate\_z\_abs**[ i ] is used to derive z-axis coordinate of the i-th keypoint.

**gfv\_coordinate\_z\_sign\_flag**[ i ] specifies the sign of the z-axis coordinate of the i-th key point. When gfv\_coordinate\_z\_sign\_flag[ i ] is not present, it is inferred to be equal to 0.

**gfv\_coordinate\_dx\_abs**[ i ] specifies a difference value that is used to derive x-axis coordinate of the i-th keypoint.

**gfv\_coordinate\_dx\_sign\_flag**[ i ] specifies the sign of the difference value of the x-axis coordinate of the i-th keypoint. When gfv\_coordinate\_dx\_sign\_flag[ i ] is not present, it is inferred to be equal to 0.

**gfv\_coordinate\_dy\_abs**[ i ] specifies a difference value that is used to derive y-axis coordinate of the i-th keypoint.

**gfv\_coordinate\_dy\_sign\_flag**[ i ] specifies the sign of the difference value of the y-axis coordinate of the i-th keypoint. When gfv\_coordinate\_yd\_sign\_flag[i] is not present, it is inferred to be equal to 0.

**gfv\_coordinate\_dz\_abs**[ i ] specifies a difference value that is used to derive z-axis coordinate of the i-th keypoint.

**gfv\_coordinate\_dz\_sign\_flag**[ i ] specifies the sign of the difference value of the z-axis coordinate of the i-th key point. When gfv\_coordinate\_dz\_sign\_flag[ i ] is not present, it is inferred to be equal to 0.

If gfv\_coordinate\_z\_max\_value\_minus1 is present**,** the variable CroppedDepth is set equal to gfv\_coordinate\_z\_max\_value\_minus1 + 1. Otherwise, CroppedDepth is set equal to 0.

When gfv\_kps\_pred\_flag is equal to 1, the variables coordinateDeltaX[ i ], coordinateDeltaY[ i ] and coordinateDeltaZ[ i ] indicating the delta x-axis coordinate, delta y-axis coordinate and delta z-axis coordinate of the i-th keypoint, respectively, are derived as follows:

coordinateDeltaX[ i ] = ( 1 − 2 \* gfv\_coordinate\_dx\_sign\_flag[ i ] ) \* gfv\_coordinate\_dx\_abs[ i ] ÷  
 ( 1  <<  ( gfv\_coordinate\_precision\_factor\_minus1 + 1 ) )  
coordinateDeltaY[ i ] = ( 1 − 2 \* gfv\_coordinate\_dy\_sign\_flag[ i ] ) \* gfv\_coordinate\_dy\_abs[ i ] ÷  
 ( 1  <<  ( gfv\_coordinate\_precision\_factor\_minus1 + 1 ) )  
if( gfv\_coordinate\_z\_present\_flag )  
 coordinateDeltaZ[ i ] = ( 1 − 2 \* gfv\_coordinate\_dz\_sign\_flag[ i ] ) \* gfv\_coordinate\_dz\_abs[ i ] ÷  
 ( 1  <<  ( gfv\_coordinate\_precision\_factor\_minus1 + 1 ) )

The variables coordinateX[ i ], coordinateY[ i ], and, when gfv\_coordinate\_z\_present\_flag is equal to 1, coordinateZ[ i ] indicating the x-axis coordinate, y-axis coordinate and z-axis coordinate of the i-th keypoint, respectively, are derived as follows:

If gfv\_kps\_pred\_flag is equal to 0, the following applies:

coordinateX[ i ] = ( 1 − 2 \* gfv\_coordinate\_x\_sign\_flag[ i ] ) \* gfv\_coordinate\_x\_abs[ i ] ÷  
 ( 1  <<  ( gfv\_coordinate\_precision\_factor\_minus1 + 1 ) )  
coordinateY[ i ] = ( 1 − 2 \* gfv\_coordinate\_y\_sign\_flag[ i ] ) \* gfv\_coordinate\_y\_abs[ i ] ÷  
 ( 1  <<  ( gfv\_coordinate\_precision\_factor\_minus1 + 1 ) )  
if (gfv\_coordinate\_z\_present\_flag )  
 coordinateZ[ i ] = ( 1 − 2 \* gfv\_coordinate\_z\_sign\_flag[ i ] ) \* gfv\_coordinate\_z\_abs[ i ] ÷  
 ( 1  <<  ( gfv\_coordinate\_precision\_factor\_minus1 + 1 ) )

Otherwise (gfv\_kps\_pred\_flag is equal to 1), the following applies:

if( gfv\_base\_pic\_flag ) {  
 coordinateX[ i ] = (( i > 0 ) ? coordinateX[ i − 1 ] : 0 ) + coordinateDeltaX[ i ]  
 coordinateY[ i ] = (( i > 0 ) ? coordinateY[ i − 1 ] : 0 ) + coordinateDeltaY[ i ]  
 if (gfv\_coordinate\_z\_present\_flag )   
 coordinateZ[ i ] = (( i > 0 ) ? coordinateZ[ i − 1 ] : 0 ) + coordinateDeltaZ[ i ]  
} else if( gfv\_cnt  = =  0 ) {  
 coordinateX[ i ] = BaseKpCoordinateX[ i ] + coordinateDeltaX[ i ]  
 coordinateY[ i ] = BaseKpCoordinateY[ i ] + coordinateDeltaY[ i ]  
 if (gfv\_coordinate\_z\_present\_flag )   
 coordinateZ[ i ] = BaseKpCoordinateZ[ i ] + coordinateDeltaZ[ i ]  
} else {  
 coordinateX[ i ] = PrevKpCoordinateX[ i ] + coordinateDeltaX[ i ]  
 coordinateY[ i ] = PrevKpCoordinateY[ i ] + coordinateDeltaY[ i ]  
 coordinateZ[ i ] = PrevKpCoordinateZ[ i ] + coordinateDeltaZ[ i ]  
}

The following applies for derivation of the variables BaseKpCoordinateX[ i ], BaseKpCoordinateY[ i ], BaseKpCoordinateZ[ i ], PrevKpCoordinateX[ i ], PrevKpCoordinateY[ i ], and PrevKpCoordinateZ[ i ]:

if( gfv\_base\_pic\_flag ) {  
 PrevKpCoordinateX[ i ] = BaseKpCoordinateX[ i ] = coordinateX[ i ]  
 PrevKpCoordinateY[ i ] = BaseKpCoordinateY[ i ] = coordinateY[ i ]  
 if (gfv\_coordinate\_z\_present\_flag )  
 PrevKpCoordinateZ[ i ] = BaseKpCoordinateZ[ i ] = coordinateZ[ i ]  
} else {  
 PrevKpCoordinateX[ i ] = coordinateX[ i ]  
 PrevKpCoordinateY[ i ] = coordinateY[ i ]  
 PrevKpCoordinateZ[ i ] = coordinateZ[ i ]  
}

**gfv\_matrix\_present\_flag** equal to 1 indicates that matrix parameters are present. gfv\_matrix\_present\_flag equal to 0 indicates that matrix parameters are not present. When gfv\_coordinate\_present\_flag is equal to 0, gfv\_matrix\_present\_flag shall be equal to 1.

**gfv\_matrix\_pred\_flag** equal to 1 indicates that the syntax elements gfv\_matrix\_element\_int[ i ][ j ][ k ][ m ] and gfv\_matrix\_element\_dec[ i ][ j ][ k ][ m ] are present and the syntax element gfv\_matrix\_element\_sign\_flag[ i ][ j ][ k ][ m ] may be present. gfv\_matrix\_pred\_flag equal to 0 indicates that the syntax elements gfv\_matrix\_delta\_element\_int[ i ][ j ][ k ][ m ] and gfv\_matrix\_delta\_element\_dec[ i ][ j ][ k ][ m ] are present and the syntax element gfv\_matrix\_delta\_element\_sign\_flag [ i ][ j ][ k ][ m ] may be present. When gfv\_matrix\_pred\_flag is not present, it is inferred to be 0.

When gfv\_matrix\_present\_flag is equal to 1, gfv\_base\_pic\_flag is equal to 0, and gfv\_matrix\_pred\_flag is equal to 1, there shall be a previous GFV SEI message in decoding order with the same gfv\_id as the current GFV SEI message and gfv\_base\_pic\_flag equal to 1 in the current CLVS.

**gfv\_matrix\_element\_precision\_factor\_minus1** plus 1 indicates the precision of matrix elements signalled in the SEI message. The value of gfv\_matrix\_element\_precision\_factor\_minus1 shall be in the range of 0 to 31, inclusive. When gfv\_matrix\_present\_flag is equal to 1, gfv\_base\_pic\_flag is equal to 0, and gfv\_matrix\_pred\_flag is equal to 1, the value of gfv\_matrix\_element\_precision\_factor\_minus1 is inferred to be equal to the gfv\_matrix\_element\_precision\_factor\_minus1 of the previous GFV SEI message in decoding order with the same gfv\_id as the current GFV SEI message and gfv\_base\_pic\_flag equal to 1.

**gfv\_num\_matrix\_types\_minus1** plus 1indicates the number of matrix types signalled in the SEI message. The value of gfv\_num\_matrix\_types\_minus1 shall be in the range of 0 to 26 − 1, inclusive. It is a requirement of bitstream conformance that when gfv\_matrix\_pred\_flag is equal to 1 and gfv\_base\_pic\_flag is equal to 0, the value of gfv\_num\_matrix\_types\_minus1 shall be equal to the value of gfv\_num\_matrix\_types\_minus1 in each of the preceding GFV SEI message in decoding order in the current CLVS which has the same gfv\_id value as the gfv\_id value in the current SEI and has gfv\_base\_pic\_flag equal to 1. When gfv\_matrix\_present\_flag is equal to 1, gfv\_base\_pic\_flag is equal to 0, and gfv\_matrix\_pred\_flag is equal to 1, the value of gfv\_matrix\_type\_num\_minus1 is inferred to be equal to the gfv\_matrix\_type\_num\_minus1 of the previous GFV SEI message in decoding order with the same gfv\_id as the current GFV SEI message and gfv\_base\_pic\_flag equal to 1.

**gfv\_matrix\_type\_idx**[ i ]indicates the index of the i-th matrix type as specified in Table 3. The value of gfv\_matrix\_type\_idx[ i ] shall be in the range of 0 to 63, inclusive. In bitstreams conforming to this version of this Specification, the value of gfv\_matrix\_type\_idx[ i ] shall be in the range of 0 to 31, inclusive. Decoders conforming to this version of this Specification shall allow gfv\_matrix\_type\_idx[ i ] to be greater than 31 to appear in the bitstream and the decoder shall ignore all information for the i-th type of matrix for which gfv\_matrix\_type\_idx[ i ] is greater than 31.

**Table 3 - Specification of gfv\_matrix\_type\_idx**[ i ]

|  |  |
| --- | --- |
| **Value** | **Specification** |
| 0 | Affine translation matrix with the size of 2\*2 or 3\*3. |
| 1 | Covariance matrix with size of 2\*2 or 3\*3. |
| 2 | Mouth matrix representing mouth motion. |
| 3 | Eye matrix representing the open-close status and level of eyes. |
| 4 | Head rotation paramters with the size of 2\*2 or 3\*3 representing the head rotation in 2D space or 3D space. |
| 5 | Head translation matrix with the size of 1\*2 or 1\*3 representing head translationin 2D space or 3D space. |
| 6 | Head location matrix with size of 1\*2 or 1\*3 representing the head location in 2D space or 3D space. |
| 7 | Compact feature matrix with the size being specified by gfv\_matrix\_width\_minus1[i] and gfv\_matrix\_height\_minus1[i]. |
| 8…31 | Other matrix that may be used as determined by the application with the size being specified by gfv\_matrix\_width\_minus1[i] and gfv\_matrix\_height\_minus1[i]. |
| 32…63 | Reserved |

NOTE 8 – The undefined matrxi type is used to represent the matrxi type rather than affine translation matrix, covariance matrix, rotation matrix, translation matrix and compact feature matrix. It can be used by the user to extend the matrix type.

**gfv\_num\_matrices\_equal\_to\_num\_kps\_flag**[ i ] equal to 1 indicates that the number of matrices of the i-th matrix type is equal to gfv\_num\_kps\_minus1 + 1. gfv\_num\_matrices\_equal\_to\_num\_kps\_flag[ i ] equal to 0 indicates the number of matrices of the i-th matrix type is not equal to gfv\_num\_kps\_minus1 + 1. If gfv\_matrix\_present\_flag is equal to 1, gfv\_base\_pic\_flag is equal to 0, gfv\_matrix\_pred\_flag is equal to 1, gfv\_matrix\_type\_idx[ i ] is equal to 0 or 1, and gfv\_coordinate\_present\_flag is equal to 1, the value of gfv\_num\_matrices\_equal\_to\_num\_kps\_flag[ i ] is inferred to be equal to the gfv\_num\_matrices\_equal\_to\_num\_kps\_flag[ i ], when present, in the previous GFV SEI message in decoding order with the same gfv\_id as the current GFV SEI message and gfv\_base\_pic\_flag equal to 1. Otherwise, when gfv\_num\_matrices\_equal\_to\_num\_kps\_flag[ i ] is not present, its vlaue is inferred to be equal to 0.

**gfv\_num\_matrices\_info**[ i ] provides information to derive the number of the matrices of the i-th matrix type. The value of gfv\_num\_matrices\_info[ i ] shall be in the range of 0 to 210 − 1, inclusive. When gfv\_matrix\_present\_flag is equal to 1, gfv\_base\_pic\_flag is equal to 0, gfv\_matrix\_pred\_flag is equal to 1, gfv\_matrix\_type\_idx[ i ] is equal to 0 or 1, and either gfv\_coordinate\_present\_flag is equal to 0 or gfv\_num\_matrix\_equal\_to\_num\_kps\_flag[ i ] is equal to 0, the value of gfv\_num\_matrices\_info[ i ] is inferred to be equal to the gfv\_num\_matrices\_info[ i ], when present, in the previous GFV SEI message in decoding order with the same gfv\_id as the current GFV SEI message and gfv\_base\_pic\_flag equal to 1.

**gfv\_matrix\_width\_minus1**[ i ] plus 1 indicates the width of the matrix of the i-th matrix type. The value of gfv\_matrix\_width\_minus1[ i ] shall be in the range of 0 to 210 − 1, inclusive. When gfv\_matrix\_present\_flag is equal to 1, gfv\_matrix\_pred\_flag is equal to 0, gfv\_matrix\_pred\_flag is equal to 1, and gfv\_matrix\_type\_idx[ i ] is equal to 2 or 3 or is greater than or equal to 7, the value of gfv\_matrix\_width\_minus1[ i ] is inferred to be equal to the gfv\_matrix\_width\_minus1[ i ], when present, in the previous GFV SEI message in decoding order with the same gfv\_id as the current GFV SEI message and gfv\_base\_pic\_flag equal to 1.

**gfv\_matrix\_height\_minus1**[ i ] plus 1 indicates the height of the matrix of the i-th matrix type. The value of gfv\_matrix\_height\_minus1[ i ] shall be in the range of 0 to 210 − 1, inclusive. When gfv\_matrix\_present\_flag is equal to 1, gfv\_base\_pic\_flag is equal to 0, gfv\_matrix\_pred\_flag is equal to 1, and gfv\_matrix\_type\_idx[ i ] is equal to 2 or 3 or is greater than or equal to 7, the value of gfv\_matrix\_height\_minus1[ i ] is inferred to be equal to the gfv\_matrix\_height\_minus1[ i ], when present, in the previous GFV SEI message in decoding order with the same gfv\_id as the current GFV SEI message and gfv\_base\_pic\_flag equal to 1.

**gfv\_matrix\_for\_3D\_space\_flag**[ i ] equal to 1 indicates the matrix of the i-th matrix type is a matrix defined in three-dimensional space. gfv\_matrix\_for\_3D\_space\_flag[ i ] equal to 0 indicates the matrix of the i-th matrix type is a matrix defined in two-dimensional space. When gfv\_matrix\_present\_flag is equal to 1, gfv\_base\_pic\_flag is equal to 0, gfv\_matrix\_pred\_flag is equal to 1, gfv\_matrix\_type\_idx[ i ] is equal to 4, 5, or 6, and gfv\_coordinate\_present\_flag is equal to 0, the value of gfv\_matrix\_for\_3D\_space\_flag[ i ] is inferred to be equal to the gfv\_matrix\_for\_3D\_space\_flag[ i ], when present, in the previous GFV SEI message in decoding order with the same gfv\_id as the current GFV SEI message and gfv\_base\_pic\_flag equal to 1.

When gfv\_matrix\_width\_minus1[ i ] is not present, it is inferred as follows:

* If gfv\_matrix\_type\_idx[ i ] is equal to 0, 1 or 4, and one of coordinate\_z\_present\_flag and gfv\_matrix\_for\_3D\_space\_flag[ i ] is present and equal to 1, gfv\_matrix\_width\_minus1[i] is inferred to be equal to 2.
* Otherwise, if gfv\_matrix\_type\_idx[ i ] is equal to 0, 1 or 4, and one of coordinate\_z\_present\_flag and gfv\_matrix\_for\_3D\_space\_flag[ i ] is present and equal to 0, gfv\_matrix\_width\_minus1[ i ] is inferred to be equal to 1.
* Otherwise (gfv\_matrix\_type\_idx[ i ] is equal to 5 or 6), gfv\_matrix\_width\_minus1[ i ] is inferred to be equal to 0.

When gfv\_matrix\_height\_minus1[ i ] is not present, it is inferred as follows:

* If gfv\_matrix\_type\_idx is equal to 0, 1, 4, 5 or 6, and one of gfv\_coordinate\_z\_present\_flag and gfv\_matrix\_for\_3D\_space\_flag[ i ] is present and equal to 1, gfv\_matrix\_height\_minus1[ i ] is inferred to be equal to 2.
* Otherwise (gfv\_matrix\_type\_idx is equal to 0, 1, 4, 5 or 6, and one of gfv\_coordinate\_z\_present\_flag and gfv\_matrix\_for\_3D\_space\_flag[ i ] is 0), gfv\_matrix\_height \_minus1[ i ] is inferred to be equal to 1.

The variables matrixWidth[ i ] and matrixHeight[ i ] indicating the width and height of the matrix of the i-th matrix type are derived as follows:

if( gfv\_matrix\_pred\_flag ) {  
 matrixWidth[ i ] = BaseMatrixWidth[ i ]  
 matrixHeight[ i ] = BaseMatrixHeight[ i ]  
} else {  
 matrixWidth[ i ] = gfv\_matrix\_width\_minus1[ i ] + 1  
 matrixHeight[ i ] = gfv\_matrix\_height\_minus1[ i ] + 1  
}  
if( gfv\_base\_pic\_flag ) {  
 BaseMatrixWidth[ i ] = matrixWidth[ i ]  
 BaseMatrixHeight[ i ] = matrixHeight[ i ]  
}

**gfv\_num\_matrices\_minus1**[ i ] plus 1 indicates the number of matrices of the i-th matrix type. The value of gfv\_num\_matrices\_minus1[ i ] shall be in the range of 0 to 210 − 1, inclusive. When gfv\_matrix\_present\_flag is equal to 1, gfv\_base\_pic\_flag is equal to 0, gfv\_matrix\_pred\_flag is equal to 1, and gfv\_matrix\_type\_idx[ i ] is greater than or equal to 7, the value of gfv\_num\_matrices\_minus1[ i ] is inferred to be equal to the gfv\_num\_matrices\_minus1[ i ], when present, in the previous GFV SEI message in decoding order with the same gfv\_id as the current GFV SEI message and gfv\_base\_pic\_flag equal to 1.

The variable numMatrices[ i ] indicating the number of the matrices of the i-th matrix type is derived as follows:

if( gfv\_matrix\_pred\_flag )  
 numMatrices[ i ] = BaseNumMatrices[ i ]  
else if( gfv\_matrix\_type\_idx[ i ]  = =  0  | |  gfv\_matrix\_type\_idx[ i ]  = =  1 ) {  
 if( gfv\_coordinate\_present\_flag )  
 numMatrices[ i ] = gfv\_num\_matrices\_equal\_to\_num\_kps\_flag[ i ] ? gfv\_num\_kps\_minus1 + 1:  
 ( gfv\_num\_matrices\_info[ i ] < gfv\_num\_kps\_minus1 ? gfv\_num\_matrices\_info [ i ] + 1:  
 gfv\_num\_matrices\_info [ i ] + 2 )  
 else  
 numMatrices[ i ] = gfv\_num\_matrices\_info[ i ] + 1  
} else if( gfv\_matrix\_type\_idx[ i ]  >=  2  &&  gfv\_matrix\_type\_idx[ i ] < 7 )  
 numMatrices[ i ] = 1  
else  
 numMatrices[ i ] = gfv\_num\_matrices\_minus1[ i ] + 1  
if( gfv\_base\_pic\_flag )  
 BaseNumMatrices[ i ] = numMatrices[ i ]

It is a requirement of bitstream conformance that when gfv\_matrix\_pred\_flag is equal to 1 and gfv\_base\_pic\_flag is equal to 0, the values of numMatrices[ i ], matrixWidth[ i ], and matrixHeight[ i ] for i in the range of 0 to gfv\_num\_matrix\_types\_minus1, inclusive shall be respectively equal to the values of numMatrices[ i ], matrixWidth[ i ], and matrixHeight[ i ] for i in the range of 0 to gfv\_num\_matrix\_types\_minus1, inclusive in each of the preceding GFV SEI message in decoding order in the current CLVS which has the same gfv\_id value as the gfv\_id value in the current SEI and has gfv\_base\_pic\_flag equal to 1.

**gfv\_matrix\_element\_int**[ i ][ j ][ k ][ m ] indicates the integer part of the value of the matrix element at position (m, k) of the j-th matrix of the i-th matrix type. The value of gfv\_matrix\_element\_int[ i ][ j ][ k ][ m ] shall be in the range of 0 to 232 − 2, inclusive.

**gfv\_matrix\_element\_dec**[ i ][ j ][ k ][ m ] indicates the decimal part of the value of the matrix element at position (m, k) of the j-th matrix of the i-th matrix type. The length of gfv\_matrix\_element\_dec[ i ][ j ][ k ][ m ] is gfv\_matrix\_element\_precision\_factor\_minus1 + 1 bits.

**gfv\_matrix\_element\_sign\_flag**[ i ][ j ][ k ][ m ] indicates the sign of the matrix element at position (m, k) of the j-th matrix of the i-th matrix type. When gfv\_matrix\_element\_sign\_flag[ i ][ j ][ k ][ m ]is not present, it is inferred to be equal to 0.

**gfv\_matrix\_delta\_element\_int**[ i ][ j ][ k ][ m ] indicates the integer part of the difference value of the matrix element at position (m, k) of the j-th matrix of the i-th matrix type.

**gfv\_matrix\_delta\_element\_dec**[ i ][ j ][ k ][ m ] indicates the decimal part of the difference value of the matrix element at position (m, k) of the j-th matrix of the i-th matrix type.

**gfv\_matrix\_delta\_element\_sign\_flag**[ i ][ j ][ k ][ m ] indicates the sign of the difference value of the matrix element at position (m, k) of the j-th matrix of the i-th matrix type. When gfv\_matrix\_element\_sign\_flag[ i ][ j ][ k ][ m ]is not present, it is inferred to be equal to 0.

When gfv\_matrix\_pred\_flag is equal to 1, the variable matrixElementDeltaVal[ i ][ j][ k ][ m ] representing the difference value of the matrix element at position (m, k) of the j-th matrix of the i-th matrix type is derived as follows:

matrixElementDeltaVal[ i][ j ][ k ][ m ] =   
The variable matrixElementVal[ i ][ j][ k ][ m ] representing the value of the matrix element at position (m, k) of the j-th matrix of the i-th matrix type is derived as follows:

If gfv\_matrix\_pred\_flag is equal to 0, the following applies:

matrixElementVal[ i][ j ][ k ][ m ] =   
  
if( gfv\_base\_pic\_flag )  
 BaseMatrixElementVal[ i][ j ][ k ][ m ] = matrixElementVal[ i][ j ][ k ][ m ]

Otherwise (gfv\_matrix\_pred\_flag is equal to 1), the following applies:

if( gfv\_cnt  = =  0 )  
 matrixElementVal[ i][ j ][ k ][ m ] = BaseMatrixElementVal[ i][ j ][ k ][ m ] +  
 matrixElementDeltaVal[ i][ j ][ k ][ m ]  
else  
 matrixElementVal[ i][ j ][ k ][ m ] = PrevMatrixElementVal[ i][ j ][ k ][ m ] +  
 matrixElementDeltaVal[ i][ j ][ k ][ m ]

The following applies:

if( gfv\_base\_pic\_flag )  
 PrevMatrixElementVal[ i][ j ][ k ][ m ] = BaseMatrixElementVal[ i][ j ][ k ][ m ] =  
 matrixElementVal[ i][ j ][ k ][ m ]  
else  
 PrevMatrixElementVal[ i][ j ][ k ][ m ] = matrixElementVal[ i][ j ][ k ][ m ]

For a particular gfv\_id value, the following process is used in increasing order of gfv\_cnt to generate a video picture per each GFV SEI message that has gfv\_base\_pic\_flag equal to 0 and a unique value of gfv\_cnt within a picture unit:

DeriveSigParam( )  
TranslatorNN( sigKeyPoint, sigMatrix )  
DeriveInputTensors( )  
if( gfv\_base\_pic\_flag  = =  0  &&  gfv\_drive\_pic\_fusion\_flag  = =  0 ) {  
 if( ChromaFormatIdc  = =  0 )  
 GenerativeNN( inputBaseY, inputBaseKeyPoint, inputBaseMatrix, inputDriveKeyPoint,  
 inputDriveMatrix, CroppedWidth, CroppedHeight, CroppedDepth )  
 else  
 GenerativeNN( inputBaseY, inputBaseCb, inputBaseCr, inputBaseKeyPoint, inputBaseMatrix,  
 inputDriveKeyPoint, inputDriveMatrix, CroppedWidth, CroppedHeight, CroppedDepth )  
} else if( gfv\_base\_pic\_flag  = =  0  &&  gfv\_drive\_pic\_fusion\_flag  = =  1 ) {  
 if( ChromaFormatIdc  = =  0 )  
 GenerativeNN( inputBaseY, inputDriveY, inputBaseKeyPoint, inputBaseMatrix, inputDriveKeyPoint,  
 inputDriveMatrix, CroppedWidth, CroppedHeight, CroppedDepth )  
 else  
 GenerativeNN( inputBaseY, inputBaseCb, inputBaseCr, inputDriveY, inputDriveCb, inputDriveCr,  
 inputBaseKeyPoint, inputBaseMatrix,, inputDriveKeyPoint, inputDriveMatrix,  
 CroppedWidth, CroppedHeight, CroppedDepth )  
}  
StoreOutputTensors( )

The process DeriveSigParam( ) for deriving the inputs of TranslatorNN( ) is specified as follows:

The keypoint coordinate array sigKeyPoint and the matrix sigMatrix are derived as follows:

if( gfv\_coordinate\_present\_flag )  
 for( i = 0; i  <=  gfv\_num\_kps\_minus1; i++ ) {  
 sigKeyPoint[ i ][ 0 ] = coordinateX[ i ]  
 sigKeyPoint[ i ][ 1 ] = coordinateY[ i ]  
 if( gfv\_coordinate\_z\_present\_flag )  
 sigKeyPoint[ i ][ 2 ] = coordinateZ[ i ]  
 }  
else  
 for( i = 0; i  <=  gfv\_num\_kps\_minus1; i++ ) {  
 sigKeyPoint[ i ][ 0 ] = 0  
 sigKeyPoint[ i ][ 1 ] = 0  
 if ( gfv\_coordinate\_z\_present\_flag )  
 sigKeyPoint[ i ][ 2 ] = 0  
 }  
if( gfv\_matrix\_present\_flag )  
 for ( i = 0; i  <=  gfv\_num\_matrix\_types\_minus1; i++ )  
 for ( j = 0; j < numMatrices[ i ]; j++ )  
 for( k = 0; k < matrixHeight [ i ]; k++ )  
 for( l = 0;l < matrixWidth [ i ]; l++)  
 sigMatrix[ i ][ j ][ k ][ l ] = matrixElementVal[ i ][ j][ k][ l ]  
else  
 for( i = 0; i  <=  gfv\_num\_matrix\_types\_minus1; i++ )  
 for ( j = 0; j < numMatrices[ i ]; j++ )  
 for( k = 0; k < matrixHeight [ i ]; k++ )  
 for( l = 0;l < matrixWidth [ i ]; l++)  
 sigMatrix[ i ][ j ][ k ][ l ] = 0

TranslatorNN( ) is a process to translate the various formats of the facial parameters carried in the SEI message to the fixed format of the facial parameters to be input to the generative network to generate the output picture

Inputs to TranslatorNN( ) are:

* sigKeyPoint and sigMatrix

Outputs of TranslatorNN( ) are:

* convKeyPoint and convNumKeyPoint
* convMatrix and convNumMatrix, convMatrixWidth, convMatrixHeight

The process DeriveInputTensors( ) for deriving the inputs of GenerativeNN( ) is specified as follows:

When gfv\_base\_pic\_flag is equal to 1, the BasePicture input tensor inputBaseY, inputBaseCb and inputBaseCr are derived as follows:

for( x = 0; x < CroppedWidth; x++ )  
 for ( y = 0; y < CroppedHeight; y++ )  
 inputBaseY[ x ][ y ] = InpY( baseCroppedYPic[ x ][ y ] )  
if( ChromaFormatIdc  !=  0 )  
 for( x = 0; x < CroppedWidth / SubWidthC; x++ )  
 for ( y = 0; y < CroppedHeight / SubHeightC; y++ ) {  
 inputBaseCb[ x][ y ] = InpC( baseCroppedCbPic[ x][ y ] )  
 inputBaseCr[ x][ y ] = InpC( baseCroppedCrPic[ x ][ y ] )  
 }

When gfv\_base\_pic\_flag is equal to 0 and gfv\_drive\_pic\_fusion\_flag is equal to 1, the DrivePicture luma sample array inputDriveY, inputDriveCb and input DriveCr are derived as follows:

for( x = 0; x< CroppedWidth; x++ )  
 for ( y = 0; y< CroppedHeight; y++ )  
 inputDriveY[ x ][ y ] = InpY( driveCroppedYPic[ x ][ y ] )  
if( ChromaFormatIdc  !=  0 )  
 for( x = 0; x< CroppedWidth / SubWidthC; x++ )  
 for( y = 0; y < CroppedHeight / SubHeightC; y++ ) {  
 InputDriveCb[ x][ y ] = InpC( driveCroppedCbPic[ x][ y ] )  
 InputDriveCr[ x][ y ] = InpC( driveCroppedCrPic[ x ][ y ] )  
 }

When gfv\_base\_pic\_flag is equal to 0, the keypoint coordinate array inputDriveKeyPoint and the matrix inputDriveMatrix for the current picture are derived as follows:

for( i = 0; i < = convNumKeyPoint; i++ ) {  
 inputDriveKeyPoint[ i ][ 0 ] = convKeyPoint[ i ][ 0 ]  
 inputDriveKeyPoint [ i ][ 1 ] = convKeyPoint[ i ][ 1 ]  
 inputDriveKeyPoint [ i ][ 2 ] = convKeyPoint[ i ][ 2 ]  
}  
for( j = 0; j < convNumMatrix; j++ )  
 for( k = 0; k < convMatrixHeight; k++ )  
 for( m = 0; m < convMatrixWidth; m++ )  
 inputDriveMatrix[ j ][ k ][ m ] = convMatrix [ j ][ k ][ m ]

When gfv\_base\_pic\_flag is equal to 1, the keypoint coordinate array inputBaseKeyPoint and the matrix inputBaseMatrix for the base picture are derived as follows:

for( i = 0; i  <=  convNumKeyPoint; i++ ) {  
 inputBaseKeyPoint[ i ][ 0 ] = convKeyPoint[ i ][ 0 ]  
 inputBaseKeyPoint [ i ][ 1 ] = convKeyPoint[ i ][ 1 ]  
 inputBaseKeyPoint [ i ][ 2 ] = convKeyPoint[ i ][ 2 ]  
}  
for( j = 0; j < convNumMatrix; j++ )  
 for( k = 0; k < convMatrixHeight; k++ )  
 for( l = 0; l < convMatrixWidth; l++ )  
 inputBaseMatrix[ j ][ k ][ l ] = convMatrix [ j ][ k ][ l ]

Where the functions InpY( ) and InpC( ) are specified as follows:

InpY( x ) = x ÷ ( ( 1  <<  BitDepthY ) − 1 )  
InpC( x ) = x ÷ ( ( 1  <<  BitDepthC ) − 1 )

GenerativeNN( ) is a process to generate the sample values of an output picture corresponding to a driving picture. It is only invoked when gfc\_base\_pic\_flag is equal to 0. Input values to GenerativeNN( ) and output values from GenerativeNN( ) are real numbers.

Inputs to GenerativeNN( ) are:

* When gfv\_base\_pic\_flag is equal to 0, gfv\_drive\_pic\_fusion\_flag is equal to 0, and ChromaFormatIdc is equal to 0: inputBaseY, inputBaseKeyPoint, inputBaseMatrix, inputDriveKeyPoint, inputDriveMatrix, CroppedWidth, CroppedHeight,and CroppedDepth.
* When gfv\_base\_pic\_flag is equal to 0, gfv\_drive\_pic\_fusion\_flag is equal to 0, and ChromaFormatIdc is not equal to 0: inputBaseY, inputBaseCb, inputBaseCr, inputBaseKeyPoint, inputBaseMatrix, inputDriveKeyPoint, inputDriveMatrix, CroppedWidth, CroppedHeight, and CroppedDepth.
* When gfv\_base\_pic\_flag is equal to 0, gfv\_drive\_pic\_fusion\_flag is equal to 1, gfv\_chroma\_key\_info\_present\_flag is equal to 0, and ChromaFormatIdc is equal to 0: inputBaseY, inputDriveY, inputBaseKeyPoint, inputBaseMatrix, inputDriveKeyPoint, inputDriveMatrix, CroppedWidth, CroppedHeight, and CroppedDepth.
* When gfv\_base\_pic\_flag is equal to 0, gfv\_drive\_pic\_fusion\_flag is equal to 1, gfv\_chroma\_key\_info\_present\_flag is equal to 0, and ChromaFormatIdc is not equal to 0: inputBaseY, inputBaseCb, inputBaseCr, inputDriveY, inputDriveCb, inputDriveCr , inputBaseKeyPoint, inputBaseMatrix,, inputDriveKeyPoint, inputDriveMatrix, CroppedWidth, CroppedHeight, and CroppedDepth.
* When gfv\_base\_pic\_flag is equal to 0, gfv\_drive\_pic\_fusion\_flag is equal to 1, gfv\_chroma\_key\_info\_present\_flag is equal to 1, and ChromaFormatIdc is equal to 0: inputBaseY, inputDriveY, inputBaseKeyPoint, inputBaseMatrix, inputDriveKeyPoint, inputDriveMatrix, CroppedWidth, CroppedHeight, CroppedDepth, gfv\_chroma\_key\_thr\_value[ 0 ], gfv\_chroma\_key\_thr\_value[ 1 ], and GfvChromaKeyValue[ 0 ].
* When gfv\_base\_pic\_flag is equal to 0, gfv\_drive\_pic\_fusion\_flag is equal to 1, gfv\_chroma\_key\_info\_present\_flag is equal to 0, and ChromaFormatIdc is not equal to 0: inputBaseY, inputBaseCb, inputBaseCr, inputDriveY, inputDriveCb, inputDriveCr , inputBaseKeyPoint, inputBaseMatrix, inputDriveKeyPoint, inputDriveMatrix, CroppedWidth, CroppedHeight, CroppedDepth, gfv\_chroma\_key\_thr\_value[ 0 ], gfv\_chroma\_key\_thr\_value[ 1 ], and, when specified, GfvChromaKeyValue[ 0 ], GfvChromaKeyValue[ 1 ], and GfvChromaKeyValue[ 2 ].

Outputs of GenerativeNN( ) are:

* A luma sample array genY
* When ChromaFormatIdc is not equal to 0, two chroma sample arrays genCb and genCr.

The process StoreOutputTensors( ) for deriving the output is specified as follows:

When gfv\_base\_pic\_flag is equal to 0, the output sample array outYPic[ x ][ y ], outCbPic[ x ][ y ], and outCrPic[ x ][ y ] are derived as follows:

for( x = 0; x < CroppedWidth; x++ )  
 for( y = 0; y < CroppedHeight; y++ )  
 outputYPic[ x ][ y ] = OutY( genY[ x ][ y ] )  
if( ChromaFormatIdc  !=  0 )  
 for( x = 0; x < CroppedWidth / SubWidthC; x++ )  
 for( y = 0; y < CroppedHeight / SubHeightC; y++ ) {  
 outputCbPic[ x ][ y ] = OutC( genCb[ x ][ y ] )  
 outputCrPic[ x][ y ] = OutC( genCr[ x ][ y ] )  
 }

When gfv\_base\_pic\_flag is equal to 1, the output sample array outYPic[ x ][ y ], outCbPic[ x ][ y ], and outCrPic[ x ][ y ] are derived as follows (each output picture derived by the process StoreOutputTensors( ) is referred to as a GFV-generated picture):

for( x = 0; x< CroppedWidth; x++ )  
 for( y = 0; y< CroppedHeight; y++ )  
 outputYPic[ x ][ y ] = baseCroppedYPic[ x ][ y ]  
if( ChromaFormatIdc  !=  0 )  
 for( x = 0; x< CroppedWidth / SubWidthC; x++ )  
 for( y = 0; y< CroppedHeight / SubHeightC; y++ ) {  
 outputCbPic[ x ][ y ] = baseCroppedCbPic[ x ][ y ]  
 outputCrPic[ x][ y ] = baseCroppedCbPic[ x ][ y ]  
 }

Where the functions OutY( ) and OutC( ) are specified as follows:

OutY( x ) = Clip3( 0, ( 1  <<  BitDepthY ) − 1 , x \* ( ( 1  <<  BitDepthY ) − 1 )  
OutC( x ) = Clip3( 0, ( 1  <<  BitDepthC ) − 1 , x \* ( ( 1  <<  BitDepthC ) − 1 )

The output order of GFV-generated pictures corresponding to the GFV SEI messages in a picture unit with the same gfv\_id value and different gfv\_cnt values shall be in increasing order of the gfv\_cnt values. For any two pictures picA and picB wherein picA precedes picB in output order, any GFV-generated picture corresponding to a GFV SEI message with a particular gfv\_id value and associated with picA shall precede, in output order, any GFV-generated picture corresponding to a GFV SEI message with the particular gfv\_id value and associated with picB.

8.37 Generative face video enhancement SEI message

8.37.1 Generative face video enhancement SEI message syntax

|  |  |
| --- | --- |
| generative\_face\_video\_enhancement ( payloadSize ) { | **Descriptor** |
| **gfve\_id** | ue(v) |
| **gfve\_gfv\_id** | ue(v) |
| **gfve\_gfv\_cnt** | ue(v) |
| **gfve\_nn\_present\_flag** | u(1) |
| if( gfve\_nn\_present\_flag ) { |  |
| **gfve\_nn\_base\_flag** | u(1) |
| **gfve\_nn\_mode\_idc** | ue(v) |
| if( gfve\_nn\_mode\_idc  = =  1 ) { |  |
| while( !byte\_aligned( ) ) |  |
| **gfve\_nn\_alignment\_zero\_bit\_a** | u(1) |
| **gfve\_nn\_tag\_uri** | st(v) |
| **gfve\_nn\_uri** | st(v) |
| } |  |
| } |  |
| **gfve\_matrix\_element\_precision\_factor** | ue(v) |
| **gfve\_num\_matrices\_minus1** | ue(v) |
| for(i=0; i <= gfve\_num\_matrices\_minus1; i++){ |  |
| **gfve\_matrix\_height\_minus1**[ i ] | ue(v) |
| **gfve\_matrix\_width\_minus1**[ i ] | ue(v) |
| for( j = 0; j <= gfve\_matrix\_height\_minus1[ i ]; j++ ) |  |
| for( k = 0; k <= gfve\_matrix\_width\_minus1[ i ]; k++ ) { |  |
| **gfve\_matrix\_element**[ i ][ j ][ k ] | ue(v) |
| if( !gfve\_matrix\_element[ i ][ j ][ k ]) |  |
| **gfve\_matrix\_element\_sign\_flag**[ i ][ j ][ k ] | u(1) |
| } |  |
| } |  |
| if( gfve\_nn\_present\_flag ) |  |
| if( gfve\_nn\_mode\_idc  = =  0 ) { |  |
| while( !byte\_aligned( ) ) |  |
| **gfve\_nn\_alignment\_zero\_bit\_b** | u(1) |
| for( i = 0; more\_data\_in\_payload( ); i++ ) |  |
| **gfve\_nn\_payload\_byte**[ i ] | b(8) |
| } |  |
| } |  |

8.37.2 Generative face video enhancement SEI message semantics

The generative face video enhancement (GFVE) SEI message indicates enhancement facial parameters and specifies an enhancement network, denoted as EnhancerNN( ), that may be used to enhance the visual quality of the face pictures generated with GFV SEI message.

NOTE 1 – Enhancement facial parameters could be determined from source pictures prior to encoding.

NOTE 2 – When the current picture is not a base picture, the GFV SEI message may be used to generate a face picture based the facial parameters conveyed by the GFV SEI message, and the GFVE SEI message may be further used to enhance the generated face picture to improve the visual quality.

Use of this SEI message requires the definition of the following variables:

* Input and output picture width and height in units of luma samples, denoted herein by CroppedWidth and CroppedHeight, respectively.
* Luma sample array baseCroppedYPic and chroma sample arrays baseCroppedCbPic and baseCroppedCrPic for a decoded output picture, denoted as BasePicture, corresponding to a base picture.
* Luma sample array genCroppedYPic and chroma sample arrays genCroppedCbPic and genCroppedCrPic for a generated picture with associated GFV SEI message, denoted as GenPicture, corresponding to a driving picture.
* Bit depth BitDepthY for the luma sample array of the input and output pictures.
* Bit depth BitDepthC for the chroma sample arrays, if any, of the input and output pictures.
* A chroma format indicator, denoted herein by ChromaFormatIdc, as described in subclause ‎7.3.

The variables SubWidthC and SubHeightC are derived from ChromaFormatIdc as specified by Table 2.

**gfve\_id** contains an identifying number that may be used to identify GFVE SEI message and specify a neural network that may be used as EnhancerNN( ). The value of gfve\_id shall be in the range of 0 to 232 − 2, inclusive. Values of gfve\_id from 256 to 511, inclusive, and from 231 to 232 − 2, inclusive, are reserved for future use by ITU-T | ISO/IEC. Decoders conforming to this edition of this document encountering a GFVE SEI message with gfve\_id in the range of 256 to 511, inclusive, or in the range of 231 to 232 − 2, inclusive, shall ignore the SEI message.

**gfve\_gfv\_id** and **gfve\_gfv\_cnt** specifies gfv\_id and gfv\_cnt of the associated GFV SEI message. The associated GFV SEI message is a GFV SEI message in the same picture unit with the GFVE SEI message having gfv\_id equal to gfve\_gfv\_id, gfv\_cnt equal to gfve\_gfv\_cnt. The GFVE message is used to enhance the picture generated with the associated GFV SEI message.

For a GFVE SEI message, the following applies:

* The GFVE SEI message shall be present in the same picture unit with the associated GFV SEI message. And the associated GFV SEI message shall be precede the GFVE SEI message in the decoding order. When the associated GFV SEI message is not present in the picture unit containing the GFVE SEI message, the GFVE SEI message shall be ignored.
* When a GFVE SEI message with gfve\_gfv\_cnt equal to 0 is present in the first picture unit in a CLVS in decoding order, gfve\_nn\_present\_flag shall be present and equal to 1.
* If a GFV SEI message A is associated with a GFVE SEI message A, and a GFV SEI message B is associated with a GFVE SEI message B, and GFV SEI message A precede the GFV SEI message B in decoding order, the GFVE SEI A shall also precede GFVE SEI message B in decoding order.
* If a GFVE SEI message A with gfve\_gfv\_id equal to gfveGfvIdA, gfve\_gfv\_cnt value equal to gfveGfvCntA, and a GFVE SEI message B with gfve\_gfv\_id equal to gfveGfvIdB, gfve\_gfv\_cnt value equal to gfveGfvCntB, are present in the same picture unit, and gfveGfvIdA is equal to gfveGfvIdB and gfveGfvCntA is less than gfveGfvCntB, the GFVE SEI message A shall precede the GFVE SEI message B in decoding order.
* If a GFVE SEI message A with gfve\_gfv\_id equal to gfveGfvIdA, gfve\_gfv\_cnt value equal to gfveGfvCntA, and a GFV SEI message B with gfv\_id equal to gfvIdB, gfv\_cnt value equal to gfvCntB, are present in the same picture unit, if gfveGfvIdA is equal to gfvIdB, gfveGfvCntA is less than gfvCntB, the GFVE SEI message A shall precede the GFV SEI message B in decoding order.
* When either of the following conditions is true, GFVE SEI messages shall have the same SEI payload content:
  + The GFVE SEI messages are present in the same picture unit, have gfve\_nn\_base\_flag present and equal to 1 and have the same value of gfve\_id, gfve\_gfv\_id, gfve\_gfv\_cnt.
  + The GFVE SEI messages are present in the same picture unit, have gfve\_nn\_base\_flag not present and have the same value of gfve\_id, gfve\_gfv\_id, gfve\_gfv\_cnt.

**gfve\_nn\_present\_flag** equal to 1 indicatesa neural network that may be used as a EnhancerNN( ) is contained or indicated by the SEI message. gfve\_nn\_present\_flag equal to 0 indicatesa neural network that may be used as a EnhancerNN( ) is not contained or indicated by the SEI message.

**gfve\_nn\_base\_flag**, **gfve\_nn\_mode\_idc**, **gfve\_nn\_alignment\_zero\_bit\_a**, **gfve\_nn\_tag\_uri**, **gfve\_nn\_uri**, **gfve\_nn\_alignment\_zero\_bit\_b**, and **gfve\_nn\_payload\_byte**[ i ] specify a neural network that may be used as a EnhancerNN( ). gfve\_nn\_base\_flag, gfve\_nn\_mode\_idc, gfve\_nn\_alignment\_zero\_bit\_a, gfve\_nn\_tag\_uri, gfve\_nn\_uri, gfve\_nn\_alignment\_zero\_bit\_b, and gfve\_nn\_payload\_byte[ i ] have the same syntax and semantics as nnpfc\_base\_flag, nnpfc\_mode\_idc, nnpfc\_alignment\_zero\_bit\_a, nnpfc\_tag\_uri, nnpfc\_uri, nnpfc\_alignment\_zero\_bit\_b, and nnpfc\_payload\_byte[ i ], respectively.

**gfve\_matrix\_element\_precision\_factor** indicates quantization factor of matrix elements signalled in the SEI message. The value of gfve\_matrix\_element\_precision\_factor shall be in the range of 0 to 32, inclusive.

**gfve\_num\_matrices\_minus1** plus 1 specifies the number of matrices signalled in the SEI message. The value of gfv\_num\_matrices\_minus1 shall be in the range of 0 to 210 − 1, inclusive.

**gfve\_matrix\_height\_minus1**[ i ] plus 1 indicates the height of the i-th matrix. The value of gfve\_matrix\_height\_minus1[ i ] shall be in the range of 0 to 210 − 1, inclusive.

**gfve\_matrix\_width\_minus1**[ i ] plus 1 indicates the width of the i-th matrix. The value of gfve\_matrix\_width\_minus1[ i ] shall be in the range of 0 to 210 − 1, inclusive.

**gfve\_matrix\_element**[ i ][ j ][ k ] is used to derive the value of the element at position (k, j) of the i-th matrix. The value of gfve\_matrix\_element[ i ][ j ][ k ] shall be in the range of 0 to 232 − 2, inclusive.

**gfve\_matrix\_element\_sign\_flag**[ i ][ j ][ k ] indicates the sign of the matrix element at position (k, j) of the i-th matrix.

The variable matrixElementVal[ i ][ j][ k ] representing the value of the matrix element at position (k, j) of the i-th matrix is derived as follows:

matrixElementVal[ i][ j ][ k ] =

The following process is invoked for each GFVE SEI message to enhance the picture generated with the associated GFV SEI message:

DeriveInputTensors( )  
if( ChromaFormatIdc  = =  0 )  
 EnhancerNN( inputBaseY, inputGenY, inputMatrix)  
else  
 EnhancerNN( inputBaseY, inputBaseCb, inputBaseCr, inputGenY, inputGenCb, inputGenCr, inputMatrix)  
StoreOutputTensors( )

The process DeriveInputTensors( ) for deriving the inputs of EnhancerNN( ) is specified as follows:

The BasePicture input tensor inputBaseY, inputBaseCb and inputBaseCr are derived as follows:

for( x = 0; x < CroppedWidth; x++ )  
 for ( y = 0; y < CroppedHeight; y++ )  
 inputBaseY[ x ][ y ] = InpY( baseCroppedYPic[ x ][ y ] )  
if( ChromaFormatIdc  !=  0 )  
 for( x = 0; x < CroppedWidth/ SubWidthC; x++ )  
 for ( y = 0; y < CroppedHeight/ SubHeightC; y++ ) {  
 inputBaseCb[ x][ y ] = InpC( baseCroppedCbPic[ x][ y ] )  
 inputBaseCr[ x][ y ] = InpC( baseCroppedCrPic[ x ][ y ] )  
 }

The GenPicture input tensor inputGenY, inputGenCb and inputGenCr are derived as follows:

for( x = 0; x < CroppedWidth; x++ )  
 for ( y = 0; y < CroppedHeight; y++ )  
 inputGenY[ x ][ y ] = InpY( genCroppedYPic[ x ][ y ] )  
if( ChromaFormatIdc  !=  0 )  
 for( x = 0; x < CroppedWidth / SubWidthC; x++ ) {  
 for ( y = 0; y < CroppedHeight / SubHeightC; y++ ) {  
 inputGenCb[ x][ y ] = InpC( genCroppedCbPic[ x][ y ] )  
 inputGenCr[ x][ y ] = InpC( genCroppedCrPic[ x ][ y ] )  
 }

The matrix input tensor inputMatrix is derived as follows:

for( i = 0; i  <=  gfve\_num\_matrices\_minus1; i++ )  
 for ( j = 0; j  <=  gfve\_matrix\_height\_minus1[ i ]; j++)  
 for( k = 0; k  <=  gfve\_matrix\_width\_minus1[ i ]; k++ )  
 inputMatrix[ i ][ j ][ k ] = matrixElementVal[ i ][ j][ k ]

Where the functions InpY( ) and InpC( ) are specified as follows:

InpY( x ) = x ÷ ( ( 1  <<  BitDepthY ) − 1 )  
InpC( x ) = x ÷ ( ( 1  <<  BitDepthC ) − 1 )

EnhancerNN( ) is a process to enhance the sample values of a generated picture that is generated with the associated GFV SEI message. Input values to EnhancerNN( ) and output values from EnhancerNN( ) are real numbers.

Inputs to EnhancerNN( ) are:

* When ChromaFormatIdc is equal to 0: inputBaseY, inputGenY, inputMatrix
* When ChromaFormatIdc is not equal to 1: inputBaseY, inputBaseCb, inputBaseCr, inputGenY, inputGenCb, inputGenCr, inputMatrix

Outputs of EnhancerNN( ) are:

* A luma sample array enhanceY
* When ChromaFormatIdc is not equal to 0, two chroma sample arrays enhanceCb and enhanceCr.

The process StoreOutputTensors( ) for deriving the output is specified as follows:

The output sample array outYPic[ x ][ y ], outCbPic[ x ][ y ], and outCrPic[ x ][ y ] are derived as follows:

for( x=0; x< CroppedWidth; x++ )  
 for( y=0; y< CroppedHeight; y++ )  
 outputYPic[ x ][ y ] = OutY( enhanceY[ x ][ y ] )  
if( ChromaFormatIdc  !=  0 )  
 for( x = 0; x< CroppedWidth / SubWidthC; x++ )  
 for( y = 0; y< CroppedHeight / SubHeightC; y++ ) {  
 outputCbPic[ x ][ y ] = OutC( enhanceCb[ x ][ y ] )  
 outputCrPic[ x][ y ] = OutC( enhanceCr[ x ][ y ] )  
 }

Where the functions OutY( ) and OutC( ) are specified as follows:

OutY( x ) = Clip3( 0, ( 1  <<  BitDepthY ) − 1 , x \* ( ( 1  <<  BitDepthY ) − 1 )  
OutC( x ) = Clip3( 0, ( 1  <<  BitDepthC ) − 1 , x \* ( ( 1  <<  BitDepthC ) − 1 )

8.38 Film grain regions characteristics SEI message

8.38.1 Film grain regions characteristics SEI message syntax

|  |  |
| --- | --- |
| film\_grain\_regions\_characteristics( payloadSize ) { | **Descriptor** |
| **fgr\_cancel\_flag** | u(1) |
| if( !fgr\_cancel\_flag ) { |  |
| **fgr\_single\_model\_flag** | u(1) |
| if (fgr\_single\_model\_flag) |  |
| **fgr\_model\_id** | u(2) |
| } |  |
| **fgr\_separate\_colour\_description\_present\_flag** | u(1) |
| if( fgr\_separate\_colour\_description\_present\_flag ) { |  |
| **fgr\_bit\_depth\_luma\_minus8** | u(3) |
| **fgr\_bit\_depth\_chroma\_minus8** | u(3) |
| **fgr\_full\_range\_flag** | u(1) |
| **fgr\_colour\_primaries** | u(8) |
| **fgr\_transfer\_characteristics** | u(8) |
| **fgr\_matrix\_coeffs** | u(8) |
| } |  |
| **fgr\_blending\_mode\_id** | u(2) |
| **fgr\_log2\_scale\_factor** | u(4) |
| for( c = 0; c < 3; c++ ) |  |
| **fgr\_comp\_model\_present\_flag**[ c ] | u(1) |
| for( c = 0; c < 3; c++ ) |  |
| if( fgr\_comp\_model\_present\_flag[ c ] ) { |  |
| **fgr\_num\_intensity\_intervals\_minus1**[ c ] | u(8) |
| **fgr\_num\_model\_values\_minus1**[ c ] | u(3) |
| for( i = 0; i <= fgr\_num\_intensity\_intervals\_minus1[ c ]; i++ ) { |  |
| **fgr\_intensity\_interval\_lower\_bound**[ c ][ i ] | u(8) |
| **fgr\_intensity\_interval\_upper\_bound**[ c ][ i ] | u(8) |
| for( j = 0; j <= fgr\_num\_model\_values\_minus1[ c ]; j++ ) |  |
| **fgr\_comp\_model\_value**[ c ][ i ][ j ] | se(v) |
| **}** |  |
| **}** |  |
| **fgr\_region\_information\_present\_flag** | u(1) |
| if( fgr\_region\_information\_present\_flag ) { |  |
| **fgr\_region\_based\_adaptation\_flag** | u(1) |
| if( fgr\_region\_based\_adaptation\_flag ) { |  |
| **fgr\_active\_regions\_number\_minus1** | u(8) |
| for( i = 0; i <= **fgr\_active\_regions\_number\_minus1**; i++ ) { |  |
| **fgr\_region\_top[** i ] | u(16) |
| **fgr\_region\_left**[ i ] | u(16) |
| **fgr\_region\_width**[ i ] | u(16) |
| **fgr\_region\_height**[ i ] | u(16) |
| if (!fgr\_single\_model\_flag) { |  |
| **fgr\_regions\_model\_id**[ i ] | u(2) |
| } |  |
| **fgr\_film\_grain\_enabled\_flag**[ i ] | u(1) |
| if ( fgr\_film\_grain\_enabled\_flag[ i ] ) { |  |
| for( c = 0; c < 3; c++ ) |  |
| if ( fgr\_comp\_model\_present\_flag[ c ] ) { |  |
| **fgr\_region\_interval\_min**[ i ][ c ] | **u(8)** |
| **fgr\_region\_interval\_max**[ i ][ c ] | **u(8)** |
| } |  |
| } |  |
| } |  |
| } |  |
| else |  |
| **fgr\_alpha\_channel\_adaptation\_flag** | u(1) |
| } |  |
| **fgr \_persistence\_flag** | u(1) |
| } |  |
| } |  |

8.38.2 Film grain regions characteristics SEI message semantics

This SEI message provides the decoder with a parameterized model for a film grain synthesis process. The film grain synthesis process should be applied to the decoded pictures prior to their display.

Compared to the FGC SEI message defined in 8.5, this SEI message also provides the decoder with parameters defining how to apply locally the film grain synthesis.

Use of this SEI message requires the definition of the following variables:

– A picture width and picture height in units of luma samples, denoted herein by PicWidthInLumaSamples and PicHeightInLumaSamples, respectively.

– When the syntax element rdf\_separate\_colour\_description\_present\_flag of the film grain regions characteristics SEI message is equal to 0, the following additional variables:

– A chroma format indicator, denoted herein by ChromaFormatIdc, as described in clause 7.3.

– A bit depth for the samples of the luma component, denoted herein by BitDepthY, and when ChromaFormatIdc is not equal to 0, a bit depth for the samples of the two associated chroma components, denoted herein by BitDepthC.

The film grain models specified in the film grain regions characteristics SEI message are expressed for application to decoded pictures that have 4:4:4 colour format with luma and chroma bit depths corresponding to the luma and chroma bit depths of the film grain model and use the same colour representation domain as the identified film grain model. When the colour format of the decoded video is not 4:4:4 or the decoded video uses a different luma or chroma bit depth from that of the film grain model or uses a different colour representation domain from that of the identified film grain model, an unspecified conversion process is expected to be applied to convert the decoded pictures to the form that is expressed for application of the film grain model.

NOTE 4 – Because the use of a specific method is not required for performing the film grain generation function used by the display process, a decoder could, if desired, down-convert the model information for chroma in order to simulate film grain for other chroma formats (4:2:0 or 4:2:2) rather than up-converting the decoded video (using a method not specified in this Specification) before performing film grain generation.

**fgr\_cancel\_flag** equal to 1 indicates that the SEI message cancels the persistence of any previous film grain regions characteristics SEI message in output order that applies to the current layer. fgr\_cancel\_flag equal to 0 indicates that film grain regions characteristics follows.

**fgr\_single\_model\_flag** equal to 1 indicates that the film grain simulation models used in current picture have the same fgr\_model\_id. fgr\_single\_model\_flag equal to 0 indicates that the film grain simulation models used in different regions of the current picture may have different fgr\_model\_id.

**fgr\_model\_id** identifies the film grain simulation model as specified in Table 5. The value of fgr\_model\_id shall be in the range of 0 to 1, inclusive. The values of 2 and 3 for fgr\_model\_id 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 film grain regions characteristics SEI messages with fgr\_model\_id equal to 2 or 3.

**Table 5 – fgr\_model\_id values**

|  |  |
| --- | --- |
| **Value** | **Description** |
| 0 | Frequency filtering |
| 1 | Auto-regression |

**fgr\_separate\_colour\_description\_present\_flag** equal to 1 indicates that a distinct combination of luma bit depth, chroma bit depth, video full range flag, colour primaries, transfer characteristics, and matrix coefficients for the film grain characteristics specified in the SEI message is present in the film grain regions characteristics SEI message syntax. fgr\_separate\_colour\_description\_present\_flag equal to 0 indicates that the combination of luma bit depth, chroma bit depth, video full range flag, colour primaries, transfer characteristics, and matrix coefficients for the film grain characteristics specified in the SEI message are the same as indicated in VUI parameters for the CLVS.

NOTE 5 – When fgr\_separate\_colour\_description\_present\_flag is equal to 1, any of the luma bit depth, chroma bit depth, video full range flag, colour primaries, transfer characteristics, and matrix coefficients specified for the film grain characteristics specified in the SEI message could differ from that for the pictures in the CLVS.

When VUI parameters are not present for the CLVS or the value of vui\_colour\_description\_present\_flag is equal to 0, and equivalent information to that conveyed when vui\_colour\_description\_present\_flag is equal to 1 is not conveyed by external means, fgr\_separate\_colour\_description\_present\_flag shall be equal to 1.

The input image Î, which may be the decoded picture or converted from the decoded picture, used in the equations in this clause is in the same colour representation domain as the simulated film grain signal. Therefore, when any of these parameters does differ from that for the pictures in CLVS, the input image Î used in the equations in this clause would be in a different colour representation domain than that for the pictures in the CLVS. For example, when the value of fgr\_bit\_depth\_luma\_minus8 + 8 is greater than BitDepthY (i.e., the bit depth of the luma component of the pictures in the CLVS), the bit depth of the input image Î used in the equations in this clause is also greater than BitDepthY. In such a case, the input image Î would be generated by converting the actual decoded picture to be in the same colour representation domain as the simulated film grain signal. The process for converting an actual decoded picture to the 4:4:4 colour format with same colour representation domain as the simulated film grain signal is not specified in this Specification.

**fgr\_bit\_depth\_luma\_minus8** plus 8 specifies the bit depth used for the luma component of the film grain characteristics specified in the SEI message. When fgr\_bit\_depth\_luma\_minus8 is not present in the film grain regions characteristics SEI message, the value of fgr\_bit\_depth\_luma\_minus8 is inferred to be equal to BitDepthY − 8.

The value of fgBitDepth[ 0 ] is derived as follows:

fgBitDepth[ 0 ] = fgr\_bit\_depth\_luma\_minus8 + 8 (19)

**fgr\_bit\_depth\_chroma\_minus8** plus 8 specifies the bit depth used for the Cb and Cr components of the film grain characteristics specified in the SEI message. When fgr\_bit\_depth\_chroma\_minus8 is not present in the film grain regions characteristics SEI message, the value of fgr\_bit\_depth\_chroma\_minus8 is inferred to be equal to BitDepthC − 8.

The value of fgBitDepth[ c ] for c = 1 and 2 is derived as follows:

fgBitDepth[ c ] = fgr\_bit\_depth\_chroma\_minus8 + 8, with c = 1, 2 (20)

**fgr\_full\_range\_flag** has the same semantics as specified in clause 7.3 for the vui\_full\_range\_flag syntax element, except as follows:

– fgr\_full\_range\_flag specifies the video full range flag of the film grain characteristics specified in the SEI message, rather than the video full range flag used for the CLVS.

– When fgr\_full\_range\_flag is not present in the film grain regions characteristics SEI message, the value of fgr\_full\_range\_flag is inferred to be equal to vui\_full\_range\_flag.

**fgr\_colour\_primaries** has the same semantics as specified in clause 7.3 for the vui\_colour\_primaries syntax element, except as follows:

– fgr\_colour\_primaries specifies the colour primaries of the film grain characteristics specified in the SEI message, rather than the colour primaries used for the CLVS.

– When fgr\_colour\_primaries is not present in the film grain regions characteristics SEI message, the value of fgr\_colour\_primaries is inferred to be equal to vui\_colour\_primaries.

**fgr\_transfer\_characteristics** has the same semantics as specified in clause 7.3 for the vui\_transfer\_characteristics syntax element, except as follows:

– fgr\_transfer\_characteristics specifies the transfer characteristics of the film grain characteristics specified in the SEI message, rather than the transfer characteristics used for the CLVS.

– When fgr\_transfer\_characteristics is not present in the film grain regions characteristics SEI message, the value of fgr\_transfer\_characteristics is inferred to be equal to vui\_transfer\_characteristics.

**fgr\_matrix\_coeffs** has the same semantics as specified in clause 7.3 for the vui\_matrix\_coeffs syntax element, except as follows:

– fgr\_matrix\_coeffs specifies the matrix coefficients of the film grain characteristics specified in the SEI message, rather than the matrix coefficients used for the CLVS.

– When fgr\_matrix\_coeffs is not present in the film grain regions characteristics SEI message, the value of fgr\_matrix\_coeffs is inferred to be equal to vui\_matrix\_coeffs.

– The values allowed for fgr\_matrix\_coeffs are not constrained by the chroma format of the decoded video pictures that is indicated by the value of ChromaFormatIdc for the semantics of the VUI parameters.

**fgr\_blending\_mode\_id** identifies the blending mode used to blend the simulated film grain with the input images as specified in Table 6. fgr\_blending\_mode\_id shall be in the range of 0 to 1, inclusive. The values of 2 and 3 for fgr\_blending\_mode\_id 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 film grain characteristic SEI messages with fgr\_blending\_mode\_id equal to 2 or 3.

**Table 6 – fgr\_blending\_mode\_id values**

|  |  |
| --- | --- |
| **Value** | **Description** |
| 0 | Additive |
| 1 | Multiplicative |

Depending on the value of fgr\_blending\_mode\_id, the blending mode is specified as follows:

– If fgr\_blending\_mode\_id is equal to 0, the blending mode is additive as specified by:

Igrain[ c ][ x ][ y ] = Clip3( 0, ( 1 << fgBitDepth[ c ] ) − 1, Î[ c ][ x ][ y ] + G[ c ][ x ][ y ] ) (21)

– Otherwise (fgr\_blending\_mode\_id is equal to 1), the blending mode is multiplicative as specified by:

Igrain[ c ][ x ][ y ] = Clip3( 0, ( 1 << fgBitDepth[ c ] ) − 1, Î[ c ][ x ][ y ] + (22)  
 Round( ( Î[ c ][ x ][ y ] \* G[ c ][ x ][ y ] ) ÷ ( ( 1  <<  fgBitDepth[ c ] ) − 1 ) ) )

where Î[ c ][ x ][ y ] represents the sample value at coordinates x, y of the colour component c of the input image Î, G[ c ][ x ][ y ] is the simulated film grain value at the same position and colour component, and fgBitDepth[ c ] is the number of bits used for each sample in a fixed-length unsigned binary representation of the arrays Igrain[ c ][ x ][ y ], Î[ c ][ x ][ y ], and G[ c ][ x ][ y ], where c = 0..2, x = 0..PicWidthInLumaSamples − 1, and y = 0..PicHeightInLumaSamples − 1.

**fgr\_log2\_scale\_factor** specifies a scale factor used in the film grain characterization equations.

**fgr\_comp\_model\_present\_flag**[ c ] equal to 0 indicates that film grain is not modelled on the c-th colour component, where c equal to 0 refers to the luma component, c equal to 1 refers to the Cb component, and c equal to 2 refers to the Cr component. fgr\_comp\_model\_present\_flag[ c ] equal to 1 indicates that syntax elements specifying modelling of film grain on colour component c are present in the SEI message.

When fgr\_separate\_colour\_description\_present\_flag is equal to 0 and ChromaFormatIdc is equal to 0, the value of fgr\_comp\_model\_present\_flag[ 1 ] and fgr\_comp\_model\_present\_flag[ 2 ] shall be equal to 0.

**fgr\_num\_intensity\_intervals\_minus1**[ c ] plus 1 specifies the number of intensity intervals for which a specific set of model values has been estimated.

NOTE 6 – The intensity intervals could overlap in order to simulate multi-generational film grain.

**fgr\_num\_model\_values\_minus1**[ c ] plus 1 specifies the number of model values present for each intensity interval in which the film grain has been modelled. The value of fgr\_num\_model\_values\_minus1[ c ] shall be in the range of 0 to 5, inclusive.

**fgr\_intensity\_interval\_lower\_bound**[ c ][ i ] specifies the lower bound of the i-th intensity interval for which the set of model values applies.

**fgr\_intensity\_interval\_upper\_bound**[ c ][ i ] specifies the upper bound of the i-th intensity interval for which the set of model values applies.

The variable intensityIntervalIdx[ c ][ x ][ y ][ j ] represents the j-th index to the list of intensity intervals selected for the sample value Î[ c ][ x ][ y ] for c = 0..2, x = 0..PicWidthInLumaSamples − 1, y = 0..PicHeightInLumaSamples − 1, and j = 0..numApplicableIntensityIntervals[ c ][ x ][ y ] − 1, where numApplicableIntensityIntervals[ c ][ x ][ y ] is derived below.

The minimum and maximum interval index for the sample of colour component c at coordinate x, y is derived as follows:

* If fgr\_region\_based\_adaptation\_flag is equal to 0, intensityIntervalMin[ c ][ x ][ y ] is set equal to 0 and intensityIntervalMax[ c ][ x ][ y ] is set equal to fgr\_num\_intensity\_intervals\_minus1[ c ]
* Otherwise, intensityIntervalMax[ c ][ x ][ y ] and intensityIntervalMin[ c ][ x ][ y ] are derived as follows:

intensityIntervalMin[ c ][ x ][ y ] = 1  
intensityIntervalMax[ c ][ x ][ y ] = 0  
for( i = 0; i <= fgr\_active\_regions\_number\_minus1; i++ )  
 if ( x >= fgr\_region\_left[ i ] && x < fgr\_region\_left[ i ] + fgr\_region\_width[ i ] &&  
 y >= fgr\_region\_top[ i ] && y < fgr\_region\_top[ i ] + fgr\_region\_height[ i ] )  
 if ( fgr\_film\_grain\_enabled\_flag[ i ] ) { (23)  
 intensityIntervalMin[ c ][ x ][ y ] = fgr\_region\_interval\_min[ i ][ c ]  
 intensityIntervalMax[ c ][ x ][ y ] = fgr\_region\_interval\_max[ i ][ c ]  
 } else {  
 intensityIntervalMin[ c ][ x ][ y ] = 1  
 intensityIntervalMax[ c ][ x ][ y ] = 0  
 }

Depending on the value of fgr\_model\_id, the selection of the one or more intensity intervals for the sample value Î[ c ][ x ][ y ] is specified as follows:

– The variable numApplicableIntensityIntervals[ c ][ x ][ y ] is initially set equal to 0.

– If fgr\_model\_id is equal to 0, the following applies:

* The top-left sample location ( xB, yB ) of the current 8x8 block b that contains the sample value Î[ c ][ x ][ y ] is derived as ( xB, yB ) = ( x / 8, y / 8 ).
* The average value bavg of the current 8x8 block b is derived as follows:

sum8x8 = 0  
for( i = 0; i < 8; i++ )  
 for( j = 0; j < 8; j++ )  
 sum8x8 += Î[ c ][ xB \* 8 + i ][yB \* 8 + j ] (23)  
bavg = Clip3( 0, 255, ( sum8x8 + ( 1  <<  ( fgBitDepth[ c ] − 3 ) ) )  >>  ( fgBitDepth[ c ] − 2 ) )

* The value of intensityIntervalIdx[ c ][ x ][ y ][ j ] is derived as follows:

for( i = intensityIntervalMin[ c ][ x ][ y ], j = 0; i <= intensityIntervalMax[ c ][ x ][ y ]; i++ )  
 if( bavg  >=  fgr\_intensity\_interval\_lower\_bound[ c ][ i ]  &&  
 bavg  <=  fgr\_intensity\_interval\_upper\_bound[ c ][ i ] ) {  
 intensityIntervalIdx[ c ][ x ][ y ][ j ] = i (24)  
 j++  
 }  
numApplicableIntensityIntervals[ c ][ x ][ y ] = j

– Otherwise (fgr\_model\_id is equal to 1), the value of intensityIntervalIdx[ c ][ x ][ y ][ j ] is derived as follows:

I8[ c ][ x ][ y ] = ( fgBitDepth[ c ]  = =  8 ) ? ( Î[ c ][ x ][ y ] :  
 Clip3( 0, 255, ( Î[ c ][ x ][ y ] + ( 1  <<  ( fgBitDepth[ c ] − 9 ) ) )  >>  ( fgBitDepth[ c ] − 8 ) )  
for( i = intensityIntervalMin[ c ][ x ][ y ], j = 0; i <= intensityIntervalMax[ c ][ x ][ y ]; i++ )  
 if( I8[ c ][ x ][ y ]  >=  fgr\_intensity\_interval\_lower\_bound[ c ][ i ]  && (25)  
 I8[ c ][ x ][ y ]  <=  fgr\_intensity\_interval\_upper\_bound[ c ][ i ] ) {  
 intensityIntervalIdx[ c ][ x ][ y ][ j ] = i  
 j++  
 }  
numApplicableIntensityIntervals[ c ][ x ][ y ] = j

Samples that do not fall into any of the defined intervals (i.e., those samples for which the value of numApplicableIntensityIntervals[ c ][ x ][ y ] is equal to 0) are not modified by the grain generation function. Samples that fall into more than one interval (i.e., those samples for which the value of numApplicableIntensityIntervals[ c ][ x ][ y ] is greater than 1) will originate multi-generation grain. Multi-generation grain results from adding the grain computed independently for each of the applicable intensity intervals.

In the equations in the remainder of this clause, the variable sj in each instance of the list fgr\_comp\_model\_value[ c ][ sj ] is the value of intensityIntervalIdx[ c ][ x ][ y ][ j ] derived for the sample value Î[ c ][ x ][ y ].

**fgr\_comp\_model\_value**[ c ][ i ][ j ] specifies the j-th model value for the colour component c and the i-th intensity interval. The set of model values has different meaning depending on the value of fgr\_model\_id.

The value of fgr\_comp\_model\_value[ c ][ i ][ j ] is constrained as follows, and could be additionally constrained as specified elsewhere in this clause:

– If fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ j ] shall be in the range of 0 to 2fgBitDepth[ c ] − 1, inclusive.

– Otherwise (fgr\_model\_id is equal to 1), fgr\_comp\_model\_value[ c ][ i ][ j ] shall be in the range of −2( fgBitDepth[ c ] − 1 ) to 2( fgBitDepth[ c ] − 1 ) − 1, inclusive.

Depending on the value of fgr\_model\_id, the synthesis of the film grain is modelled as follows:

– If fgr\_model\_id is equal to 0, a frequency filtering model enables simulating the original film grain for c = 0..2, x = 0..PicWidthInLumaSamples − 1, and y = 0..PicHeightInLumaSamples − 1 as specified by:

G[ c ][ x ][ y ] = ( fgr\_comp\_model\_value[ c ][ sj ][ 0 ] \* Q[ c ][ x ][ y ] + fgr\_comp\_model\_value[ c ][ sj ][ 5 ] \*  
 G[ c − 1 ][ x ][ y ] ) >> fgr\_log2\_scale\_factor (26)

where Q[ c ] is a two-dimensional random process generated by filtering 16x16 blocks gaussRv with random-valued elements gaussRvij generated with a normalized Gaussian distribution (independent and identically distributed Gaussian random variable samples with zero mean and unity variance) and where the value of an element G[ c − 1 ][ x ][ y ] used in the right-hand side of the equation is inferred to be equal to 0 when c − 1 is less than 0.

NOTE 7 – A normalized Gaussian random variable can be generated from two independent, uniformly distributed random values over the interval from 0 to 1 (and not equal to 0), denoted as uRv0 and uRv1, using the Box-Muller transformation specified by:

gaussRvi,j = Sqrt( −2 \* Ln( uRv0 ) ) \* Cos( 2 \* π \* uRv1 ) (27)

where π is Archimedes' constant 3.141 592 653 589 793....

The band-pass filtering of blocks gaussRv can be performed in the discrete cosine transform (DCT) domain as follows:

for( y = 0; y < 16; y++ )  
 for( x = 0; x < 16; x++ )  
 if( ( x < fgr\_comp\_model\_value[ c ][ sj ][ 3 ]  &&  y < fgr\_comp\_model\_value[ c ][ sj ][ 4 ] )  | | (28)  
 x > fgr\_comp\_model\_value[ c ][ sj ][ 1 ]  | |  y > fgr\_comp\_model\_value[ c ][ sj ][ 2 ] )  
 gaussRv[ x ][ y ] = 0  
filteredRv = IDCT16x16( gaussRv )

where IDCT16x16( z ) refers to a unitary inverse discrete cosine transformation (IDCT) operating on a 16x16 matrix argument z as specified by:

IDCT16x16( z ) = r \* z \* rT (29)

where the superscript T indicates a matrix transposition and r is the 16x16 matrix with elements rij specified by:

(30)

where π is Archimedes' constant 3.141 592 653 589 793....

Q[ c ] is formed by the frequency-filtered blocks filteredRv.

NOTE 8 – Coded model values are based on blocks of size 16x16, but a decoder implementation could use other block sizes. For example, decoders implementing the IDCT on 8x8 blocks could down-convert by a factor of two the set of coded model values fgr\_comp\_model\_value[ c ][ sj ][ i ] for i equal to 1..4.

NOTE 9 – To reduce the degree of visible blocks that result from mosaicking the frequency-filtered blocks filteredRv, decoders could apply a low-pass filter to the boundaries between frequency-filtered blocks.

– Otherwise (fgr\_model\_id is equal to 1), an auto-regression model enables simulating the original film grain for c = 0..2, x = 0..PicWidthInLumaSamples − 1, and y = 0..PicHeightInLumaSamples − 1 as specified by:

G[ c ][ x ][ y ] = ( fgr\_comp\_model\_value[ c ][ sj ][ 0 ] \* n[ c ][ x][ y ] +  
 fgr\_comp\_model\_value[ c ][ sj ][ 1 ] \* ( G[ c ][ x − 1 ][ y ] +  
 ( ( fgr\_comp\_model\_value[ c ][ sj ][ 4 ] \* G[ c ][ x ][ y − 1 ] ) >>  
 fgr\_log2\_scale\_factor ) ) + fgr\_comp\_model\_value[ c ][ sj ][ 3 ] \*  
 ( ( fgr\_comp\_model\_value[ c ][ sj ][ 4 ] \* ( G[ c ][ x − 1 ][ y − 1 ] + G[ c ][ x + 1 ][ y − 1 ] ) ) >> (31)  
 fgr\_log2\_scale\_factor ) +  
 fgr\_comp\_model\_value[ c ][ sj ][ 5 ] \* ( G[ c ][ x − 2 ][ y ] +  
 ( ( fgr\_comp\_model\_value[ c ][ sj ][ 4 ] \* fgr\_comp\_model\_value[ c ][ sj ][ 4 ] \* G[ c ][ x ][ y − 2 ] ) >>  
 ( 2 \* fgr\_log2\_scale\_factor ) ) ) +  
 fgr\_ comp\_model\_value[ c ][ sj ][ 2 ] \* G[ c − 1 ][ x ][ y ] ) >> fgr\_log2\_scale\_factor

where n[ c ][ x ][ y ] is a random value with normalized Gaussian distribution (independent and identically distributed Gaussian random variable samples with zero mean and unity variance for each value of c, x, and y) and where the value of an element G[ c ][ x ][ y ] used in the right-hand side of the equation is inferred to be equal to 0 when any of the following conditions are true:

– c is less than 0,

– x is less than 0,

– y is less than 0.

fgr\_comp\_model\_value[ c ][ i ][ 0 ] provides the first model value for the model as specified by fgr\_model\_id. fgr\_comp\_model\_value[ c ][ i ][ 0 ] corresponds to the standard deviation of the Gaussian noise term in the generation functions specified in Equations 26 through 31.

fgr\_comp\_model\_value[ c ][ i ][ 1 ] provides the second model value for the model as specified by fgr\_model\_id. When fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 1 ] shall be greater than or equal to 0 and less than 16.

When not present in the film grain regions characteristics SEI message, fgr\_comp\_model\_value[ c ][ i ][ 1 ] is inferred as follows:

– If fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 1 ] is inferred to be equal to 8.

– Otherwise (fgr\_model\_id is equal to 1), fgr\_comp\_model\_value[ c ][ i ][ 1 ] is inferred to be equal to 0.

fgr\_comp\_model\_value[ c ][ i ][ 1 ] is interpreted as follows:

– If fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 1 ] indicates the horizontal high cut frequency to be used to filter the DCT of a block of 16x16 random values.

– Otherwise (fgr\_model\_id is equal to 1), fgr\_comp\_model\_value[ c ][ i ][ 1 ] indicates the first order spatial correlation for neighbouring samples ( x − 1, y ) and ( x, y − 1 ).

fgr\_comp\_model\_value[ c ][ i ][ 2 ] provides the third model value for the model as specified by fgr\_model\_id. When fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 2 ] shall be greater than or equal to 0 and less than 16.

When not present in the film grain regions characteristics SEI message, fgr\_comp\_model\_value[ c ][ i ][ 2 ] is inferred as follows:

– If fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 2 ] is inferred to be equal to fgr\_comp\_model\_value[ c ][ i ][ 1 ]

– Otherwise (fgr\_model\_id is equal to 1), fgr\_comp\_model\_value[ c ][ i ][ 2 ] is inferred to be equal to 0.

fgr\_comp\_model\_value[ c ][ i ][ 2 ] is interpreted as follows:

– If fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 2 ] indicates the vertical high cut frequency to be used to filter the DCT of a block of 16x16 random values.

– Otherwise (fgr\_model\_id is equal to 1), fgr\_comp\_model\_value[ c ][ i ][ 2 ] indicates the colour correlation between consecutive colour components.

fgr\_comp\_model\_value[ c ][ i ][ 3 ] provides the fourth model value for the model as specified by fgr\_model\_id. When fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 3 ] shall be greater than or equal to 0 and less than or equal to fgr\_comp\_model\_value[ c ][ i ][ 1 ].

When not present in the film grain regions characteristics SEI message, fgr\_comp\_model\_value[ c ][ i ][ 3 ] is inferred to be equal to 0.

fgr\_comp\_model\_value[ c ][ i ][ 3 ] is interpreted as follows:

– If fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 3 ] indicates the horizontal low cut frequency to be used to filter the DCT of a block of 16x16 random values.

– Otherwise (fgr\_model\_id is equal to 1), fgr\_comp\_model\_value[ c ][ i ][ 3 ] indicates the first order spatial correlation for neighbouring samples ( x − 1, y − 1 ) and ( x + 1, y − 1 ).

fgr\_comp\_model\_value[ c ][ i ][ 4 ] provides the fifth model value for the model as specified by fgr\_model\_id. When fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 4 ] shall be greater than or equal to 0 and less than or equal to fgr\_comp\_model\_value[ c ][ i ][ 2 ].

When not present in the film grain regions characteristics SEI message, fgr\_comp\_model\_value[ c ][ i ][ 4 ] is inferred to be equal to fgr\_model\_id.

fgr\_comp\_model\_value[ c ][ i ][ 4 ] is interpreted as follows:

– If fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 4 ] indicates the vertical low cut frequency to be used to filter the DCT of a block of 16x16 random values.

– Otherwise (fgr\_model\_id is equal to 1), fgr\_comp\_model\_value[ c ][ i ][ 4 ] indicates the aspect ratio of the modelled grain.

fgr\_comp\_model\_value[ c ][ i ][ 5 ] provides the sixth model value for the model as specified by fge\_model\_id.

When not present in the film grain regions characteristics SEI message, fge\_comp\_model\_value[ c ][ i ][ 5 ] is inferred to be equal to 0.

fgr\_comp\_model\_value[ c ][ i ][ 5 ] is interpreted as follows:

– If fgr\_model\_id is equal to 0, fgr\_comp\_model\_value[ c ][ i ][ 5 ] indicates the colour correlation between consecutive colour components.

– Otherwise (fgr\_model\_id is equal to 1), fgr\_comp\_model\_value[ c ][ i ][ 5 ] indicates the second order spatial correlation for neighbouring samples ( x, y − 2 ) and ( x − 2, y ).

**fgr\_region\_based\_adaptation\_flag** equal to 1 indicates that the spatial adaption of the film grain is defined per each region of the present SEI message and described by a rectangle. When not present in the film grain regions characteristics SEI message, fgr\_region\_based\_adaptation\_flag is inferred to be equal to 0.

**fgr\_active\_regions\_number\_minus1** plus 1 indicates the number of regions for which the film grain adaptation is specified. fgr\_active\_regions\_number\_minus1 shall be in the range of 0 to 255, inclusive.

**fgr\_region\_top**[ i ], **fgr\_region\_left**[ i ], **fgr\_region\_width**[ i ], and **fgr\_region\_height**[ i ] specify the coordinates of the top-left corner and the width and height, respectively, of the bounding box of the i-th region in the picture.

The value of fgr\_region\_left[ i ] shall be in the range of 0 to PicWidthInLumaSamples − 1, inclusive.

The value of fgr\_region\_top[ i ] shall be in the range of 0 to PicHeightInLumaSamples  − 1, inclusive.

The value of fgr\_region\_width[ i ] shall be in the range of 0 to PicWidthInLumaSamples  − fgr\_region\_left [ i ], inclusive.

The value of fgr\_region\_height[ i ] shall be in the range of 0 to PicHeightInLumaSamples  − fgr\_region\_top[ i ], inclusive.

**fgr\_regions\_model\_id**[ i ] identifies the film grain simulation model as specified in Table 5 for the i-th region in the picture. The value of fgr\_regions\_model\_id[ i ] shall be in the range of 0 to 1, inclusive. The values of 2 and 3 for fgr\_model\_id 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 film grain regions characteristics SEI messages with fgr\_model\_id equal to 2 or 3.

**fgr\_film\_grain\_enabled\_flag**[ i ] equal to 1 indicates that the film grain characteristics using region intervals are applied on the i-th region. fge\_film\_grain\_enabled\_flag[ i ] equals to 0 indicates that no film grain synthesis is applied to the i-th region of the current picture.

**fgr\_region\_interval\_min**[ i ][ c ] and **fgr\_region\_interval\_max**[ i ][ c ] specify the range of interval indexes for color component c that are applicable to the i-th region. fgr\_region\_interval\_min[ i ][ c ] shall be in the range of 0 to fg\_num\_intensity\_intervals\_minus1[ c ], inclusive, and fgr\_region\_interval\_max[ i ][ c ] shall be in the range of fgr\_region\_interval\_min[ i ][ c ] to fg\_num\_intensity\_intervals\_minus1[ c ], inclusive.

NOTE – Overlapping regions are processed in the increasing order of region index i, which means that a sample belonging to the region with index i=N with min and max film grain intervals indexes equal to x and y, respectively, and also belonging to region with index i=M (with M>N) with min and max film grain intervals indexes equal to x’ and y’, respectively, is processed with film grain characteristics defined with min and max film grain intervals indexes equal to x’ and y’.

**fgr\_alpha\_channel\_adaptation\_flag** equal to 1 indicates that the decoded alpha planes coded in auxiliary pictures of type AUX\_ALPHA are applied to the simulated film grain values defined in the present SEI message prior to be applied are associated primary picture.

When an AU contains an SDI SEI message with sdi\_aux\_id[ i ] equal to 1 for at least one value of i, the SDI SEI message shall precede the FRGC SEI message in decoding order.

When an access unit contains an auxiliary picture picA in a layer, with nuh\_layer\_id equal to nuhLayerIdA, that is indicated as an alpha auxiliary layer by an SDI SEI message, the alpha channel sample values of picA persist in output order until one or more of the following conditions are true:

– The next picture, in output order, with nuh\_layer\_id equal to nuhLayerIdA is output.

– A CLVS containing the auxiliary picture picA ends.

– The bitstream ends.

– A CLVS of any associated primary layer of the auxiliary picture layer with nuh\_layer\_id equal to nuhLayerIdA ends.

If the current PU contains an SDI SEI message with sdi\_aux\_id[ i ] equal to 1 for at least one value of i, the simulated film grain values are weighted by the auxiliary data as follow:

– If fgr\_blending\_mode\_id is equal to 0, the blending mode is additive as specified by:

Igrain[ c ][ x ][ y ] = Clip3( 0, ( 1 << fgBitDepth[ c ] ) − 1, Î[ c ][ x ][ y ] + G[ c ][ x ][ y ]\* Iaux[ c ][ x ][ y ])

– Otherwise (fgr\_blending\_mode\_id is equal to 1), the blending mode is multiplicative as specified by:

Igrain[ c ][ x ][ y ] = Clip3( 0, ( 1 << fgBitDepth[ c ] ) − 1, Î[ c ][ x ][ y ] +  
 Round( ( Î[ c ][ x ][ y ] \* G[ c ][ x ][ y ]\* Iaux[ c ][ x ][ y ]) ) ÷ ( ( 1  <<  fgBitDepth[ c ] ) − 1 ) ) )

where Î[ c ][ x ][ y ] represents the sample value at coordinates x, y of the colour component c of the input image Î, G[ c ][ x ][ y ] is the simulated film grain value at the same position and colour component, and fgBitDepth[ c ], is the number of bits used for each sample in a fixed-length unsigned binary representation of the arrays Igrain[ c ][ x ][ y ], Î[ c ][ x ][ y ], and G[ c ][ x ][ y ], where c = 0..2, x = 0..PicWidthInLumaSamples − 1, and y = 0..PicHeightInLumaSamples – 1, and where Iaux[ c ][ x ][ y ] represents the sample value at coordinates x, y of the colour component c of the decoded auxiliary picture.

**fgr\_persistence\_flag** specifies the persistence of the film grain regions characteristics SEI message for the current layer.

fgr\_persistence\_flag equal to 0 specifies that the film grain regions characteristics SEI message applies to the current decoded picture only.

fgr\_persistence\_flag equal to 1 specifies that the film grain regions characteristics SEI message applies to the current decoded picture and persists for all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture in the current layer in an AU associated with a film grain regions characteristics SEI message is output that follows the current picture in output order.

8.39 Packed regions information SEI message

8.39.1 Packed regions information SEI message syntax

|  |  |
| --- | --- |
| packed\_regions\_info( payloadSize) | **Descriptor** |
| **pri\_cancel\_flag** | u(1) |
| if ( !pri\_cancel\_flag ) { |  |
| **pri\_persistence\_flag** | u(1) |
| **pri\_num\_regions\_minus1** | ue(v) |
| **pri\_use\_max\_dimensions\_flag** | u(1) |
| **pri\_log2\_unit\_size** | u(4) |
| **pri\_region\_size\_len\_minus1** | u(4) |
| **pri\_region\_id\_present\_flag** | u(1) |
| **pri\_target\_pic\_params\_present\_flag** | u(1) |
| if( pri\_target\_pic\_params\_present\_flag ) { |  |
| **pri\_target\_pic\_width\_minus1** | u(16) |
| **pri\_target\_pic\_height\_minus1** | u(16) |
| } |  |
| **pri\_num\_resampling\_ratios\_minus1** | ue(v) |
| for( i = 1; i <= pri\_num\_resampling\_ratios\_minus1; i++ ) { |  |
| **pri\_resampling\_width\_num\_minus1**[ i ] | ue(v) |
| **pri\_resampling\_width\_denom\_minus1**[ i ] | ue(v) |
| **pri\_fixed\_aspect\_ratio\_flag**[ i ] | u(1) |
| if( !pri\_fixed\_aspect\_ratio\_flag[ i ] ) { |  |
| **pri\_resampling\_height\_num\_minus1**[ i ] | ue(v) |
| **pri\_resampling\_height\_denom\_minus1**[ i ] | ue(v) |
| } |  |
| } |  |
| for( i = 0; i <= pri\_num\_regions\_minus1; i++ ) { |  |
| if( pri\_region\_id\_present\_flag ) |  |
| **pri\_region\_id**[ i ] | ue(v) |
| **pri\_region\_top\_left\_in\_units\_x**[ i ] | u(v) |
| **pri\_region\_top\_left\_in\_units\_y**[ i ] | u(v) |
| **pri\_region\_width\_in\_units\_minus1**[ i ] | u(v) |
| **pri\_region\_height\_in\_units\_minus1**[ i ] | u(v) |
| if( pri\_num\_resampling\_ratios\_minus1 > 0 ) |  |
| **pri\_resampling\_ratio\_idx**[ i ] | u(v) |
| if( pri\_target\_pic\_params\_present\_flag ) |  |
| **pri\_target\_region\_top\_left\_x**[ i ] | u(v) |
| **pri\_target\_region\_top\_left\_y**[ i ] | u(v) |
| } |  |
| } |  |
| } |  |
| } |  |

8.39.2 Packed regions information SEI message semantics

The packed regions info SEI message provides information regarding rectangular regions packed with the coded picture. This information may optionally be used to reconstruct a target picture from the samples of the cropped decoded picture corresponding to the regions described in this SEI message.

Use of this SEI message requires the definition of the following variables:

– A picture width and picture height in units of luma samples, denoted herein by PicWidthInLumaSamples and PicHeightInLumaSamples, respectively.

– A maximum picture width and maximum picture height in units of luma samples, denoted herein by MaxPicWidth and MaxPicHeight, respectively.

– A chroma format indicator, denoted herein by ChromaFormatIdc, as described in clause 7.3.

– A bit depth for the samples of the luma component, denoted herein by BitDepthY, and when ChromaFormatIdc is not equal to 0, a bit depth for the samples of the two associated chroma components, denoted herein by BitDepthC.

**pri\_cancel\_flag** equal to 1 indicates that the SEI message cancels the persistence of any previous packed regions information SEI message in output order that applies to the current layer. pri\_cancel\_flagequal to 0 indicates that packed regions information follows.

**pri\_persistence\_flag** specifies the persistence of the packed regions information SEI message for the current layer.

pri\_persistence\_flag equal to 0 specifies that the packed regions information applies to the current decoded picture only.

pri\_persistence\_flag equal to 1 specifies that the packed regions information SEI message applies to the current decoded picture and persists for all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture in the current layer in an AU associated with a packed regions information SEI message is output that follows the current picture in output order.

**pri\_num\_regions\_minus1** plus 1 specifies the number of regions for which information is signalled.

**pri\_use\_max\_dimensions\_flag** equal to 1 specifies that MaxPicWidth, MaxPicHeight, PicWidthInLumaSamples and PicHeightInLumaSamples will be used in variable calculations. pri\_use\_max\_dimensions\_flag equal to 0 specifies that MaxPicWidth, MaxPicHeight, PicWidthInLumaSamples and PicHeightInLumaSamples will not be used in variable calculations for the region parameters.

**pri\_log2\_unit\_size** specifies a unit size used in variable calculations for the region parameters.

The variable priUnitSize is set equal to 1 << pri\_log2\_unit\_size.

**pri\_region\_size\_len\_minus1** plus 1 specifies the number of bits used to signal pri\_region\_top\_left\_in\_units\_x[ i ], pri\_region\_top\_left\_in\_units\_y[ i ],pri\_region\_width\_in\_units\_minus1[ i ], pri\_region\_height\_in\_units\_minus1[ i ], pri\_target\_region\_top\_left\_x[ i ], and pri\_target\_region\_top\_left\_y[ i ].

**pri\_region\_id\_present\_flag** equal to 1 indicates the pri\_region\_id[ i ] syntax element is present.pri\_region\_id\_present\_flag equal to 0 indicates the pri\_region\_id[ i ] syntax element is not present.

**pri\_target\_pic\_params\_present\_flag** equal to 1 indicates the pri\_target\_region\_top\_left\_x[ i ], pri\_target\_region\_top\_left\_y[ i ], pri\_target\_pic\_width\_minus1, and pri\_target\_pic\_height\_minus1 syntax elements are present.pri\_target\_pic\_params\_present\_flag equal to 0 indicates the pri\_target\_region\_top\_left\_x[ i ], pri\_target\_region\_top\_left\_y[ i ], pri\_target\_pic\_width\_minus1, and pri\_target\_pic\_height\_minus1 syntax elements are not present.

**pri\_target\_pic\_width\_minus1** plus 1and **pri\_target\_pic\_height\_minus1** plus 1, when present,indicate the width and height, respectively, in luma samples of the target picture that may be reconstructed from the samples of the cropped decoded picture corresponding to the regions described in this SEI message.

**pri\_num\_resampling\_ratios\_minus1** specifies the number of resampling ratios that are signalled.

**pri\_resampling\_width\_num\_minus1**[ i ] plus 1 and **pri\_resampling\_width\_denom\_minus1**[ i ] plus 1 specify the numerator and denominator, respectively, for the width resampling of the i-th resampling ratio. Both pri\_resampling\_width\_num\_minus1[ i ] and pri\_resampling\_width\_denom\_minus1[ i ] shall be in the range of 0 to 65 535, inclusive.

The values of pri\_resampling\_ratio\_width\_num\_minus1[ 0 ] and pri\_resampling\_ratio\_width\_denom\_minus1[ 0 ] are inferred to be equal to 0.

**pri\_fixed\_aspect\_ratio\_flag**[ i ] equal to 1 specifies that the pri\_resampling\_height\_num\_minus1[ i ] and pri\_resampling\_height\_denom\_minus1[ i ] syntax elements are not present. pri\_fixed\_aspect\_ratio\_flag[ i ] equal to 0 specifies that the pri\_resampling\_height\_num\_minus1[ i ] and pri\_resampling\_height\_denom\_minus1[ i ] syntax elements are present.

**pri\_resampling\_height\_num\_minus1**[ i ] plus 1 and **pri\_resampling\_height\_denom\_minus1**[ i ] plus 1 specify the numerator and denominator, respectively, for the height resampling of the i-th resampling ratio. Both pri\_resampling\_height\_num\_minus1[ i ] and pri\_resampling\_height\_denom\_minus1[ i ] shall be in the range of 0 to 65 535, inclusive. When not present, the values of pri\_resampling\_height\_num\_minus1[ i ] and pri\_resampling\_height\_denom\_minus1[ i ] are inferred to be equal to the pri\_resampling\_width\_num\_minus1[ i ] and pri\_resampling\_width\_denom\_minus1[ i ], respectively.

**pri\_region\_id**[ i ] indicates the ID of the i-th region. When not present, the value of pri\_region\_id[ i ] is inferred to be equal to i.

**pri\_region\_top\_left\_in\_units\_x**[ i ] and **pri\_region\_top\_left\_in\_units\_y**[ i ] specify the horizontal and vertical positions, respectively, of the top left sample of the i-th region in units. The length of the syntax elements are pri\_region\_size\_len\_minus1 + 1.

The variables priRegionTopLeftX[ i ] and priRegionTopLeftY, representing the horizontal and vertical positions, respectively, in luma samples of the region in the cropped decoded picture, are derived as follows: [Ed. (JB): Add equation numbers when integrated.]

if( !pri\_use\_max\_dimensions\_flag ) {  
 priRegionTopLeftX[ i ] = pri\_region\_top\_left\_in\_units\_x[ i ] \* priUnitSize  
 priRegionTopLeftY[ i ] = pri\_region\_top\_left\_in\_units\_y[ i ] \* priUnitSize  
} else {  
 priRegionTopLeftX[ i ] = ( pri\_region\_top\_left\_in\_units\_x[ i ] \* priUnitSize \*   
 PicWidthInLumaSamples + MaxPicWidth /2 ) / MaxWidth  
 priRegionTopLeftY[ i ] = ( pri\_region\_top\_left\_in\_units\_y[ i ] \* priUnitSize\*   
 PicHeightInLumaSamples + MaxPicHeight /2 ) / MaxHeight  
}

**pri\_region\_width\_in\_units\_minus1**[ i ] plus 1 and **pri\_region\_height\_in\_units\_minsu1**[ i ] plus 1 specify the horizontal and vertical positions, respectively, of the width and height of the i-th region in units. The length of the syntax elements are pri\_region\_size\_len\_minus1 + 1.

The variables priRegionWidth[ i ] and priRegionHeight[ i ], representing the width and height, respectively, in luma samples of the i-th region in the cropped decoded picture are derived as follows:

if( !pri\_use\_max\_dimensions\_flag ) {  
 priRegionWidth[ i ] = ( pri\_region\_width\_in\_units\_minus1 [ i ] + 1) \* priUnitSize   
 priRegionHeight[ i ] = ( pri\_region\_height\_in\_units\_minus1 [ i ] + 1) \* priUnitSize   
} else {  
 priRegionWidth[ i ] = ( ( pri\_region\_width\_in\_units\_minus1 [ i ] + 1) \* priUnitSize \*  
 PicWidthInLumaSamples + MaxPicWidth/2 ) / MaxWidth  
 priRegionHeight[ i ] = ( ( pri\_region\_height\_in\_units\_minus1 [ i ] + 1) \* priUnitSize \*   
 PicHeightInLumaSamples + MaxPicHeight/2 ) / MaxHeight  
}

The variables SubWidthC and SubHeightC are derived from ChromaFormatIdc as specified by Table 2.

It is a requirement of bitstream conformance that priRegionWidth[ i ] % SubWidthC shall be equal to 0 and priRegionHeight[ i ] % SubHeightC shall be equal to 0.

**pri\_resampling\_ratio\_idx**[ i ] specifies the index of the resampling ratio used for the i-th region. The length of the syntax element is Ceil( Log2( pri\_num\_resampling\_ratios\_minus1 + 1 ) ).

The variables priResampleWidthNum[ i ], priResampleWidthDenom[ i ], priResampleHeightNum[ i ]  and  priResampleHeightDenom[ i ]  are derived as follows.

priResampleWidthNum[ i ] = pri\_resampling\_width\_num\_minus1[ pri\_rsampling\_ratio\_idx[ i ] ] + 1

priResampleWidthDenom[ i ] =

pri\_resampling\_width\_denom\_minus1[ pri\_rsampling\_ratio\_idx[ i ] ] + 1

priResampleHeightNum[ i ] =

pri\_resampling\_height\_num\_minus1[ pri\_rsampling\_ratio\_idx[ i ] ] + 1

priResampleHeightDenom[ i ] =

pri\_resampling\_height\_denom\_minus1[ pri\_rsampling\_ratio\_idx[ i ] ] + 1

**pri\_target\_region\_top\_left\_x**[ i ] and **pri\_target\_region\_top\_left\_y**[ i ], when present, indicate the horizontal and vertical positions, respectively, of the top left sample position in luma samples of the i-th region in the reconstructed target picture.

The variables priTargetRegionWidth and priTargetHeight, representing the width and height, respectively, in luma samples of the resampled region in the reconstructed target picture, are derived as follows:

priTargetRegionWidth = Round( ( priRegionWidth[ i ] \* priResampleWidthNum[ i ] ) ÷

( priResampleWidthDenom[ i ]  \* SubWidthC ) ) \* SubWidthC

priTargetRegionHeight = Round( ( priRegionHeight[ i ] \* priResampleHeightNum[ i ] ) ÷

( priResampleHeightDenom[ i ] \* SubHeightC ) ) \* SubHeightC

When reconstructing a target picture with luma sample array of size (pri\_target\_pic\_width\_minus1 + 1) × (pri\_target\_pic\_height\_minus1 + 1), all luma sample values are initialized to value 1  << ( BitDepthY – 1) and chroma samples, if present, to 1  << ( BitDepthC – 1).

If for any sample position (x,y) and regions j and k the following conditions are all met:

– pri\_region\_id[ j ] > pri\_region\_id[ k ]  
– x is in (priRegionTopLeftX[ j ] .. priRegionTopLeftX[ j ] + priRegionWidth[ j ])   
– y is in (priRegionTopLeftY[ j ] .. priRegionTopLeftY[ j ] + priRegionHeight[ j ]   
– x is in (priRegionTopLeftX[ k ] .. priRegionTopLeftX[ k ] + priRegionWidth[ k])   
– y is in (priRegionTopLeftY[ k ] .. priRegionTopLeftY[ k ] + priRegionHeight[ k ]

Then the reconstructed target picture sample at position (x, y) should be determined by the parameters signalled for the j-th region.

8.40 Constituent rectangles SEI message

8.40.1 Constituent rectangles SEI message syntax

|  |  |
| --- | --- |
| constituent\_rectangles( payloadSize ) { | Descriptor |
| **cr\_num\_rects\_minus1** | u(12) |
| **cr\_rect\_id\_enabled\_\_flag** | u(1) |
| **cr\_associated\_rect\_id\_enabled\_flag** | u(1) |
| **cr\_rect\_group\_id\_present\_flag** | u(1) |
| if ( cr\_rect\_d\_present\_flag  | |  cr\_associated\_rect\_id\_enabled\_flag  | |   cr\_rect\_group\_id\_present\_flag ) |  |
| **cr\_rect\_id\_len** | u(4) |
| **cr\_rect\_type\_enabled\_flag** | u(1) |
| **cr\_rect\_type\_descriptions\_enabled\_flag** | u(1) |
| **cr\_subpics\_partitioning\_flag** | u(1) |
| if( !cr\_subpics\_partitioning\_flag ) { | u(1) |
| **cr\_rect\_same\_size\_flag** | u(1) |
| if ( cr\_rect\_same\_size\_flag ) { |  |
| **cr\_num\_cols\_minus1** | ue(v) |
| **cr\_num\_rows\_minus1** | ue(v) |
| } else { |  |
| **cr\_log2\_unit\_size** | u(4) |
| **cr\_rect\_size\_len\_minus1** | u(4) |
| } |  |
| } |  |
| for( i = 0; i <= cr\_num\_rects\_minus1; i++ ) { |  |
| if ( cr\_rect type\_enabled\_flag ) { |  |
| **cr\_rect\_type\_present\_flag**[ i ] | u(1) |
| if ( cr\_rect\_type\_present\_flag[ i ] ) |  |
| **cr\_rect\_type\_idc**[ i ] | u(8) |
| } |  |
| if ( cr\_rect type\_idc[ i ] != 255 ) { |  |
| if ( cr\_rect id\_enabled\_flag ) { |  |
| **cr\_rect\_type\_present\_flag**[ i ] | u(1) |
| if ( cr\_rect id\_present\_flag ) |  |
| **cr\_rect\_id**[ i ] | u(v) |
| } |  |
| if ( cr\_associated\_rect id\_enabled\_flag ) { |  |
| **cr\_associated\_rect\_id\_present\_flag**[ i ] | u(1) |
| if ( cr\_associated\_rect\_id\_present\_flag[ i ] ) |  |
| **cr\_associated\_rect\_id**[ i ] | u(v) |
| } |  |
| if ( cr\_group\_id\_present\_flag ) { |  |
| **cr\_rect\_group\_id**[ i ] | u(v) |
| if ( cr\_rect type\_description\_enabled\_flag ) |  |
| **cr\_rect\_type\_description\_present\_flag**[ i ] | u(1) |
| } |  |
| if( !cr\_subpics\_partitioning\_flag &&  !cr\_rects\_same\_size\_flag ) { |  |
| **cr\_rect\_top\_left\_in\_units\_x**[ i ] | u(v) |
| **cr\_rect\_top\_left\_in\_units\_y**[ i ] | u(v) |
| **cr\_rect\_width\_in\_units\_minus1**[ i ] | u(v) |
| **cr\_rect\_height\_in\_units minus1**[ i ] | u(v) |
| } |  |
| } |  |
| if( cr\_rect\_type\_descriptions\_enabled\_flag ) { |  |
| while( !byte\_aligned( ) ) |  |
| **cr\_bit\_equal\_to\_zero** /\* equal to 0 \*/ | f(1) |
| for( i = 0; i <= cr\_num\_rects\_minus1; i++ ) |  |
| if( rect\_type\_description\_present\_flag[ i ] ) |  |
| **cr\_rect\_type\_description**[ i ] | st(v) |
| } |  |
| } |  |

8.40.2 Constituent rectangles SEI message semantics

The constituent rectangles SEI message enables composition of multiple rectangles within a coded picture and provides information about the rectangles, including ID, type, text description, location, and size.

Use of this SEI message requires the definition of the following variables:

– A picture width and picture height in units of luma samples, denoted herein by PicWidthInLumaSamples and PicHeightInLumaSamples, respectively.

– A maximum picture width and maximum picture height in units of luma samples, denoted herein by MaxPicWidth and MaxPicHeight, respectively.

– A chroma format indicator, denoted herein by ChromaFormatIdc, as described in clause 7.3.

– A bit depth for the samples of the luma component, denoted herein by BitDepthY, and when ChromaFormatIdc is not equal to 0, a bit depth for the samples of the two associated chroma components, denoted herein by BitDepthC.

– A subpicture count, denoted by NumSubpics.

– Arrays of the top left X and Y positions of the subpictures, denoted herein by SubPicTopLeftX[ i ] and SubPicTopLeftY[ i ], respectively.

– Arrays of the width and height of the subpictures, denoted herein by SubPicTopLeftX[ i ] and SubPicTopLeftY[ i ], respectively.

If this SEI message is present in any picture unit that is not the first picture unit of a CLVS in decoding order, a constituent rectangles SEI message with the same payload content shall be present in the first picture unit of the CLVS in decoding order.

**cr\_num\_rects\_minus1** plus 1 specifies the number of constituent rectangles for which information is signalled in the SEI message.

**cr\_rect\_id\_enabled\_flag** equal to 1 specifies that the cr\_rect\_id\_present\_flag syntax element is present in the SEI message.cr\_rect\_id\_present\_flag equal to 0 specifies that the cr\_rect\_id\_present\_flag syntax element is not present in the SEI message.

**cr\_associated\_rect\_id\_enabled\_flag** equal to 1 specifies that the cr\_associated\_rect\_id\_present\_flag[ i ] syntax elements are present. cr\_associated\_rect\_id\_enabled\_flag equal to 0 specifies that the cr\_associated\_rect\_id\_present\_flag[ i ] syntax elements are not present.

**cr\_rect\_group\_id\_present\_flag** equal to 1 specifies that cr\_rect\_group\_id[ i ] syntax elements are present. cr\_rect\_group\_present\_flag equal to 0 specifies that\_cr\_rect\_group\_id[ i ] syntax elements are not present.

**cr\_rect\_id\_len** specifies the length of the cr\_rect\_id[ i ], cr\_associated\_rect\_id [ i ], and cr\_rect\_group\_id[ i ] syntax elements. syntax elements.

**cr\_rect\_type\_enabled\_flag** equal to 1 specifies that the cr\_rect\_type\_present\_flag[ i ] syntax element is present in the SEI message.cr\_rect\_type\_enabled\_flag equal to 0 specifies that the cr\_rect\_type\_ present\_flag[ i ] syntax element is not present in the SEI message.

**cr\_rect\_type\_descriptions\_enabled\_flag** equal to 1 specifies that the cr\_rect\_type\_description\_present\_flag[ i ] syntax element is present in the SEI message.cr\_rect\_type\_descriptions\_enabled\_flag equal to 0 specifies that the cr\_rect\_type\_description\_present\_flag[ i ] syntax element is not present in the SEI message.

**cr\_subpics\_partitioning\_flag** equal to 1 indicates that the subpic partitioning parameters in the SPS are used to determine the of constituent rectangle sizes and positions. cr\_subpics\_partitioning\_flag equal to 0 indicates that determination of the constituent rectangle sizes and positions is not based on subpic partitioning parameters in the SPS.

When cr\_subpics\_partitioning\_flag is equal to 1, it is a requirement of bitstream conformance that NumSubpics be greater than or equal to cr\_num\_rects\_minus1 + 1.

**cr\_rect\_same\_size\_flag** equal to 1 indicates that all constituent rectangles have the same size and are arranged in a grid pattern. cr\_rect\_same\_size\_flag equal to 0 indicates that the sizes of the constituent rectangles may differ.

**cr\_num\_cols\_minus1** plus 1and **cr\_num\_rows\_minus1** plus 1 specify the number of columns and rows, respectively, of the constituent rectangle grid when cr\_rect\_same\_size\_flag equal to 1.

The variable crNumCols is set equal to cr\_num\_cols\_minus1 + 1.

The variable crNumRows is set equal to cr\_num\_rows\_minus1 + 1.

**cr\_log2\_unit\_size** specifies a unit size used in variable calculations for the constituent rectangle parameters.

The variable crUnitSize is set equal to 1 << cr\_log2\_unit\_size.

**cr\_rect\_size\_len\_minus1** plus 1 specifies the length of syntax elements cr\_rect\_top\_left\_in\_units\_x[ i ], cr\_rect\_top\_left\_in\_units\_y[ i ], cr\_rect\_width\_in\_units\_minus1[ i ], and cr\_rect\_height\_in\_units minus1[ i ].

**cr\_rect\_type\_present\_flag**[ i ] equal to 1 specifies that the cr\_rect\_type\_idc[ i ] syntax element is present in the SEI message. cr\_rect\_type\_present\_flag[ i ] equal to 0 specifies that the cr\_rect\_type\_idc[ i ] syntax element is not present in the SEI message.

**cr\_rect\_type\_idc**[ i ] indicates the constituent picture type of the i-th rectangle from Table X. When not present and i equal to 0, the value of cr\_rect\_type\_idc[ i ] is inferred to be equal to 0. When not present and i greater than 0, cr\_rect\_type\_idc[ i ] is inferred to be equal to cr\_rect\_type\_idc[ i − 1 ].

**Table X – Mapping of cr\_rect\_type\_idc[ i ] to the type of constituent rectangle**

|  |  |  |
| --- | --- | --- |
| **cr\_rect\_type\_idc[ i ]** | **Name** | **Type of constituent rectangle** |
| 0 | CR\_TEXTURE | Texture |
| 1 | CR\_ALPHA | Alpha plane |
| 2 | CR\_DEPTH | Depth picture |
| 3 | CR\_OBJECT\_MASK | Object mask |
| 4 .. 127 |  | Reserved |
| 128..254 |  | Unspecified |
| 255 | CR\_EMPTY | Empty |

**cr\_rect\_id\_present\_flag** equal to 1 specifies that the cr\_rect\_id[ i ] syntax element is present in the SEI message.cr\_rect\_id\_present\_flag equal to 0 specifies that the cr\_rect\_id[ i ] syntax element is not present in the SEI message.

**cr\_rect\_id**[ i ] indicates the ID of the i-th rectangle. The length of the syntax element is cr\_rect\_id\_len bits. When not present and i equal to 0, the value of cr\_rect\_id[ i ] is inferred to be equal to 0. When not present and i greater than 0, the value of cr\_rect\_id[ i ] is inferred to be equal to cr\_rect\_id[ i – 1 ] + 1.

It is a requirement of bitstream conformance that when j not equal to k, cr\_rect\_id[ j ] shall not be equal to cr\_rect\_id[ k ].

**cr\_associated\_rect\_id\_present\_flag** equal to 1 specifies that the cr\_associated\_rect\_id[ i ] syntax element is present in the SEI message.cr\_rect\_id\_present\_flag equal to 0 specifies that the cr\_associated\_rect\_id[ i ] syntax element is not present in the SEI message.

**cr\_associated\_rect\_id**[ i ] indicates the ID of an associated primary rectangle of the i-th rectangle. The length of the syntax element is cr\_rect\_id\_len bits.

It is a requirement of bitstream conformance that cr\_associated\_rect\_id[ i ] equal cr\_rect\_id[ j ] for some value of j in 0 .. cr\_num\_rects\_minus1.

When not present, the value of cr\_associated\_rect\_id[ i ] is undefined.

**cr\_rect\_group\_id**[ i ] indicates the group ID of the i-th rectangle. The length of the syntax element is cr\_rect\_id\_len bits. When not present, the value of cr\_rect\_group\_id[ i ] is inferred to be equal to 0.

**cr\_rect\_type\_description\_present\_flag**[ i ] equal to 1 specifies that the cr\_rect\_type\_description[ i ] syntax element is present in the SEI message. cr\_rect\_type\_description\_present\_flag[ i ] equal to 0 specifies that the cr\_rect\_type\_description[ i ] syntax element is not present in the SEI message. When not present, the value of cr\_rect\_type\_description\_present\_flag[ i ] is inferred to be equal to 0.

**cr\_rect\_top\_left\_in\_units\_x**[ i ] and **cr\_rect\_top\_left\_in\_units\_y**[ i ], when present, indicate the horizontal and vertical positions, respectively, of the top left position of the i-th constituent picture rectangle in units.

The variables SubWidthC and SubHeightC are derived from ChromaFormatIdc as specified by Table 2.

**cr\_rect\_width\_in\_units\_minus1**[ i ] plus 1 and **cr\_rect\_height\_in\_units\_minus1**[ i ] plus 1, when present, indicate the width and height, respectively, of the i-th constituent rectangle in units. The length of the syntax elements are cr\_rect\_size\_len\_minus1 + 1.

The variables crRectTopLeftX[ i ] and crRectTopLeftY[ i ], representing the x and y location, respectively, and variables crRectWidth[ i ] and crRectHeight[ i ], representing the width and height, respectively, of the i-th constituent rectangle are derived as follows.

If cr\_subpics\_partitioning\_flag is equal to 0 and cr\_rect\_same\_size\_flag is equal to 0, the following applies:

– The variable crRectTopLeftX[ i ] is set equal to cr\_rect\_top\_left\_in\_units\_x[ i ] \* crUnitSize

– The variable crRectTopLeftY[ i ] is set equal to cr\_rect\_top\_left\_in\_units\_y[ i ] \* crUnitSize

– The variable crRectWidth[ i ] is set equal to ( cr\_rect\_width\_in\_units minus1 + 1 ) \* crUnitSize

– The variable crRectHeight[ i ] is set equal to ( cr\_rect\_height\_in\_units minus1 + 1 ) \* crUnitSize

Otherwise, if cr\_subpics\_partitioning\_flag is equal to 1, the following applies:

– The variable crRectTopLeftX[ i ] is set equal to SubPicTopLeftX[ i ]

– The variable crRectTopLeftY[ i ] is set equal to SubPicTopLeftY[ i ]

– The variable crRectWidth[ i ] is set equal to SubPicWidth[ i ]

– The variable crRectHeight[ i ] is set to equal to SubPicHeight[ i ]

Otherwise (cr\_rect\_same\_size\_flag is equal to 1), the following applies:

– The variable crRectTopLeftX[ i ] is set equal to (i % crNumCols ) \* MaxPicWidth / crNumCols

– The variable crRectTopLeftY[ i ] is set equal to (i / crNumCols ) \* MaxPicHeight / crNumRows

– The variable crRectWidth [ i ] is set equal to MaxPicWidth / crNumCols

– The variable crRectHeight [ i ] is set to equal to MaxPicHeight / crNumRows

When PicWidthInLumaSamples is not equal to MaxPicWidth, the following applies:

– crRectTopLeftX[ i ] is set equal to (crRectTopLeftX[ i ] \* PicWidthInLumaSamples + MaxPicWidth / 2 ) / MaxPicWidth

– crRectWidth[ i ] is set equal to (crRectWidth[ i ] \* PicWidthInLumaSamples + MaxPicWidth / 2 ) / MaxPicWidth

When PicHeightInLumaSamples is not equal to MaxPicHeight, the following applies:

– crRectTopLeftY[ i ] is set equal to (crRectTopLeftY[ i ] \* PicWidthInLumaSamples + MaxPicHeight / 2 ) / MaxPicHeight

– crRectHeight[ i ] is set equal to (crRectHeight[ i ] \* PicWidthInLumaSamples + MaxPicHeight / 2 ) / MaxPicHeight

It is a requirement of bitstream conformance that for each sample position (x, y) in the coded picture there shall be at most one rectangle, j, for which both of the following conditions apply:

– x is in (crRectTopLeftX[ j ] .. crRectTopLeftX[ j ] + crRectWidth[ j ] - 1)

– y is in (crRectTopLeftY[ j ] .. crRectTopLeftY[ j ] + crRectHeight[ i ] - 1)

It is a requirement of bitstream conformance that crRectTopLeftX[ i ] % SubWidthC shall be equal to 0, crRectTopLeftX [ i ] % SubHeightC shall be equal to 0, crRectWidth[ i ] % SubWidthC shall be equal to 0, and crRectHeight[ i ] % SubHeightC shall be equal to 0.

**cr\_bit\_equal\_to\_zero** shall be equal to 0.

**cr\_rect\_type\_description**[ i ] specifies a text description of the constituent rectangle. The length of the syntax element shall be less than or equal to 4097 bytes, not including the null termination byte.

8.41 Quality metrics SEI message

8.41.1 Quality metrics SEI message syntax

|  |  |
| --- | --- |
| **quality\_metric( payloadSize ) {** | **Descriptor** |
| **qm\_metric\_definitions\_present\_flag** | u(1) |
| **qm\_clvs\_values\_present\_flag** | u(1) |
| **qm\_pic\_values\_present\_flag** | u(1) |
| **qm\_num\_metrics\_minus1** | u(4) |
| if ( qm\_metric\_definitions\_present\_flag ) { |  |
| **qm\_gain\_enabled\_flag** | u(1) |
| for( i = 0; i <= qm\_num\_metrics\_minus1; i++ ) { |  |
| if ( qm\_gain\_enabled\_flag ) |  |
| **qm\_gain\_flag**[ i ] | u(1) |
| if ( qm\_gain\_flag[ i ] ) |  |
| **qm\_gain\_reference\_flag**[ i ] | u(1) |
| **qm\_metric\_type**[ i ] | u(8) |
| **qm\_three\_component\_flag**[ i ] | u(1) |
| if ( qm\_metric\_type[ i ] = = 0 | | qm\_metric\_type[ i ] >= 128) { |  |
| **qm\_metric\_increasing\_flag**[ i ] | u(1) |
| **qm\_full\_reference\_flag**[ i ] | u(1) |
| **qm\_value\_len\_minus1\_in\_bytes**[ i ] | u(2) |
| **qm\_metric\_description\_present\_flag**[ i ] | u(1) |
| } |  |
| } |  |
| } |  |
| while( !byte\_aligned( ) ) |  |
| **qm\_bit\_equal\_to\_zero** /\* equal to 0 \*/ | f(1) |
| for( i = 0; i <= qm\_num\_metrics\_minus1; i++ ) { |  |
| if ( qm\_metric\_description\_present\_flag[ i ] ) |  |
| **qm\_metric\_description**[ i ] | st(v) |
| if( qm\_clvs\_values\_present\_flag ) |  |
| for( i = 0; i <= qm\_num\_metrics\_minus1; i++ ) |  |
| for( c = 0; c < ( qm\_three\_component\_flag ? 3 : 1 ); c++ ) |  |
| **qm\_clvs\_metric\_value**[ i ][ c ] | u(v) |
| if( qm\_pic\_values\_present\_flag ) |  |
| for( i = 0; i <= qm\_num\_metrics\_minus1; i++ ) |  |
| for( c = 0; c < ( qm\_three\_component\_flag ? 3 : 1 ); c++ ) |  |
| **qm\_pic\_metric\_value**[ i ][ c ] | u(v) |
| } |  |

8.41.2 Quality metrics SEI message semantics

The quality metrics SEI message signals quality metric values indicating any of the following:

– The quality of a single picture.

– The mean quality of all the pictures corresponding to a CLVS.

– The quality gain of a single picture, which is difference between the quality of a single picture relative to the quality of a gain reference picture.

– The mean quality gain of all the pictures corresponding to a CLVS.

Use of this SEI message requires the definition of the following variables:

– A chroma format indicator, denoted herein by ChromaFormatIdc, as described in subclause ‎7.3.

– A count of pictures NumPics.

– Lists of picture widths and heights in units of luma samples, denoted herein by PicWidth[ i ] and PicHeight[ i ], respectively, where i is in the range of 0 to NumPics − 1, inclusive.

– A list of pictures TestPicList[ i ] where i is in the range of 0 to NumPics − 1, inclusive.

– When any qm\_gain\_flag[ i ] is equal to 1, a list of gain reference pictures GainRefPicList[ i ] for i is in the range of 0 to NumPics − 1, inclusive.

The variables SubWidthC and SubHeightC are derived from ChromaFormatIdc as specified by Table 2.

TestPicList[ i ][ cIdx ] and ReferencePicList[ i ][ cIdx ] denote the cIdx-th sample array of the i-th picture among TestPicList and ReferencePicList, respectively.

TestPicList[ i ][ cIdx ][ x ][ y ] and ReferencePicList[ i ][ cIdx ][ x ][ y ] denote the sample at location ( x, y ) within the cIdx-th sample array of the i-th picture among TestPicList and ReferencePicList, respectively, where x is in the range of 0 to ( ( cIdx = = 0 ) ? PicWidth[ i ] : PicWidth[ i ] / SubWidthC ) − 1, inclusive and y is in the range of 0 to ( ( cIdx = = 0 ) ? PicHeight[ i ] : PicHeight[ i ] / SubHeightC ) − 1, inclusive.

Let currPicIdx have a value such that the output time of TestPicList[ currPicIdx ] is equal to the output time of the current picture.

**qm\_metric\_definitions\_present\_flag** equal to 1 specifies that information defining the quality metrics is present. qm\_metric\_definitions\_present\_flag equal to 0 specifies that information defining the quality metrics is not present.

When this SEI message is the first quality metrics SEI message in a CLVS, in decoding order, qm\_metric\_definitions\_present\_flag shall be equal to 1.

Otherwise, (this SEI message is not the first quality metrics SEI message in a CLVS, in decoding order,) it is a requirement of bitstream conformance that at least one of the two following conditions shall be met:

– qm\_metric\_definitions\_present\_flag is equal to 0

– the values of the qm\_metric\_type[ i ], qm\_three\_component\_flag[ i ], qm\_gain\_flag[ i ], qm\_gain\_reference\_flag[ i ], qm\_metric\_increasing\_flag[ i ], qm\_full\_reference\_flag[ i ], qm\_value\_len\_minus1\_in\_bytes[ i ], qm\_metric\_description\_present\_flag[ i ], and qm\_metric\_description[ i ] syntax elements, when present, shall be equal to the respective syntax elements in the first quality metric SEI message in the CLVS

**qm\_clvs\_values\_present\_flag** equal to 1 specifies that qm\_clvs\_metric\_value[ i ][ c ] syntax elements are present. qm\_clvs\_values\_present\_flag equal to 0 specifies qm\_clvs\_metric\_value[ i ][ c ] syntax elements are not present.

**qm\_pic\_values\_present\_flag** equal to 1 specifies that qm\_pic\_metric\_value[ i ][ c ] syntax elements are present. qm\_pic\_values\_present\_flag equal to 0 specifies that qm\_pic\_metric\_value[ i ][ c ] syntax elements are not present.

**qm\_num\_metrics\_minus1** plus 1 specifies the number of quality metric entries signaled.

**qm\_gain\_enabled\_flag** equal to 1 specifies that the qm\_gain\_flag[ i ] syntax element is present. qm\_gain\_enabled\_flag equal to 0 specifies that the qm\_gain\_flag[ i ] syntax element is not present.

**qm\_gain\_flag**[ i ] and **qm\_gain\_reference\_flag**[ i ], when present, indicate the interpretation of the values of the qm\_clvs\_metric\_value[ i ][ c ] and qm\_pic\_metric\_value[ i ][ c ] syntax elements in clause 8.41.2.1. When qm\_gain\_flag[ i ] and qm\_gain\_reference\_flag[ i ] are not present, they are inferred to be equal to 0.

**qm\_metric\_type**[ i ] specifies the type of the quality metric of the i-th entry, as specified in Table X. The value of qm\_metric\_type[ i ] shall be in the range of 0..8, inclusive, or in the range of 128..255, inclusive, in bitstreams conforming to this version of the Specification. Values in the range of of 9..127, inclusive, for qm\_metric\_type[ i ] are reserved for future use by ITU-T | ISO/IEC and shall not be present in bitstreams conforming to this version of this Specification.

When the value of qm\_metric\_type [ i ] is in the range of 9 .. 127, decoders conforming to this version of this Specification shall ignore all the syntax elements for the i-th entry in this syntax structure.

When the value of qm\_metric\_type[ i ] is in the range of 128 .. 255, the quality metric type is unspecified or specified by other means not specified in this Specification.

**Table X. Interpretation of qm\_metric\_type[ i ]**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **t** | **Description** | **IncreasingFlag[t]** | **FullReferenceFlag[t]** | **NumBytes[t]** |
| 0 | User-defined |  |  |  |
| 1 | PSNR [2] | 1 | 1 | 2 |
| 2 | PSNR-YUV [2] | 1 | 1 | 2 |
| 3 | SSIM [2] | 1 | 1 | 2 |
| 4 | MS-SSIM [1] | 1 | 1 | 2 |
| 5 | MOS [1] | 1 | 0 | 1 |
| 6 | wPSNR [2] | 1 | 1 | 2 |
| 7 | WS-PSNR [2] | 1 | 1 | 2 |
| 8 | MSE | 0 | 1 | 2 |
| 9..127 | Reserved |  |  |  |
| 128..255 | Unspecified |  |  |  |

**qm\_three\_component\_flag**[ i ] equal to 1 indicates that 3 component values will be present for the i-th metric. qm\_three\_component\_flag[ i ] equal to 0 indicates that a single value will be present for the i-th metric. It is a requirement of bitstream conformance that when ChromaFormatIdc is equal to 0, qm\_three\_component\_flag[ i ] shall be equal to 0.

**qm\_metric\_increasing\_flag**[ i ] equal to 1 indicates that a higher value of the i-th metric value represents an improvement in quality. qm\_metric\_increasing\_flag[ i ] equal to 0 indicates that a lower value of the i-th metric value represents an improvement in quality. When not present, the value of qm\_metric\_increasing\_flag[ i ] is inferred to be equal to IncreasingFlag[qm\_metric\_type[ i ] ] in Table X.

**qm\_full\_reference\_flag**[ i ] equal to 1 indicates that the quality metric is a full reference quality metric, calculated by comparison of the pictures in TestPicList with the respective quality reference pictures. qm\_full\_reference\_flag[ i ] equal to 0 indicates that the quality metric may or may not include comparison of the pictures in TestPicList with the respective quality reference picture. When not present, the value of qm\_full\_reference\_flag[ i ] is inferred to be equal to FullReferenceFlag[qm\_metric\_type[ i ] ] in Table X.

**qm\_value\_len\_minus1\_in\_bytes**[ i ] plus 1 specifies the length in bytes of the qm\_pic\_metric\_value[ i ][ c ] syntax element. When not present, the qm\_value\_len\_minus1\_in\_bytes[ i ] is inferred to be equal to NumBytes[ [qm\_metric\_type[ i ] ] – 1.

**qm\_metric\_description\_present\_flag**[ i ] equal to 1 specifies that qm\_metric\_description[ i ] is present. qm\_metric\_description\_present\_flag[ i ] equal to 0 specifies that qm\_metric\_description[ i ] is not present.

**qm\_bit\_equal\_to\_zero** shall be equal to 0.

**qm\_metric\_description**[ i ] specifies a text description of the i-th quality metric. The length of the syntax element shall be less than or equal to 4097 bytes, not including the null termination byte.

**qm\_clvs\_metric\_value**[ i ][ c ] specifies the mean value of the i-th quality metric for the c-th component of the CLVS. The length of the syntax element is 8 \* (qm\_value\_len\_minus1\_in\_bytes[ i ] + 1) bits.

**qm\_pic\_metric\_value**[ i ][ c ] specifies the value of the i-th quality metric for the c-th component of the current picture. The length of the syntax element is 8 \* (qm\_value\_len\_minus1\_in\_bytes[ i ] + 1) bits.

The semantics of the quality metric value are determined by the values of qm\_metric\_type[ i ], qm\_gain\_flag[ i ], and qm\_gain\_reference\_flag[ i ].

When qm\_pic\_values\_present\_flag is equal to 1, qm\_pic\_metric\_value[ i ][ c ] indicates the picture metric value, picMetricValue[ i ][ c ], of type qm\_metric\_type[ i ] as described in Table X, and the following applies:

– If qm\_gain\_flag[ i ] is equal to 0, qm\_pic\_metric\_value[ i ][ c ] has value derived by the process specified in clause X.X with testPic, picWidth, and picHeight assigned to be TestPicList[ currPicIdx ], PicWidth[ currPicIdx ], and PicHeight[ currPicIdx ], respectively.

– Otherwise (qm\_gain\_flag[ i ] is equal to 1), the following applies:

– picMetricValueTest[ i ][ c ] is set equal to picMetricValue[ i ][ c ] derived by the process specified in clause X.X with testPic, picWidth, and picHeight assigned to be TestPicList[ currPicIdx ], PicWidth[ currPicIdx ], and PicHeight[ currPicIdx ], respectively.

– picMetricValueGainRef[ i ][ c ] is set equal to picMetricValue[ i ][ c ] derived by the process specified in clause X.X with testPic, picWidth, and picHeight assigned to be GainRefPicList[ currPicIdx ], PicWidth[ currPicIdx ], and PicHeight[ currPicIdx ], respectively.

– qm\_pic\_metric\_value[ i ][ c ] has the value picMetricValueTest[ i ][ c ] − picMetricValueGainRef[ i ][ c ].

When qm\_clvs\_values\_present\_flag is equal to 1, qm\_clvs\_metric\_value[ i ][ c ] indicates the mean value of the picture metric values, listPicMetricValue[ j ][ i ][ c ], calculated over all pictures in TestPicList, of type qm\_metric\_type[ i ] as described in Table X, where each listPicMetricValue[ j ][ i ][ c ] is derived as follows for each value of j in the range of 0 to NumPics − 1, inclusive:

– If qm\_gain\_flag[ i ] is equal to 0, listPicMetricValue[ j ][ i ][ c ] is equal to the picMetrictValue[ i ][ c ] derived by the process specified in clause X.X is performed with testPic, picWidth, and picHeight assigned to be TestPicList[ j ], PicWidth[ j ], and PicHeight[ j ], respectively.

– Otherwise (qm\_gain\_flag[ i ] is equal to 1), the following applies:

– listPicMetricValueTest[ j ][ i ][ c ] is the set equal to picMetricValue[ i ][ c ] derived by the process specified in clause X.X with testPic, picWidth, and picHeight assigned to be TestPicList[ currPicIdx ], PicWidth[ currPicIdx ], and PicHeight[ currPicIdx ], respectively.

– picMetricValueGainRef[ j ][ i ][ c ] is the set equal to picMetricValue[ i ][ c ] derived by the process specified in clause X.X with testPic, picWidth, and picHeight assigned to be GainRefPicList[ currPicIdx ], PicWidth[ currPicIdx ], and PicHeight[ currPicIdx ], respectively.

– qm\_pic\_metric\_value[ j ][ i ][ c ] has the value picMetricValueTest[ j ][ i ][ c ] − picMetricValueGainRef[ j ][ i ][ c ].

8.41.2.1 Process for derivation of picMetricValue[ i ][ c ]

Inputs to this process are a tested picture testPic, a picture width picWidth in luma samples, and a picture height picHeight in luma samples.

Let a quality reference picture referencePic be the original picture that was given as input to the encoding system and has the output time equal to the output time of testPic.

testPic[ cIdx ] and referencePic[ cIdx ] denote the cIdx-th sample array of the testPic and referencePic, respectively.

testPic[ cIdx ][ x ][ y ] and referencePic[ cIdx ][ x ][ y ] denote the sample at location ( x, y ) within the cIdx-th sample array of testPic and referencePic, respectively.

The pictures quality metric, picMetricValue[ i ][ c ] is derived as follows:

– When qm\_metric\_type[ i ] is equal to 0,

– listPicMetricValueTest[ j ][ i ][ c ] is the set equal to picMetricValue[ i ][ c ] derived by the process specified in clause X.X with testPic, picWidth, and picHeight assigned to be TestPicList[ currPicIdx ], PicWidth[ currPicIdx ], and PicHeight[ currPicIdx ], respectively.

– picMetricValueGainRef[ j ][ i ][ c ] is the set qual to picMetricValue[ i ][ c ] derived by the process specified in clause X.X with testPic, picWidth, and picHeight assigned to be GainRefPicList[ currPicIdx ], PicWidth[ currPicIdx ], and PicHeight[ currPicIdx ], respectively.

– When qm\_metric\_increasing\_flag[ i ] is equal 1, a higher value of picMetricValue[ i ][ c ] indicates testPic is of better quality than for a picture with a lower value of picMetricValue[ i ][ c ].

– When qm\_full\_reference\_flag[ i ] is equal to 1, of picMetricValue[ i ][ c ] indicates a quality metric value calculated from a comparison of testPic with referencePic.

– qm\_metric\_description[ i ] provides a text description of the quality metric indicated by picMetricValue[ i ][ c ].

– Additional interpretation of picMetricValue[ i ][ c ] is determined by external means not specified in this Specification.

– When qm\_metric\_type[ i ] is equal to 1,

– picMetricValue[ i ][ 0 ] is set equal to the PSNR value calculated using clauses 9.4.2 and D.2 of ISO/IEC 23001-11 [2] for the luma components of testPic and referencePic, with bit depth OrigBitDepth, width picWidth, and height picHeight.

– When qm\_three\_component\_flag[ i ] is equal to 1,

– picMetricValue[ i ][ 1 ] and picMetricValue[ i ][ 2 ] are set equal to the PSNR values calculated using clauses 9.4.2 and D.2 of ISO/IEC 23001-11 [2] for the Cb and Cr components, respectively, of testPic and referencePic, with bit depth OrigBitDepth width picWidth/SubWidthC, and height picHeight/SubHeightC.

– When qm\_metric\_type[ i ] is equal to 2,

– picMetricValue[ i ][ 0 ] is set equal to the value of the variable psnrYUV calculated as follows:

– The variable psnrY is set equal to the PSNR value calculated using clauses 9.4.2 and D.2 of ISO/IEC 23001-11 [2] for the luma components of testPic and referencePic, with bit depth OrigBitDepth width picWidth, and height picHeight.

– The variables psnrU and psnrV are set equal to the PSNR values calculated using clauses 9.4.2 and D.2 of ISO/IEC 23001-11 [2] for the Cb and Cr components, respectively, of testPic and referencePic, with bit depth OrigBitDepth, width picWidth/SubWidthC, and height picHeight/SubHeightC.

psnrYUV = (10\*psnrY + psnrU + psnrV ) ÷ 12

– When qm\_metric\_type[ i ] is equal to 3,

– picMetricValue[ i ][ 0 ] is set equal to the value of SSIM calculated from clauses 9.4.2 and D.5 of ISO/IEC 23001-11 [2] for the luma components of testPic and referencePic, with bit depth OrigBitDepth width picWidth, and height picHeight.

– When qm\_three\_component\_flag[ i ] is equal to 1,

– picMetricValue[ i ][ 1 ] and picMetricValue[ i ][ 2 ] are set equal to the SSIM values calculated from clauses 9.4.2 and D.5 of ISO/IEC 23001-11 [2] for the Cb and Cr components, respectively, of testPic and referencePic, with bit depth OrigBitDepth, width picWidth/SubWidthC, and height picHeight/SubHeightC.

– When qm\_metric\_type[ i ] is equal to 4,

– picMetricValue[ i ][ 0 ] is set equal to the value of MS-SSIM calculated from clause 4.3.3 of ISO/IEC 23001-10 [1] for the luma components of testPic and referencePic, with bit depth OrigBitDepth.

– When qm\_three\_component\_flag[ i ] is equal to 1,

– picMetricValue[ i ][ 1 ] and picMetricValue[ i ][ 2 ] are set equal to the MS-SSIM values calculated from clause 4.3.3 of ISO/IEC 23001-10 [1] for the Cb and Cr components, respectively, of testPic and referencePic, with bit depth OrigBitDepth.

– When qm\_metric\_type[ i ] is equal to 5,

– picMetricValue[ i ][ 0 ] is set equal to the value of MOS specified in clause 4.3.6 of ISO/IEC 23001-10 [1].

– When qm\_metric\_type[ i ] is equal to 6,

– picMetricValue[ i ][ 0 ] is set equal to the value of wPSNR calculated from clauses 9.4.2 and D.3 of ISO/IEC 23001-11 [2] for the luma components of testPic and referencePic, with bit depth OrigBitDepth width picWidth, and height picHeight.

– When qm\_three\_component\_flag[ i ] is equal to 1,

– picMetricValue[ i ][ 1 ] and picMetricValue[ i ][ 2 ] are set equal to the wPSNR values calculated from clauses 9.4.2 and D.3 of ISO/IEC 23001-11 [2] for the Cb and Cr components, respectively, of testPic and referencePic, with bit depth OrigBitDepth, width picWidth/SubWidthC, and height picHeight/SubHeightC.

– When qm\_metric\_type[ i ] is equal to 7,

– picMetricValue[ i ][ 0 ] is set equal to the value of WS-PSNR calculated from clauses 9.4.2 and D.4 of ISO/IEC 23001-11 [2] for the luma components of testPic and referencePic, with bit depth OrigBitDepth, width picWidth, and height picHeight.

– When qm\_three\_component\_flag[ i ] is equal to 1,

– picMetricValue[ i ][ 1 ] and picMetricValue[ i ][ 2 ] are set equal to the WS-PSNR values calculated from clauses 9.4.2 and D.4 of ISO/IEC 23001-11 [2]for the Cb and Cr components, respectively, of testPic and referencePic, with bit depth OrigBitDepth, width picWidth/SubWidthC, and height picHeight/SubHeightC.

– When qm\_metric\_type[ i ] is equal to 8,

– picMetricValue[ i ][ 0 ] is set equal to the value of lumaMse, interpreted as a floating-point value, derived as follows:

lumaSse = 0

for( y = 0; y < picHeight; y++)  
 for( x = 0; x < picWidth; x++ )   
 lumaSse += (testPic[ 0 ][ x ][ y ] – referencePic[ 0 ][ x ][ y ])2

lumaMse = ( lumaSse ÷ (CroppedHeight \* CroppedWidth) )/ 100

– When qm\_three\_component\_flag[ i ] is equal to 1, picMetricValue[ i ][ 1 ] and picMetricValue[ i ][ 2 ] are set equal to the values of CbMse and CrMse, respectively, interpreted as floating-point values, derived as follows:

CbSse = 0

CrSse = 0

for( y = 0; y < picHeight / SubWidthC; y++)  
 for( x = 0; x < picWidth / SubWidthC; x++ ) {  
 CbSse += (testPic[1][y][x] – referencePic[1][y][x])2

CbSse += (testPic[1][y][x] – referencePic[1][y][x])2

}

CbMse = ( CbSse ÷ ( picHeight \* CroppedWidth / (SubWidthC \* SubWidthC) ) / 100

CrMse = ( CrSse ÷ ( picHeight \* CroppedWidth / (SubWidthC \* SubWidthC) ) / 100

8.42 Lens optical correction SEI message

[Ed. (GT): To be checked: validity of cancel flag and persistence flag, relation with conformance cropping, behavior with frame packing.]

8.42.1 Lens optical correction SEI message syntax

|  |  |
| --- | --- |
| lens\_optical\_correction( payloadSize ) { | **Descriptor** |
| **loc\_cancel\_flag** | u(1) |
| if( !loc\_cancel\_flag ) { |  |
| **loc\_focal\_parameters\_present\_flag** |  |
| if( loc\_focal\_parameters\_present\_flag ) { |  |
| **loc\_focal\_center\_x** | u(16) |
| **loc\_focal\_center\_y** | u(16) |
| **loc\_focal\_length** | u(16) |
| } |  |
| **loc\_rd\_parameters\_present\_flag** | u(1) |
| if( loc\_rd\_parameters\_present\_flag ) { |  |
| **loc\_rd\_model\_id** | u(8) |
| if( loc\_rd\_model\_id = = 0 ) { |  |
| **loc\_rd\_poly3\_k1** | i(25) |
| } |  |
| if( loc\_rd\_model\_id = = 1 ) { |  |
| **loc\_rd\_poly5\_k1** | i(25) |
| **loc\_rd\_poly5\_k2** | i(25) |
| } |  |
| } |  |
| **loc\_tca\_parameters\_present\_flag** | u(1) |
| if( loc\_tca\_parameters\_present\_flag ) { |  |
| **loc\_tca\_model\_id** | u(8) |
| if( loc\_tca\_model\_id = = 0 ) { |  |
| **loc\_tca\_kr** | u(25) |
| **loc\_tca\_kb** | u(25) |
| } |  |
| if( loc\_tca\_model\_id = = 1 ) { |  |
| **loc\_tca\_br** | i(25) |
| **loc\_tca\_cr** | i(25) |
| **loc\_tca\_vr** | u(25) |
| **loc\_tca\_bb** | i(25) |
| **loc\_tca\_cb** | i(25) |
| **loc\_tca\_vb** | u(25) |
| } |  |
| } |  |
| **loc \_persistence\_flag** | u(1) |
| } |  |
| } |  |

8.42.2 Lens optical correction SEI message semantics

The Lens optical correction SEI message provides the decoder with a lens distortion model to enable image correction.

**loc\_cancel\_flag** equal to 1 indicates that the SEI message cancels the persistence of any previous lens optical correction SEI message in output order that applies to the current layer. loc\_cancel\_flag equal to 0 indicates that lens optical correction follows.

**loc\_focal\_parameters\_present\_flag** equal to 1 indicates that focal parameters of the lens are present in the lens optical correction SEI message syntax. loc\_focal\_parameters\_present\_flag equal to 0 indicates that no focal parameter is present.

**loc\_focal\_center\_x** defines the horizontal location in the picture of the focal point. It is by default half of the width of the picture.

**loc\_focal\_center\_y** defines the vertical location in the picture of the focal point. It is by default half of the height of the picture.

**loc\_focal\_length** defines the focal length of the lens, i.e., the distance from the lens in which rays of light that fall parallel to the optical axis into the lens are converged. It is expressed in units of 0.1 millimeters. A value of 0 indicates that the focal length is undefined. When not present, the value of loc\_focal\_length is inferred to be equal to 0.

**loc\_rd\_parameters\_present\_flag** equal to 1 indicates that radial distortion parameters are present in the lens optical correction SEI message syntax. loc\_rd\_parameters\_present\_flag equal to 0 indicates that no radial correction is expected.

**loc\_rd\_model\_id** identifies the type of radial distortion model as specified in Table 1. The value of loc\_rd\_model\_id shall be in the range of 0 to 1, inclusive. The values greater than 1 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 lens optical correction SEI messages with loc\_rd\_model\_id greater than 1.

Table 1 – loc\_rd\_model\_id

|  |  |
| --- | --- |
| Value | Description |
| 0 | Poly3 |
| 1 | Poly5 |

When loc\_rd\_model\_id equals to 0 the polynomial 3rd order (Poly3) model is specified as

rd = ru \* ( 1 − k1 + k1 \* ru \* ru ) (X-1)

with rd referring to the distorted radius (normalized distance to image center), ru referring to the corrected undistorted radius, and k1 being the distortion coefficient.

When loc\_rd\_model\_id equals to 1 the polynomial 5th (Poly5) order model is specified as

rd = ru \* ( 1 − k1 \* ru \* ru + k2 \* ru \* ru\* ru\* ru ) (X-2)

with rd referring to the distorted radius (normalized distance to image center), ru referring to the corrected undistorted radius, and k1 and k2 being the distortion coefficients.

**loc\_**rd\_poly3\_k1 indicates the value of the k1 parameter from the Poly3 radial distortion model from equation (X-1). When not present, the value of loc\_rd\_poly3\_k1 is inferred to be equal to 0.

**loc\_**rd\_poly5\_k1 indicates the value of the k1 parameter from the Poly5 radial distortion model from equation (X-2). When not present, the value of loc\_rd\_poly5\_k1 is inferred to be equal to 0.

**loc\_rd**\_poly5\_k2 indicates the value of the k2 parameter from the Poly5 radial distortion model from equation (X-2). When not present, the value of loc\_rd\_poly5\_k2 is inferred to be equal to 0.

**loc\_tca\_parameters\_present\_flag** equal to 1 indicates that transversal chromatic aberration parameters are present in the lens optical correction SEI message syntax. loc\_tca\_parameters\_parameters\_present\_flag equal to 0 indicates that no transversal chromatic aberration correction is expected.

**loc\_tca\_model\_id** identifies the type of radial distortion model as specified in Table 2. The value of loc\_tca\_model\_id shall be in the range of 0 to 1, inclusive. The values greater than 1 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 lens optical correction SEI messages with loc\_tca\_model\_id greater than 1.

Table 2 – loc\_tca\_model\_id

|  |  |
| --- | --- |
| Value | Description |
| 0 | Linear |
| 1 | Poly3 |

When loc\_tca\_model\_id equals to 0 the linear model is specified as

rd,R = ru,R \* kR (X-3)

rd,B = ru,B \* kB (X-4)

with rd,R and rd,B referring to the distorted radius (normalized distance to image center) of the red and blue components respectively, ru,R and ru,B referring to the corrected undistorted radius of the red and blue components respectively, and kR and kB being the transversal chromatic aberration coefficients of the red and blue components respectively. kR and kB are equal to 1 by default.

NOTE – The transversal chromatic aberration is corrected in the linear model on the red and blue components with the green component as the reference.

When loc\_tca\_model\_id equals to 1 the polynomial 3rd order (Poly3) model is specified as

rd,R = ru,R \* ( bR \* ru,R \* ru,R + cR \* ru,R + vR ) (X-5)

rd,B = ru,B \* ( bB \* ru,B \* ru,B + cB \* ru,B + vB ) (X-6)

with rd,R and rd,B referring to the distorted radius (normalized distance to image center) of the red and blue components respectively, ru,R and ru,B referring to the corrected undistorted radius of the red and blue components respectively, and {bR, cR, vR} and {bB, cB, vB} being the transversal chromatic aberration coefficients of the red and blue components respectively. The bR, cR, bB,and cB coefficients are by default equal to 0, and the vR and vB coefficients are by default equal to 1.

NOTE - The transversal chromatic aberration is corrected in the Poly3 model on the red and blue components with the green component as the reference.

**loc\_**tca\_kr indicates the value of the kR parameter from the linear transversal chromatic aberration model from equation (X-3). When not present, the value of loc\_tca\_kr is inferred to be equal to .

**loc\_**tca\_kb indicates the value of the kB parameter from the linear transversal chromatic aberration model from equation (X-4). When not present, the value of loc\_tca\_kr is inferred to be equal to .

**loc\_**tca\_br indicates the value of the bB parameter from the Poly3 transversal chromatic aberration model from equation (X-5). When not present, the value of loc\_tca\_br is inferred to be equal to 0.

**loc\_**tca\_**c**r indicates the value of the cR parameter from the Poly3 transversal chromatic aberration model from equation (X-5). When not present, the value of loc\_tca\_cr is inferred to be equal to 0.

**loc\_**tca\_vr indicates the value of the vR parameter from the Poly3 transversal chromatic aberration model from equation (X-5). When not present, the value of loc\_tca\_vr is inferred to be equal to .

**loc\_**tca\_bb indicates the value of the bB parameter from the Poly3 transversal chromatic aberration model from equation (X-6). When not present, the value of loc\_tca\_bb is inferred to be equal to 0.

**loc\_**tca\_cb indicates the value of the cB parameter from the Poly3 transversal chromatic aberration model from equation (X-6). When not present, the value of loc\_tca\_cb is inferred to be equal to 0.

**loc\_**tca\_vb indicates the value of the vB parameter from the Poly3 transversal chromatic aberration model from equation (X-6). When not present, the value of loc\_tca\_vb is inferred to be equal to .

**loc\_persistence\_flag** specifies the persistence of the lens optical correction SEI message for the current layer.

loc\_persistence\_flag equal to 0 specifies that the lens optical correction SEI message applies to the current decoded picture only.

loc\_persistence\_flag equal to 1 specifies that the lens optical correction SEI message applies to the current decoded picture and persists for all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture in the current layer in an AU associated with a lens optical correction SEI message is output that follows the current picture in output order.

8.43 Display overlays SEI message

8.43.1 Display overlays SEI message syntax

|  |  |
| --- | --- |
| display\_overlays\_info( payloadSize ) { | **Descriptor** |
| **doi\_id** | u(6) |
| **doi\_cancel\_flag** | u(1) |
| if( !doi\_cancel\_flag ) { |  |
| **doi\_persistence\_flag** | u(1) |
| **doi\_num\_display\_overlays\_minus2** | ue(v) |
| **doi\_target\_pic\_size\_present\_flag** | u(1) |
| if( doi\_target\_pic\_size\_present\_flag ) |  |
| **doi\_target\_pic\_width\_minus1** | u(16) |
| **doi\_target\_pic\_height\_minus1** | u(16) |
| } |  |
| **doi\_nuh\_layer\_id\_present\_flag** | u(1) |
| **doi\_pic\_partition\_flag** | u(1) |
| if( doi\_pic\_partition\_flag ) { |  |
| **doi\_partition\_type\_flag** | u(1) |
| **doi\_partition\_id\_len\_minus1** | u(4) |
| } |  |
| **doi\_offset\_params\_present\_flag** | u(1) |
| if( doi\_offset\_params\_present\_flag ) |  |
| **doi\_offset\_param\_length\_minus1** | u(4) |
| **doi\_resampling\_enabled\_flag** | u(1) |
| if( doi\_resampling\_enabled\_flag ) |  |
| **doi\_size\_param\_length\_minus1** | u(4) |
| for( i = 0; i < doi\_num\_display\_overlays\_minus2 + 2; i++ ) { |  |
| if ( doi\_nuh\_layer\_id\_present\_flag ) |  |
| **doi\_nuh\_layer\_id**[ i ] | u(6) |
| if ( doi\_pic\_partition\_flag ) |  |
| **doi\_partition\_id**[ i ] | u(v) |
| **doi\_alpha\_present\_flag**[ i ] | u(1) |
| if ( doi\_alpha\_present\_flag[ i ] ) { |  |
| if (doi\_nuh\_layer\_id\_present\_flag) |  |
| **doi\_alpha\_nuh\_layer\_id**[ i ] | u(6) |
| if (doi\_pic\_partition\_flag) |  |
| **doi\_alpha\_partition\_id**[ i ] | u(v) |
| } |  |
| if( i > 0 ) { |  |
| if (doi\_offset\_params\_present\_flag ) |  |
| **doi\_top\_left\_x**[ i ] | u(v) |
| **doi\_top\_left\_y**[ i ] | u(v) |
| } |  |
| if( doi\_resampling\_enabling\_flag ) { |  |
| **doi\_width\_minus1**[ i ] | u(v) |
| **doi\_height\_minus1**[ i ] | u(v) |
| } |  |
| } |  |
| } |  |
| } |  |
| } |  |

8.43.2 Display overlays SEI message semantics

The display overlays information (DOI) SEI message provides metadata to enable formation of a target display picture formed by overlaying multiple ordered display overlays in a specified order. A display overlay contains texture and optionally an alpha channel, each contained within a cropped decoded picture, a subpicture, and/or a constituent rectangle.

Use of this SEI message requires the definition of the following variables, where i is the layer identifier of a layer that may be present in the current CVS:

– An array of picture width and picture height in units of luma samples, denoted herein by PicWidthInLumaSamples[ i ] and PicHeightInLumaSamples[ i ], respectively.

– A chroma format indicator, denoted herein by ChromaFormatIdc[ i ], as described in subclause ‎7.3.

– An array of subpicture counts, denoted by NumSubpics[ i ].

– Arrays of the width and height of the subpictures, denoted herein by SubPicWidth[ i ][ j ] and SubPicHeight[ i ][ j ] respectively, where j is the subpicture index in 0 .. NumSubpics[ i ] – 1.

**doi\_id** specifies an identifier of the DOI SEI message.

**doi\_cancel\_flag** equal to 1 indicates that the SEI message cancels the persistence of any previous DOI SEI message with the same doi\_id in output order. doi\_cancel\_flag equal to 0 indicates that display overlays information follows.

**doi\_persistence\_flag** specifies the persistence of the DOI SEI message for the CVS.

doi\_persistence\_flag equal to 0 specifies that the DOI SEI message applies to the current AU only.

doi\_persistence\_idc equal to 1 specifies that the DOI SEI message applies to the current AU and persists for all subsequent AUs in output order until one or more of the following conditions are true:

– A new CVS begins.

– The bitstream ends.

– A picture in the current AU with a DOI SEI message with the same value of doi\_id is output that follows the current picture in output order.

**doi\_num\_display\_overlays\_minus2** plus 2 specifies the number of display overlays for which information is signalled in the SEI message. The value of doi\_num\_display\_overlays\_minus2 shall be in the range of 0 to 30, inclusive.

**doi\_target\_pic\_size\_present\_flag** equal to 1 specifies that the doi\_target\_pic\_width\_minus1 and doi\_target\_pic\_width\_minus1 syntax elements are present. doi\_target\_pic\_size\_present\_flagequal to 0 specifies that the doi\_target\_pic\_width\_minus1 and doi\_target\_pic\_width\_minus1 syntax elements are present.

**doi\_target\_pic\_width\_minus1** + 1, when present, indicates the width of the target picture.

**doi\_target\_pic\_height\_minus1**+ 1, when present, indicates the height of the target picture.

**doi\_nuh\_layer\_id\_present\_flag** equal to 1 specifies that the doi\_nuh\_layer\_id[ i ] syntax element is present in the SEI message. doi\_nuh\_layer\_id\_present\_flag equal to 0 specifies that the doi\_nuh\_layer\_id[ i ] syntax element is not present in the SEI message.

**doi\_pic\_partition\_flag** equal to 1 specifies that display overlay components are coded as constituent rectangles or subpictures. doi\_pic\_partition\_flag equal to 0 specifies that display overlay components are coded as pictures.

It is a requirement of bitstream conformance that at least one of doi\_nuh\_layer\_id\_present\_flag or doi\_pic\_partition\_flag shall be equal to 1.

**doi\_partition\_type\_flag** equal to 1 specifies that a display overlay component is coded as a constituent rectangle. doi\_partition\_type\_flag equal to 0 specifies that a display overlay component is coded as a subpicture.

When doi\_partition\_type\_flag equal to 1, it is a requirement of bitstream conformance that there is a constituent rectangles SEI message preceding the DOI SEI message in decoding order in the current PU.

**doi\_partition\_id\_len\_minus1** + 1 specifies the length of the doi\_partition\_id[ i ] syntax elements.

**doi\_offset\_params\_present\_flag**[ i ] equal to 1 specifies that offset parameters are present for the i-th display overlay.doi\_offset\_params\_present\_flag[ i ] equal to 0 specifies that offset parameters are not present for the i-th display overlay.

**doi\_offset\_param\_length\_minus1** plus 1 specifies the length of the doi\_top\_left\_x[ i ] and doi\_top\_left\_y[ i ], syntax elements in bits.

**doi\_resampling\_enabled\_flag** equal to 1 specifies that display overlay components may be resampled in the target display picture. doi\_resampling\_enabled\_flag equal to 0 specifies that display overlay components are not resampled in the target display picture.

**doi\_size\_param\_length\_minus1** plus 1 specifies the length of the doi\_width\_minus1[ i ] and doi\_height\_minus1[ i ] syntax elements in bits.

**doi\_nuh\_layer\_id**[ i ], when present, specifies the layer identifier of the texture component of the i-th display overlay. When not present, the value of doi\_nuh\_layer\_id[ i ] is inferred to be equal to the layer identifier of the PU containing the DOI SEI message.

If this SEI message is present in any layer in the current AU, it is a requirement of bitstream conformance that a DOI SEI message with the same value of doi\_id and the same payload is present in the layer with layer identifier doi\_nuh\_layer\_id[ 0 ].

**doi\_partition\_id**[ i ], when present and doi\_partition\_type\_flag equal to 1, specifies that the texture component of the i-th display overlay is represented by the j-th constituent rectangle when cr\_rect\_id[ j ] equal doi\_partition\_id[ i ]. doi\_partition\_id[ i ], when present and doi\_partition\_type\_flag equal to 0, specifies the subpicture index of the texture component of the i-th display overlay. When not present, the value of doi\_partition\_id[ i ] is inferred to be equal to 0.

When doi\_partition\_type\_flag equal to 1, doi\_partition\_id[ i ] shall be in the range of 0 .. cr\_num\_rects\_minus1[ doi\_nuh\_layer\_id[ i ] ] – 1. When doi\_partition\_type\_flag equal to 0, doi\_partition\_id[ i ] shall be in the range of 0 .. NumSubpics[ doi\_nuh\_layer\_id[ i ] ] – 1.

**doi\_alpha\_present\_flag**[ i ] equal to 1 specifies that an alpha component is provided for the i-th display overlay. doi\_alpha\_present\_flag[ i ] equal to 0 specifies that an alpha component is not provided for the i-th display overlay.

**doi\_alpha\_nuh\_layer\_id**[ i ], when present, specifies the layer identifier value of the alpha component of the i-th display overlay. When not present, the value of doi\_alpha\_nuh\_layer\_id[ i ] is inferred to be equal to the layer identifier of the PU containing the DOI SEI message.

**doi\_alpha\_partition\_id**[ i ], when present and doi\_partition\_type\_flag equal to 1, specifies the cr\_rect\_id[ j ] of the alpha component of the i-th display overlay. doi\_partition\_id[ i ], when present and doi\_partition\_type\_flag equal to 0, specifies the subpicture index of the alpha component of the i-th display overlay. When not present, the value of doi\_alpha\_partition\_id[ i ] is inferred to be equal to 0.

When doi\_partition\_type\_flag equal to 1, doi\_alpha\_partition\_id[ i ] shall be in the range of 0 .. cr\_num\_rects\_minus1[ doi\_nuh\_layer\_id[ i ] ] – 1. When doi\_partition\_type\_flag equal to 0, doi\_alpha\_partition\_id[ i ] shall be in the range of 0 .. NumSubpics[ doi\_nuh\_layer\_id[ i ] ] – 1.

**doi\_top\_left\_x**[ i ] and **doi\_top\_left\_y**[ i ] specify the horizontal and vertical positions, respectively, of the top left corner of the i-th display overlay in the target display picture, in luma samples. When not present, the values of doi\_top\_left\_x[ i ] and doi\_top\_left\_y[ i ] are inferred to be equal to 0. The length of the syntax elements is doi\_offset\_param\_length\_minus1 + 1 bits.

**doi\_width\_minus1**[ i ] plus 1 and **doi\_height\_minus1**[ i ] plus 1, when present, specify the width and height, respectively, in luma samples arrays of the i-th display overlay in the target display picture. The length of the syntax elements is doi\_size\_param\_length\_minus1 + 1 bits.

The variables CodedOverlayTexture[ i ] and CodedOverlayAlpha[ i ] are picture sample arrays with luma resolution CodedOverlayWidth[ i ] × CodedOverlayHeight[ i ], derived as follows:

– If doi\_pic\_partition\_flag is equal to 0, the following applies:

– CodedOverlayWidth[ i ] is set equal to PicWidthInLumaSamples[ doi\_nuh\_layer\_id[ i ] ] ].

– CodedOverlayHeight[ i ] is set equal to PicWidthInLumaSamples[ doi\_nuh\_layer\_id[ i ] ].

– If there is a picture in the AU for the layer with layer identifier doi\_nuh\_layer\_id[ i ], CodedOverlayTexture[ i ] is set equal to the cropped decoded picture from the layer with layer identifier doi\_nuh\_layer\_id[ i ] in the AU. Otherwise (there is no picture in the AU for the layer with layer identifier doi\_nuh\_layer\_id[ i ]), CodedOverlayTexture[ i ] is set equal to the previous cropped decoded picture in output order in the layer with layer identifier doi\_nuh\_layer\_id[ i ].

– Otherwise, the following applies:

– If doi\_alpha\_present\_flag[ i ] equal to 1, the following applies:

* If there is a picture in the AU for the layer with layer identifier doi\_alpha\_nuh\_layer\_id[ i ], CodedOverlayAlpha[ i ] is set equal to the cropped decoded picture from the layer with layer identifier doi\_alpha\_nuh\_layer\_id[ i ] in the AU.
* Otherwise (there is no picture in the AU for the layer with layer identifier doi\_nuh\_layer\_id[ i ]), CodedOverlayAlpha[ i ] is set equal to the previous cropped decoded picture in output order in the layer with layer identifier doi\_alpha\_nuh\_layer\_id[ i ].

– Otherwise, if doi\_partition\_type\_flag is equal to 0, the following applies:

– CodedOverlayTexture[ i ] is set equal to the subpicture with subpicture index doi\_partition\_id[ i ] from the layer with layer identifier doi\_nuh\_layer\_id[ i ].

– CodedOverlayWidth[ i ] is set equal to SubPicWidth[ doi\_nuh\_layer\_id[ i ] ][ doi\_partition\_id[ i ] ].

– CodedOverlayHeight[ i ] is set equal to SubPicHeight[ doi\_nuh\_layer\_id[ i ] ][ doi\_partition\_id[ i ] ].

– If doi\_alpha\_present\_flag[ i ] equal to 0, the following applies:

– CodedOverlayAlphaWidth[ i ] is set equal to SubPicWidth[ doi\_alpha\_nuh\_layer\_id[ i ] ][ doi\_partition\_id[ i ] ].

– CodedOverlayAlphaHeight[ i ] is set equal to SubPicHeight[ doi\_alpha\_nuh\_layer\_id[ i ] ][ doi\_partition\_id[ i ] ].

– CodedOverlayAlpha[ i ] is set equal to the subpicture with subpicture index doi\_alpha\_partition\_id[ i ] from the layer with layer identifier doi\_alpha\_nuh\_layer\_id[ i ].

– Otherwise (doi\_partition\_type\_flag is equal to 1), the following applies:

– CodedOverlayWidth[ i ] is set equal to CrRectWidth[ doi\_nuh\_layer\_id[ i ] ][ doi\_partition\_id[ i ] ].

– CodedOverlayHeight[ i ] is set equal to CrRectHeight[ doi\_nuh\_layer\_id[ i ] ][ doi\_partition\_id[ i ] ].

– CodedOverlayTexture[ i ] is set equal to the constituent rectangle with cr\_rect\_id[ j ] equal to doi\_partition\_id[ i ] from the layer with layer identifier doi\_nuh\_layer\_id[ i ].

– If doi\_alpha\_present\_flag[ i ] equal to 0

– CodedOverlayAlphaWidth[ i ] is set equal to CrWRectidth[ doi\_alpha\_nuh\_layer\_id[ i ] ][ doi\_partition\_id[ i ] ].

– CodedOverlayAlphaHeight[ i ] is set equal to CrRectHeight[ doi\_alpha\_nuh\_layer\_id[ i ] ][ doi\_partition\_id[ i ] ].

– CodedOverlayAlpha is set equal to the constituent rectangle with cr\_rect\_id[ j ] equal to doi\_alpha\_partition\_id[ i ] from the layer with layer identifier doi\_alpha\_nuh\_layer\_id[ i ].

The variables DisplayOverlayTexture[ i ] and DisplayOverlayAlpha[ i ] are sample arrays with luma resolution DisplayOverlayWidth[ i ] × DisplayOverlayHeight[ i ], derived as follows:

– If doi\_resampling\_enabled\_flag is equal to 1, the following applies:

– DisplayOverlayWidth[ i ] is set equal to doi\_width\_minus1[ i ] + 1.

– DisplayOverlayHeight[ i ] is set equal to doi\_height\_minus1[ i ] + 1.

– OverlayTexture[ i ] is derived by resampling CodedOverlayTexture[ i ] from a luma resolution of ( CodedOverlayWidth[ i ] × CodedOverlayHeight[ i ] ) to a luma resolution of ( DisplayOverlayWidth[ i ] × DisplayOverlayHeight[ i ] ).

– OverlayAlpha[ i ] is derived by resampling CodedOverlayAlpha from a luma resolution of ( CodedOverlayAlphaWidth[ i ] × CodedOverlayAlphaHeight[ i ] ) to a luma resolution of ( DisplayOverlayWidth[ i ] × DisplayOverlayHeight[ i ] )

– Otherwise (doi\_resampling\_enabled\_flag is equal to 0), the following applies:

– DisplayOverlayWidth[ i ] is set equal to CodedOverlayWidth[ i ].

– DisplayOverlayHeight[ i ] is set equal to CodedOverlayHeight[ i ].

– OverlayTexture[ i ] is set equal to CodedOverlayTexture[ i ].

– OverlayAlpha[ i ] is set equal to CodedOverlayAlpha[ i ].

The variables TargetPicWidth and TargetPicHeight, for the target display picture width and height, respectively, are derived as follows:

– If doi\_target\_pic\_size\_present\_flag)

– TargetPicWidth is set equal to doi\_target\_pic\_width\_minus1 + 1

– TargetPicHeight is set equal to doi\_target\_pic\_height\_minus1 + 1

– Otherwise

– TargetPicWidth is set equal to DisplayOverlayWidth[ 0 ].

– TargetPicHeight is set equal to DisplayOverlayHeight[ 0 ].

Let OverlayWithAlpha( tgtPic[ x ][ y ], ovlTex[ w ][ h ], ovlAlp[ w ][ h ] ) be specified as a function that returns the sample values derived by applying the alpha channel using tgtPic[ x ][ y ] as the background sample, ovlTex[ w ][ h ] as the foreground sample, and ovlAlp[ w ][ h ] as the alpha channel sample, using the process specified in subclause 8.23 if an alpha channel information SEI message is present in the CVS, or using a process determined via external means.

A target display picture should be formed as follows, with picture array, TargetPicture[ cIdx ][ x ][ y ] , with cIdx = 0..(ChromaFormatIdc = = 0 ) ? 0 : 2, x = 0..( cIdx = = 0 ) ? TargetPicWidth : TargetPicWidth / SubWidthC − 1, y = 0..( cIdx = = 0 ) ? TargetPicHeight : TargetPicHeight / SubHeightC − 1.

for( i = 0; i < doi\_num\_display\_overlays\_minus2 + 2; i++ ) {  
 for( h = 0, y = doi\_top\_left\_y[ i ]; y < DisplayOverlayHeight[ i ]; h++, y++ )   
 for( w = 0, x = doi\_top\_left\_x[ i ]; x < DisplayOverlayWidth[ i ]; w++, x++ )  
 if( !doi\_alpha\_present\_flag[ i ] )  
 TargetPicture[ 0 ][ x ][ y ] = OverlayTexture[ i ][ 0 ][ w ][ h ]   
 else  
 TargetPicture[ c ][ x ][ y ] = OverlayWithAlpha( TargetPicture[ 0 ][ x ][ y ],   
 OverlayTexture[ 0 ][ i ][ w ][ h ], OverlayAlpha[ i ][ w ][ h ] )

for( ( cIdx = 1; cIdx < ChromaFormatIdc = = 0 ) ? 1 : 3; cIdx++ ++ ) {  
 for( h = 0, y = doi\_top\_left\_y[ i ]/SubHeightC; y < DisplayOverlayHeight[ i ]/ SubHeightC; h++, y++ )   
 for( w = 0, x = doi\_top\_left\_x[ i ]/SubWidthC; x < DisplayOverlayWidth[ i ]/ SubWidthC; w++, x++ )  
 if( !doi\_alpha\_present\_flag[ i ] )  
 TargetPicture[ cIdx ][ x ][ y ] = OverlayTexture[ i ][ cIdx ][ w ][ h ]   
 else  
 TargetPicture[ cIdx ][ x ][ y ] = OverlayWithAlpha( TargetPicture[ cIdx ][ x ][ y ],   
 OverlayTexture[ cIdx ][ i ][ w ][ h ], OverlayAlpha[ i ][ w ][ h ] )

8.44 Bitdepth range information SEI message

8.44.1 Bitdepth range information SEI message syntax

|  |  |
| --- | --- |
| bitdepth\_range\_info( payloadSize ) { | **Descriptor** |
| **bri\_orig\_bit\_depth\_minus1** | u(4) |
| **bri\_num\_ranges\_minus1** | u(2) |
| if ( bri\_num\_ranges\_minus1 > 0 ) { |  |
| **bri\_nuh\_layer\_id\_present\_flag** | u(1) |
| **bri\_pic\_partition\_flag** | u(1) |
| if( bri\_pic\_partition\_flag ) { |  |
| **bri\_partition\_type\_flag** | u(1) |
| **bri\_partition\_id\_len\_minus1** | u(4) |
| } |  |
| } |  |
| for( i = 0; i <= bri\_num\_ranges\_minus1; i++ ) { |  |
| if ( bri\_nuh\_layer\_id\_present\_flag ) |  |
| **bri\_nuh\_layer\_id**[ i ] | u(6) |
| if ( bri\_pic\_partition\_flag ) |  |
| **bri\_partition\_id**[ i ] | u(v) |
| **bri\_num\_bits\_in\_range\_minus1**[ i ] | u(4) |
| **bri\_bit\_offset**[ i ] | u(4) |
| } |  |
| } |  |

8.44.2 Bitdepth range information SEI message semantics

The Bitdepth range information (BRI) SEI message enables reconstruction of a target picture of the same bit depth as an original source picture by inverting operations performed by the encoder to allocate bit ranges of the original source picture to one or more ranges coded in separate layer pictures, subpictures, or constituent rectangles.

Use of this SEI message requires the definition of the following variables, where i is the layer identifier of a layer that may be present in the current CVS:

– A chroma format indicator, denoted herein by ChromaFormatIdc[ i ], as described in clause 7.3.

– A bit depth for the samples, denoted herein by BitDepth[ i ].

– An array of picture width and picture height in units of luma samples, denoted herein by PicWidthInLumaSamples[ i ] and PicHeightInLumaSamples[ i ], respectively.

– An array of subpicture counts, denoted by NumSubpics[ i ].

– Arrays of the width and height of the subpictures, denoted herein by SubPicWidth[ i ][ j ] and SubPicHeight[ i ][ j ] respectively, where j is the subpicture index in 0 .. NumSubpics[ i ] – 1.

**bri\_orig\_bit\_depth\_minus1** plus 1 specifies the bit depth of the original picture.

The variable OrigBitDepth is set equal to bri\_orig\_bit\_depth\_minus1 + 1.

**bri\_num\_ranges\_minus1** + 1 specifies the number of ranges used for allocating bits of the original picture.

The variable briNumRanges is set equal to bri\_num\_ranges\_minus1 + 1.

**bri\_nuh\_layer\_id\_present\_flag** equal to 1 specifies that the bri\_nuh\_layer\_id[ i ] syntax element is present in the SEI message. bri\_nuh\_layer\_id\_present\_flag equal to 0 specifies that the bri\_nuh\_layer\_id[ i ] syntax element is not present in the SEI message.

**bri\_pic\_partition\_flag** equal to 1 specifies that the bri\_partition\_id[ i ] syntax element is present. bri\_pic\_partition\_flag equal to 0 specifies that the bri\_partition\_id[ i ] syntax element is not present.

**bri\_partition\_type\_flag** equal to 1 specifies that each range is coded as a constituent rectangle. bri\_partition\_type\_flag equal to 0 specifies that each range is coded as a subpicture.

When bri\_partition\_type\_flag equal to 1, it is a requirement of bitstream conformance that there is a constituent rectangles SEI message preceding the BRI SEI message in decoding order in the current PU.

**bri\_partition\_id\_len\_minus1** + 1 specifies the length of the bri\_rect\_id[ i ] and bri\_subpic\_id[ i ] syntax elements.

**bri\_nuh\_layer\_id**[ i ], when present, specifies the layer identifier of the texture component of the i-th display overlay. When not present, the value of bri\_nuh\_layer\_id[ i ] is inferred to be equal to the layer identifier of the PU containing the BDR SEI message.

If this SEI message is present in any layer in the current AU, it is a requirement of bitstream conformance that a BRI SEI message with the same payload is present in the layer with layer identifier bri\_nuh\_layer\_id[ 0 ].

**bri\_partition\_id**[ i ], when present and bri\_partition\_type\_flag equal to 1, specifies the cr\_rect\_id[ j ] of the texture component of the i-th display overlay. bri\_partition\_id[ i ], when present and bri\_partition\_type\_flag equal to 0, specifies the subpicture index of the texture component of the i-th display overlay. When not present, the value of bri\_partition\_id[ i ] is inferred to be equal to 0.

When bri\_partition\_type\_flag equal to 1, bri\_partition\_id[ i ] shall be in the range of 0 .. cr\_num\_rects\_minus1[ bri\_nuh\_layer\_id[ i ] ] – 1. When bri\_partition\_type\_flag equal to 0, bri\_partition\_id[ i ] shall be in the range of 0 to NumSubpics[ bri\_nuh\_layer\_id[ i ] ] – 1, inclusive.

**bri\_num\_bits\_in\_range\_minus1**[ i ] + 1 specifies the number of bits in the i-th range. bri\_num\_bits\_in\_range\_minus1[ i ] shall be in the range of 0 to BitDepth[ i ] – 1, inclusive.

The variable NumBits[ i ] is set equal to bri\_num\_bits\_in\_range\_minus1[ i ] + 1.

The sum of bri\_num\_bits\_in\_range\_minus1[ i ] for i in 0 to bri\_num\_ranges\_minus1 shall be less than or equal to OrigBitDepth.

**bri\_bit\_offset**[ i ] specifies a bit shifting parameter used to derive the target picture. bri\_bit\_offset[ i ] shall be in the range of 0 to BitDepth[ i ] – NumBits[ i ].

The variables CodedRangeRegion[ i ] CodedWidth[ i ], and CodedHeight[ i ] are derived as follows:

– If bri\_pic\_partition\_flag is equal to 0, the following applies:

– CodedWidth[ i ] is set equal to PicWidthInLumaSamples[ bri\_nuh\_layer\_id[ i ] ] ].

– CodedHeight[ i ] is set equal to PicWidthInLumaSamples[ bri\_nuh\_layer\_id[ i ] ].

– CodedRangeRegion[ i ] is set equal to the cropped decoded picture from the layer with layer identifier bri\_nuh\_layer\_id[ i ] in the AU

– Otherwise, if bri\_partition\_type\_flag is equal to 0, the following applies:

– CodedWidth[ i ] is set equal to SubPicWidth[ bri\_nuh\_layer\_id[ i ] ][ bri\_partition\_id[ i ] ].

– CodedHeight[ i ] is set equal to SubPicHeight[ bri\_nuh\_layer\_id[ i ] ][ bri\_partition\_id[ i ] ].

– CodedRangeRegion[ i ] is set equal to the subpicture with subpicture index bri\_partition\_id[ i ] from the layer with layer identifier bri\_nuh\_layer\_id[ i ].

– Otherwise (bri\_partition\_type\_flag is equal to 1), the following applies:

– CodedWidth[ i ] is set equal to CrRectWidth[ bri\_nuh\_layer\_id[ i ] ][ bri\_rect\_id[ i ] ].

– CodedHeight[ i ] is set equal to CrRectHeight[ bri\_nuh\_layer\_id[ i ] ][ bri\_rect\_id[ i ] ].

– CodedRangeRegion[ i ] is set equal to the constituent rectangle with cr\_rect\_id[ j ] equal to bri\_partition\_id[ i ] from the layer with layer identifier bri\_nuh\_layer\_id[ i ].

It is a requirement of bitstream conformance that CodedWidth[ i ] equal CodedWidth[ j ] and CodedHeight[ i ] equal CodedHeight[ i ] for all i and j in 0 .. bri\_num\_ranges – 1.

The samples SampleToCode[ i ] of CodedRangeRegion[ i ], for i in 0 .. briNumRanges, are indicated to have been pre-processed prior to encoding from an original source picture of resolution CodedWidth[ 0 ] x CodedHeight[ 0 ] with chroma format ChromaFormatIdc and bit depth OrigBitDepth, according to the process of subclause 8.X.2.1.

Each sample, TargetSample, of a target picture of resolution CodedWidth[ 0 ] x CodedHeight[ 0 ] with chroma format ChromaFormatIdc and bit depth OrigBitDepth, may be reconstructed using the Target sample reconstruction process described in subclause 8.X.2.2, for each sample, DecodedSample[ i ] of CodedRangeRegion[ i ] for i in 0 .. briNumRanges.

**8.X.2.1 Sample pre-process prior to encoding**

The sample to code for the i-th range SampleToCode[ i ] for i in 0 .. briNumRanges is derived as follows, with InputSample as the input sample:

currOrigBitPos = 0

for (i = 0; i < briNumRanges; i++) {

mask = ( 1 << NumBits[ i ] ) – 1

k = OrigBitDepth – currOrigBitPos – NumBits[ i ]

shiftedInput =  InputSample >> k

SampleToCode[ i ] = ( shiftedInput & mask ) << bri\_bit\_offset[ i ]

currOrigBitPos += NumBits[ i ]

}

**8.X.2.2 Target sample reconstruction process**

The target sample TargetSample is derived as follows, with DecodedSample[ i ] as the decoded sample of the i-th range:

currTargetBitPos = 0

TargetSample = 0

for (i = 0; i < briNumRanges; i++) {

k = OrigBitDepth – currTargetBitPos – NumBits[ i ]

maxVal = (1 << ( NumBits[ i ] + bri\_bit\_offset[ i ] ) ) − 1

if (bri\_bit\_offset[ i ] > 0)

decRoundingOffset = 1<< ( bri\_bit\_offset[ i ] − 1 )

else

decRoundingOffset = 0

adjustedDecodedSample = Clip3( 0, maxVal, DecodedSample[ i ] + decRoundingOffset)

shiftedDecodedSample = adjustedDecodedSample >> bri\_bit\_offset[ i ]

TargetSample += shiftedDecodedSample << k

currTargetBitPos += NumBits[ i ]

}

8.44 Digitally signed content initialization SEI message

8.44.1 Digitally signed content initialization SEI message syntax

|  |  |
| --- | --- |
| trustworthy\_content\_initialization( payloadSize ) { | **Descriptor** |
| **dsci\_hash\_method\_type** | u(8) |
| **dsci\_key\_source\_uri** | st(v) |
| **dsci\_num\_verification\_substreams\_minus1** | ue(v) |
| **dsci\_key\_retrieval\_mode\_idc** | ue(v) |
| if( dsci\_key\_retrieval\_mode\_idc = = 1){ |  |
| **dsci\_use\_key\_register\_idx\_flag** | u(1) |
| if( dsci\_use\_key\_register\_idx\_flag ) |  |
| **dsci\_key\_register\_idx** | ue(v) |
| } |  |
| **dsci\_content\_uuid\_present\_flag** | u(1) |
| if( dsci\_content\_uuid\_present\_flag) |  |
| **dsci\_content\_uuid** | u(128) |
| } |  |

8.44.1 Digitally signed content initialization SEI message semantics

The digitally signed content initialization SEI message, digitally signed content selection SEI message, and digitally signed content verification SEI message provide a mechanism for verifying that the coded video has been produced by a content provider that identifies itself via the digital certificate that is referenced in the digitally signed content initialization SEI message. This SEI message also provides information about the secure hash algorithm used for calculating message digests, which are used together with the digital signature present in digitally signed content verification SEI messages to verify the trustworthiness of the SPS, PPS, APS, PH and VCL NAL units present in the coded video sequence. It further provides information about the digital signature algorithm used and the public key of the content provider. The digitally signed content initialization SEI message may provide the public key of the content provider either by providing an URI that identifies a C2PA Manifest Store that contains the certificate of the content provider or by providing an URI that directly identifies the certificate.

When a digitally signed content initialization SEI message is present in any AU of a CVS, a digitally signed content initialization SEI message shall be present for all IDR, CRA and GDR PUs of the CVS.

When a digitally signed content initialization SEI message is present in any AU of a CVS, it shall precede all SPS, PPS, APS, PH and VCL NAL units of the AU.

The digitally signed content initialization SEI message applies to the current coded picture and all following coded pictures until one or more of the following conditions are true:

* The bitstream ends.
* A new CVS begins.
* A new digitally signed content initialization SEI message is received.

When a digitally signed content initialization SEI message is present in an AU of a CVS, a digitally signed content verification SEI message shall be present for each of the substreams that a NAL unit is assigned to. The signed content verification SEI message shall be present in the bitstream before one or more of the following conditions are true:

* The bitstream ends.
* A new CVS begins.
* A new digitally signed content initialization SEI message is received.

**dsci\_hash\_method\_type** indicates the secure hash algorithm that is used to calculate message digests for subsets of SPS, PPS, APS, PH and VCL NAL units of the coded video sequence. Based on these message digests and the digital signatures present in digitally signed content verification SEI messages, a decoder can verify that the coded video was produced by the content originator indicated by the syntax elements dsci\_key\_source\_uri, dsci\_use\_key\_register\_idx\_flag and, if dsci\_use\_key\_register\_idx\_flag flag is equal to 1, dsci\_key\_register\_idx. The supported values for the syntax element dsci\_hash\_method\_type, the block size used for calculating the message digest, and the size of the calculated message digests are specified in. Values of dsci\_hash\_method\_type that are not listed in the Table 6 are reserved for future use by ITU-T | ISO/IEC and shall not be present in payload data conforming to this version of this Specification. Decoders shall ignore trustworthy initialization SEI messages that contain reserved values for dsci\_hash\_method\_type. The secure hash algorithms listed in Table 6 are specified in the “Secure Hash Standard” FIPS PUB 180-4.

**Table 6 – Supported values of dsci\_hash\_method\_type**

|  |  |  |  |
| --- | --- | --- | --- |
| **dsci\_hash\_method\_type** | **Hash method** | **Block size (bits)** | **Message digest size (bits)** |
| 0 | SHA-1 | 512 | 160 |
| 1 | SHA-224 | 512 | 224 |
| 2 | SHA-256 | 512 | 256 |
| 3 | SHA-384 | 1024 | 384 |
| 4 | SHA-512 | 1024 | 512 |
| 5 | SHA-512/224 | 1024 | 224 |
| 6 | SHA-512/256 | 1024 | 256 |

**dsci\_key\_source\_uri** contains a URI with syntax and semantics as specified in IETF Internet Standard 66. If dsci\_key\_retrieval\_mode\_idc is equal to 0, dsci\_key\_source\_uri specifies a C2PA Manifest Store as specified in C2PA Technical Specification. If dsci\_key\_retrieval\_mode\_idc is equal to 1, the following applies:

* If dsci\_use\_key\_register\_idx\_flag is equal to 0, the URI identifies the certificate of the content provider that can be used for verifying the signatures present in following digitally signed content verification SEI messages;
* Otherwise (if dsci\_use\_key\_register\_idx\_flag is equal to 1), the URI identifies a register of certificates and the certificate of the content provider that can be used for verifying the signatures present in following digitally signed content verification SEI messages as indicated by dsci\_key\_register\_idx.

**dsci\_num\_verification\_substreams\_minus1** plus 1 indicates the number of substreams for which message digests are calculated and signatures may be present in following digitally signed content verification SEI messages.

The variable NumVerificationSubstream is derived as:

NumVerificationSubstream = dsci\_num\_verification\_substreams\_minus1 + 1.

**dsci\_key\_retrieval\_mode\_idc** equal to 0 indicates that the URI contained in dsci\_key\_source\_uri specifies a C2PA Manifest Store as specified in C2PA Technical Specification. dsci\_key\_retrieval\_mode\_idc equal to 1 indicates that the URI contained in dsci\_key\_source\_uri and, when present, dsci\_key\_register\_idx specify a certificate. In this version of this Specification dsci\_key\_retrieval\_mode\_idc shall be in the range of 0 to 1. Decoders shall also allow other values of dsci\_key\_retrieval\_mode\_idc, but shall ignore the content of the digitally signed content initialization SEI message, associated digitally signed content selection SEI messages and associated digitally signed content verification SEI messages.

**dsci\_use\_key\_register\_idx\_flag** equal to 1 indicates that the URI contained in dsci\_key\_source\_uri specifies a register of certificates and the syntax element dsci\_key\_register\_idx is present in the SEI message. dsci\_use\_key\_register\_idx\_flag equal to 0 indicates that the URI contained in dsci\_key\_source\_uri specifies a certificate and the syntax element dsci\_key\_register\_idx is not present in the SEI message.

When dsci\_key\_retrieval\_mode\_idc is equal to 0, the media asset for which Active Manifest, as specified in C2PA Technical Specification, provides content binding is the digitally signed content initialization SEI message. The following constraints apply to the C2PA Manifest Store identified by the dsci\_key\_source\_uri:

* The Active Manifest shall contain exactly one c2pa.hash.data, as specified in C2PA Technical Specification, hard binding to content assertion.
* The exclusion range indicated in the c2pa.hash.data shall match the dsci\_key\_source\_uri bytes in the digitally signed content initialization SEI message.

**dsci\_key\_register\_idx**, when present, contains an index that specifies the certificate of the content provider, in the certificate register indicated by dsci\_key\_source\_uri, which can be used for verifying the signatures present in following digitally signed content verification SEI messages.

The certificate indicated by the syntax elements dsci\_key\_retrieval\_mode\_idc, dsci\_use\_key\_register\_idx\_flag, dsci\_key\_source\_uri, and, if dsci\_use\_key\_register\_idx\_flag is equal to 1, dsci\_key\_register\_idx shall specify a digital signature method, with associated parameters (if applicable), and the public key of the content provider. When dsci\_key\_retrieval\_mode\_idc is equal to 1, the format in which this information is provided is outside the scope of this specification. It is suggested that a digital signature algorithm conforming to the “Digital Signature Standard” FIPS 186-5 is used.

**dsci\_content\_uuid\_present\_flag e**qual to 1 specifies that the syntax element dsci\_content\_uuid is present. dsci\_content\_uuid\_present\_flag equal to 0 specifies that the syntax element dsci\_content\_uuid is not present. When dsci\_key\_retrieval\_mode\_idc is equal to 0, dsci\_content\_uuid\_present\_flag shall be equal to 1.

**dsci\_content\_uuid**, when present, indicates an identifier for the video content and shall have a value specified as a UUID according to the procedures of ISO/IEC 11578:1996, Annex A.

When a digitally signed content initialization SEI message is present in an AU, the calculation of NumVerificationSubstream message digests is initialized according to the specification in FIPS PUB 180-4 for the specified dsci\_hash\_method\_type. Each SPS, PPS, APS, PH and VCL NAL unit following the digitally signed content initialization SEI message is associated to one of the NumVerificationSubstream message digests; the verification substream id is either indicated by the digitally signed content selection SEI message or, if no digitally signed content selection SEI message is present for a PU, inferred to be equal to 0. The message used for calculating the k-th message digest, with k being in the range from 0 to dsci\_num\_verification\_substreams\_minus1, inclusive, is obtained by concatenating all SPS, PPS, APS and VCL NAL units associated with the k-th verification substream. The calculation of the message digests is conducted based on blocks, where the block size is specified in Table 6 depending on the value of dsci\_hash\_method\_type. For each SPS, PPS, APS, PH and VCL NAL unit, the associated message digest is updated according to the algorithm specified in FIPS PUB 180-4 for the specified dsci\_hash\_method\_type. Note that, since the message digests are calculated for the concatenation of all SPS, PPS, APS, PH and VCL NAL units for a verification substream, some of the processing blocks typically span over two or more successive NAL units.

8.44 Digitally signed content selection SEI message

8.44.1 Digitally signed content selection SEI message syntax

|  |  |
| --- | --- |
| trustworthy\_content\_selection( payloadSize ) { | **Descriptor** |
| **dscs\_verification\_substream\_id** | u(8) |
| } |  |

8.44.1 Digitally signed content selection SEI message semantics

The digitally signed content selection SEI message provides a mechanism for associating coded pictures with one of the verification substreams indicated in a digitally signed content initialization SEI message.

When an AU contains both a digitally signed content initialization SEI message and a digitally signed content selection SEI message, the digitally signed content initialization SEI message shall precede the digitally signed content selection SEI message in decoding order.

When a CVS does not contain a digitally signed content initialization SEI message, CLVSs of the CVS shall not contain a digitally signed content selection SEI message.

When a digitally signed content selection SEI message is present in any AU of a CVS, it shall precede all SPS, PPS, APS, PH and VCL NAL units of the AU.

**dscs\_verification\_substream\_id** indicates the verification substream to which the SPS, PPS, APS, PH and VCL NAL units of the current coded picture are assigned to. When a digitally signed content initialization SEI message was present in the current coded video sequence, but no digitally signed content selection SEI message is present for a coded picture, the value of dscs\_verification\_substream\_id is inferred to be equal to 0. The value of dscs\_verification\_substream\_id shall be in the range from 0 to dsci\_num\_verification\_substreams\_minus1, inclusive.

As specified in section 3.1.2, the message digest for the verification substream with id equal to dscs\_verification\_substream\_id is updated with the SPS, PPS, APS, PH and VCL NAL units of the current coded picture according to the dsci\_hash\_method\_type specified in the preceding digitally signed content initialization SEI message.

8.44 Digitally signed content verification SEI message

8.44.1 Digitally signed content verification SEI message syntax

|  |  |
| --- | --- |
| trustworthy\_content\_verification( payloadSize ) { | **Descriptor** |
| **dscv\_verification\_substream\_id** | u(8) |
| **dscv\_signature\_length\_in\_octets\_minus1** | u(16) |
| **dscv\_signature** | u(v) |
| } |  |

8.44.1 Digitally signed content verification SEI message semantics

The digitally signed content verification SEI message provides a mechanism for verifying the digital signature of video content.

When a CVS does not contain a digitally signed content initialization SEI message, CLVSs of the CVS shall not contain digitally signed content verification SEI message.

When an AU contains both a digitally signed content initialization SEI message and a digitally signed content verification SEI message, the digitally signed content initialization SEI message shall precede the digitally signed content verification SEI message. When a PU contains both a digitally signed content selection SEI message and a digitally signed content verification SEI message, the digitally signed content selection SEI message shall precede the digitally signed content verification SEI message.

When a digitally signed content verification SEI message is present in a PU of a CVS, no SPS, PPS, APS, PH or VCL NAL unit shall be assigned to the substream that is indicated by dscv\_verification\_substream\_id, unless one or more of the following conditions are true:

* The bitstream ends.
* A new CVS begins.
* A new digitally signed content initialization SEI message is received.

**dscv\_verification\_substream\_id** indicates the verification substream to which the SEI message applies.

**dscv\_signature\_length\_in\_octets\_minus1** plus 1 specifies the length of the syntax element dscv\_signature in octets (one octet consists of 8 bits).

**dscv\_signature** contains the digital signature for the verification substream indicated by dscv\_verification\_substream\_id.

The verification of the bitstream signature consists of the following ordered steps:

1. The calculation of the message digest referred to as CurrDigest is finalized as follows:
   * The concatenation of the SPS, PPS, APS, PH and VCL NAL units for the verification substream with id equal to dscv\_verification\_substream\_id is padded according to the specification in FIPS PUB 180-4. Note that it is sufficient to pad the last NAL unit of the verification substream.
   * The calculation of the message digest CurrDigest is finalized according to the specification in FIPS PUB 180-4. The length (in bits) of the message digest is given in Table 6.
2. The reference message digest RefDigest is determined as follows:
   * If dscv\_verification\_substream\_id is greater than 0, the reference message digest RefDigest is the last calculated message digest for the verification substream with id equal to dscv\_verification\_substream\_id – 1. It is a requirement of bitstream conformance that any digitally signed content verification SEI associated with verification substream id equal to dscv\_verification\_substream\_id – 1 is present before the digitally signed content verification SEI message with verification substream id equal to dscv\_verification\_substream\_id.
   * Otherwise, if the current digitally signed content verification SEI message is the first digitally signed content verification SEI with verification id equal to 0 in the coded video sequence and the preceding coded video sequence did not contain any digitally signed content initialization SEI message (this includes the case that the current coded video sequence is the first coded video sequence in the bitstream), the RefDigest is set equal to a bitstring that consists of DigestSize bits equal to 1, where DigestSize is the size of the message digest as specified in Table 6.
   * Otherwise, the reference message digest RefDigest is the last calculated message digest for the verification substream with id equal to 0.
3. The identification string IdString is constructed by concatenating the binary representations of the reference message digest RefDigest, the current message digest, and the dsci\_hash\_method\_type and, when present, the dsci\_content\_uuid, as illustrated in Figure 1.

|  |  |  |  |
| --- | --- | --- | --- |
| RefDigest | CurrDigest | dsci\_hash\_method\_type | dsci\_content\_uuid (when present) |

Figure 1 – Construction of identification string IdString

The number of bits for RefDigest is determined by the value of dsci\_hash\_method\_type which was valid when calculating the value of RefDigest, the number of bits for CurrDigest is determined by the current value of dsci\_hash\_method\_type, and the value of dsci\_hash\_method\_type is represented with 8 bits and, when present, the value of dsci\_content\_uuid is represented with 128 bits.

1. The identification string IdString represents the message used for verifying the signature. The signature verification algorithm and the public key used for verifying the signature are indicated by the syntax elements dsci\_use\_key\_register\_idx\_flag, dsci\_key\_source\_uri, and, if dsci\_use\_key\_register\_idx\_flag is equal to 1, dsci\_key\_register\_idx.

NOTE 1 – Since the bitstring used for signature verification includes the RefDigest, it cannot only be verified that the SPS, PPS, APS, PH and VCL NAL units used for calculated the current message digest are correct, but it can additionally be verified that neither additional SPS, PPS, APS, PH and VCL NAL units were added to the bitstream nor SPS, PPS, APS, PH and VCL NAL units were removed from the bitstream.

NOTE 2 – When a decoder tunes into a bitstream, the IdString constructed for the first digitally signed content verification SEI message cannot be verified, because the value of RefDigest cannot be calculated correctly. But starting from the second digitally signed content verification SEI message, the signatures can be verified.

After verification, the message digest for the verification substream with id equal to dscv\_verification\_substream\_id is reinitialized according to the specification in FIPS PUB 180-4 for the specified dsci\_hash\_method\_type.

8.45 AI usage restrictions SEI message

8.45.1 AI usage restrictions SEI message syntax

|  |  |
| --- | --- |
| ai\_usage\_restrictions ( payloadSize ) { | Descriptor |
| **aur\_cancel\_flag** | u(1) |
| if( !aur\_cancel\_flag ) { |  |
| **aur\_persistence\_flag** | u(1) |
| **aur\_num\_restrictions\_minus1** | ue(v) |
| for( i = 0; i <= aur\_num\_restrictions\_minus1; i++ ) { |  |
| **aur\_restriction**[ i ] | ue(v) |
| **aur\_context\_present\_flag**[ i ] | u(1) |
| if( aur\_context\_present\_flag[ i ] ) |  |
| **aur\_context**[ i ] | ue(v) |
| } |  |
| } |  |

8.45.2 AI usage restrictions SEI message semantics

The artificial intelligence (AI) usage restrictions SEI message signals restriction and optional context information for usage by AI applications.

**aur\_cancel\_flag** equal to 1 specifies that the SEI message cancels the persistence of any previous AI usage restrictions SEI message in output order. aur\_cancel\_flag equal to 0 specifies that AI usage restriction information follows.

**aur\_persistence\_flag** specifies the persistence of the AI usage restrictions SEI message for the current layer.

aur\_persistence\_flag equal to 0 specifies that the AI usage restrictions SEI message applies to the current decoded picture only.

aur\_persistence\_flag equal to 1 specifies that the AI usage restrictions SEI message applies to the current decoded picture and persists for all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture in the current layer in an AU associated with an AI usage restrictions SEI message is output that follows the current picture in output order.

**aur\_num\_restrictions\_minus1** plus one specifies the number of restriction entries that follow.

**aur\_restriction**[ i ] specifies the i-th restrictionas specified in Table xx. The value of aur\_restriction[ i ] shall be in the range of 0 to 2, inclusive. Values greater than 2 for aur\_restriction[ i ] are reserved for future use by ITU-T | ISO/IEC and shall be ignored in bitstreams conforming to this version of this Specification.

**Table xx – Definition of aur\_restriction [ i ]**

|  |  |
| --- | --- |
| **Value** | **Interpretation** |
| 0 | Do not use for AI training |
| 1 | Do not use for generative (modification or creation) AI |
| 2 | Do not use in any AI application |

**aur\_context\_present\_flag** [ i ]indicates the presence of context information for the value in aur\_restriction[ i ].

**aur\_context**[ i ], when present, indicates the context for the i-th aur\_restriction[ i ] as specified in Table xy, where (aur\_context & bitMask ) not equal to zero indicates that the i-th restriction applies to the context associated with the bitMask value in Table xy. When aur\_context[ i ] is greater than 0 and ( aur\_context[ i ] & bitMask ) is equal to zero, the application of the i-th restriction is undefined in the context associated with the bitMask value in Table xy. The value of aur\_context[ i ] shall be in the range of 1 to 15, inclusive, in bitstream conforming to this version of this Specification. Values greater than 15 are reserved for future use by ITU-T | ISO/IEC and shall be ignored in bitstreams conforming to this version of this Specification. When the value of aur\_context[ i ] is greater than 15 decoders conforming to this version of this Specification shall ignore aur\_context[ i ]. aur\_context[ i ] shall not be equal to 0.

**Table xy – Definition of aur\_context**[ i ]

|  |  |
| --- | --- |
| **Bitmask** | **Interpretation** |
| 0x0001 | commercial use |
| 0x0002 | non-commercial use |
| 0x0004 | official government use |
| 0x0008 | research and academic use |

*Changes to VVC subclause D.12 for adding the support of packed regions, constitiuent rectangles, and quality metrics SEI messages*

**D.12.11 Use of the packed regions information SEI message in VVC**

For purposes of interpretation of the packed regions information SEI message, the following variables are specified:

– PicWidthInLumaSamples is set equal to the value of pps\_pic\_width\_in\_luma\_samples − ‌SubWidthC \* ‌ ( pps\_conf\_win\_left\_offset + pps\_conf\_win\_right\_offset ).

– PicHeightInLumaSamples is set equal to the value of pps\_pic\_height\_in\_luma\_samples − ‌SubHeightC \* ‌ ( pps\_conf\_win\_top\_offset + pps\_conf\_win\_bottom\_offset ).

– MaxPicWidth is set equal to the value of sps\_pic\_width\_in\_luma\_samples − ‌SubWidthC \* ‌ ( sps\_conf\_win\_left\_offset + sps\_conf\_win\_right\_offset ).

– MaxPicHeight is set equal to the value of sps\_pic\_height\_in\_luma\_samples − ‌SubHeightC \* ‌ ( sps\_conf\_win\_top\_offset + sps\_conf\_win\_bottom\_offset ).

– ChromaFormatIdc is set equal to sps\_chroma\_format\_idc.

– BitDepthY and BitDepthC are both set equal to BitDepth.

**D.12.13 Use of the constituent rectangles SEI message in VVC**

For purposes of interpretation of the constituent rectangles SEI message, the following variables are specified:

– PicWidthInLumaSamples and PicHeightInLumaSamples are set equal to pps\_pic\_width\_in\_luma\_samples and pps\_pic\_height\_in\_luma\_samples, respectively.

– MaxPicWidth and MaxPicHeight are set equal to sps\_pic\_width\_max\_in\_luma\_samples and sps\_pic\_height\_max\_in\_luma\_samples, respectively.

– ChromaFormatIdc is set equal to sps\_chroma\_format\_idc.

– BitDepthY and BitDepthC are both set equal to BitDepth.

– NumSubpics is set equal to sps\_num\_subpics\_minus1 + 1

– SubPicTopLeftX[ i ] is set equal to sps\_subpic\_ctu\_top\_left\_x[ i ] \* CtbSizeY

– SubPicTopLeftY[ i ] is set equal to sps\_subpic\_ctu\_top\_left\_y[ i ] \* CtbSizeY

– SubPicWidth[ i ] is set equal to (sps\_subpic\_width\_minus1[ i ] + 1 ) \* CtbSizeY − 1

– SubPicHeight[ i ] is set equal to (sps\_subpic\_height\_minus1[ i ] + 1 ) \* CtbSizeY – 1

D.12.14 Use of the quality metric SEI message in VVC

For purposes of interpretation of the quality metric SEI message, the derivation of the variables ChromaFormatIdc, NumPics, TestPicList, PicWidth, PicHeight, and GainRefPicList is specified below.

Let TestPicList initially consist of the cropped decoded pictures of the current CLVS in output order.

When the quality metric SEI message is included as a type of an SEI message in an SEI processing order SEI message, any quality metric SEI message that is associated with the SEI processing order SEI message shall be included in a processing order nesting SEI message.

When the quality metric SEI message is included as the i-th type of an SEI message in an SEI processing order SEI message, the following applies:

– It is a requirement of bitstream conformance that an SEI message seiB that implies post-processing to be performed shall be present as the j-th type of an SEI message in the same SEI processing order SEI message and po\_sei\_processing\_order[ j ] shall be equal to po\_sei\_processing\_order[ i ].

– The quality metric SEI message indicates the picture quality resulting from the post-processing implied by seiB.

– TestPicList is updated as follows for each post-processing stage with po\_sei\_processing\_order[ j ] less than or equal to po\_sei\_processing\_order[ i ] in a non-decreasing order of j.

– When the post-processing stage results into a picture picA with an output time that is equal to the output time of a picture picB in TestPicList, picB in TestPicList is replaced by picA.

– When the post-processing stage results into a picture picA with an output time that is not equal to the output time of any picture in TestPicList, picA is inserted in TestPicList in a manner that pictures in TestPicList remain in output order.

NumPics is set equal to the count of pictures in TestPicList.

PicWidth[ i ] and PicHeight[ i ] are set equal to the width and height of TestPicList[ i ], respectively, in luma samples.

When a quality metric SEI message is included as the k-th SEI message in a processing order nesting SEI message and qm\_gain\_flag[ i ] is equal to 1 for any value of i, the following applies:

– If qm\_gain\_reference\_flag[ i ] is equal to 0, the i-th metric value in the quality metric SEI message represents a gain of the post-processing stage with po\_sei\_processing\_order[ j ] equal to pon\_processing\_order[ k ] relative to the picture or pictures used as input to that post-processing stage and GainRefPicList is set equal to TestPicList derived for the processing stages up to but not including the processing stage with po\_sei\_processing\_order[ j ].

– Otherwise (qm\_gain\_reference\_flag[ i ] is equal to 1), the i-th metric value in the quality metric SEI message represents a cumulative gain of all the post-processing stages with po\_sei\_processing\_order[ j ] less than or equal to pon\_processing\_order[ k ] relative to the cropped decoded picture or pictures and GainRefPicList consists of the cropped decoded pictures of the current CLVS in output order.

– It is a requirement of bitstream conformance that the count of pictures in GainRefPicList shall be equal to NumPics and the width and height of GainRefPicList[ i ] in luma samples shall be equal to PicWidth[ i ] and PicHeight[ i ], respectively.

The value of ChromaFormatIdc is derived as follows:

– If the quality metric SEI message is not included in a processing order nesting SEI message, ChromaFormatIdc is set equal to sps\_chroma\_format\_idc.

– Otherwise (the quality metric SEI message is included in a processing order nesting SEI message), ChromaFormatIdc is set to a value that matches the chroma format of the pictures in TestPicList and it is a requirement of bitstream conformance that the chroma format of all the pictures in TestPicList and GainRefPicList, when present, shall be identical.

When a quality metric SEI message qmSeiA is present in a picture unit that is not the first picture unit of a CLVS in decoding order and at least one value of qm\_clvs\_metric\_value[ i ][ c ] is present, the following applies:

– If qmSeiA is not included in a processing order nesting SEI message, each value of qm\_clvs\_metric\_value[ i ][ c ] shall be equal to the value of qm\_clvs\_metric\_value[ i ][ c ] in the quality metric SEI message that is present in the first picture unit of the CLVS and is not is included in a processing order nesting SEI message.

– Otherwise (qmSeiA is included in a processing order nesting SEI message with a certain set of pon\_target\_po\_id[ i ] values), the following applies:

– It is a requirement of bitstream conformance that there shall be quality metric SEI message qmSeiB that is present in the first picture unit of the CLVS and is included in a processing order nesting SEI message with the same set of pon\_target\_po\_id[ i ] values.

– Each value of qm\_clvs\_metric\_value[ i ][ c ] in qmSeiA shall be equal to the value of qm\_clvs\_metric\_value[ i ][ c ] in qmSeiB.

D.12.15 Use of the display overlays SEI message in VVC

For purposes of interpretation of the constituent rectangles SEI message, the following variables are specified:

– PicWidthInLumaSamples[ i ] and PicHeightInLumaSamples[ i ] are set equal to pps\_pic\_width\_in\_luma\_samples and pps\_pic\_height\_in\_luma\_samples, respectively, of the picture with nuh\_layer\_id equal to i.

– ChromaFormatIdc[ i ] is set equal to sps\_chroma\_format\_idc, of the picture with nuh\_layer\_id equal to i.

– NumSubpics[ i ] is set equal to sps\_num\_subpics\_minus1 + 1 of the picture with nuh\_layer\_id equal to i.

– SubPicWidth[ i ][ j ] is set equal to (sps\_subpic\_width\_minus1[ j ] + 1 ) \* CtbSizeY – 1 of the picture with nuh\_layer\_id equal to i.

– SubPicHeight[ i ][ j ] is set equal to (sps\_subpic\_height\_minus1[ j ] + 1 ) \* CtbSizeY – 1 of the picture with nuh\_layer\_id equal to i.

It is a requirement of bitstream conformance that there shall be at least one OLS with index olsIdx that has values of OutputLayerIdInOls[ olsIdx ][ j ] equal to doi\_nuh\_layer\_id[ i ] and doi\_alpha\_nuh\_layer\_id[ i ], when present, for each value of i in the range of 0 to doi\_num\_display\_overlays\_minus2 + 1, inclusive, and for any values of j in the range of 0 to NumOutputLayersInOls[ olsIdx ] − 1, inclusive.

D.12.16 Use of the display overlays SEI message in VVC

For purposes of interpretation of the constituent rectangles SEI message, the following variables are specified:

– ChromaFormatIdc[ i ] is set equal to sps\_chroma\_format\_idc of the picture with nuh\_layer\_id equal to i.

– BitDepth[ i ] is set equal to BitDepth of the picture with nuh\_layer\_id equal to i.

– PicWidthInLumaSamples[ i ] and PicHeightInLumaSamples[ i ] are set equal to pps\_pic\_width\_in\_luma\_samples and pps\_pic\_height\_in\_luma\_samples, respectively, of the picture with nuh\_layer\_id equal to i.

– NumSubpics[ i ] is set equal to sps\_num\_subpics\_minus1 + 1 of the picture with nuh\_layer\_id equal to i.

– SubPicWidth[ i ][ j ] is set equal to (sps\_subpic\_width\_minus1[ j ] + 1 ) \* CtbSizeY – 1 of the picture with nuh\_layer\_id equal to i.

– SubPicHeight[ i ][ j ] is set equal to (sps\_subpic\_height\_minus1[ j ] + 1 ) \* CtbSizeY – 1 of the picture with nuh\_layer\_id equal to i.

*Changes to HEVC for adding support for modality information SEI message*

**D.2 SEI payload syntax**

**D.2.1 General SEI message syntax**

|  |  |
| --- | --- |
| sei\_payload( payloadType, payloadSize ) { | **Descriptor** |
| … |  |
| else if( payloadType = = 210 ) /\* Specified in Rec. ITU-T H.274 | ISO/IEC 23002-7 \*/ |  |
| nn\_post\_filter\_characteristics( payloadSize ) |  |
| else if( payloadType = = 211 ) /\* Specified in Rec. ITU-T H.274 | ISO/IEC 23002-7 \*/ |  |
| nn\_post\_filter\_activation( payloadSize ) |  |
| else if( payloadType = = 212 ) /\* Specified in Rec. ITU-T H.274 | ISO/IEC 23002-7 \*/ |  |
| phase\_indication( payloadSize ) |  |
| … |  |
| else if( payloadType = = 218) |  |
| modality\_info( payloadSize ) |  |
| … |  |
| } |  |

**D.3 SEI payload semantics**

**D3.1 General SEI payload semantics**

The list SingleLayerSeiList is set to consist of the payloadType values 2, 3, 6, 9, 15, 16, 17, 19, 22, 23, 45, 47, 56, 128, 129, 131, 132, 134 to 152, inclusive, 154 to 159, inclusive, 200 to 202, inclusive, 205, 210 to 212, inclusive, and 218.

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, 134 to 152, inclusive, 154 to 159, inclusive, 200 to 202, inclusive, 205, 210 to 212, inclusive, and 218.

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, 135 to 152, inclusive, 154 to 159, inclusive, 200 to 202, inclusive, 205, 210 to 212, inclusive, and 218.

*Changes to AVC for adding support for modality information SEI message*

**D.1 SEI payload syntax**

**D.1.1 General SEI message syntax**

|  |  |  |
| --- | --- | --- |
| sei\_payload( payloadType, payloadSize ) { | **C** | **Descriptor** |
| … |  |  |
| else if( payloadType = = 210 ) |  |  |
| nn\_post\_filter\_characteristics( payloadSize ) | 5 |  |
| else if( payloadType = = 211 ) |  |  |
| nn\_post\_filter\_activation( payloadSize ) | 5 |  |
| else if( payloadType = = 212 ) |  |  |
| phase\_indication( payloadSize ) | 5 |  |
| … |  |  |
| else if( payloadType = = 218 ) |  |  |
| modality\_info( payloadSize ) | 5 |  |
| else |  |  |
| reserved\_sei\_message( payloadSize ) | 5 |  |
| if( !byte\_aligned( ) ) { |  |  |
| **bit\_equal\_to\_one** /\* equal to 1 \*/ | 5 | f(1) |
| while( !byte\_aligned( ) ) |  |  |
| **bit\_equal\_to\_zero** /\* equal to 0 \*/ | 5 | f(1) |
| } |  |  |
| } |  |  |

**G.13 Supplemental enhancement information**

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

**G.13.2 SEI payload semantics**

The semantics of the SEI messages with payloadType in the range of 0 to 23, inclusive, or equal to 45, 47, 137, 142, 144, 147, 148, 149, 150, 151, 154, 155, 156, 200, 201, 202, 205, 210, 211, 212 or 218, which are specified in clause ‎D.2, are extended as follows:

– If payloadType is equal to 3, 8, 19, 20, or 22, the following applies:

– If the SEI message is not included in a scalable nesting SEI message, it applies to the layer representations of the current access unit that have dependency\_id equal to 0 and quality\_id equal to 0.

The semantics as specified in clause ‎D.2 apply to the bitstream that would be obtained by invoking the bitstream extraction process as specified in clause ‎G.8.8.1 with dIdTarget equal to 0 and qIdTarget equal to 0. All syntax elements and derived variables that are referred to in the semantics in clause ‎D.2 are syntax elements and variables for layer representations with dependency\_id equal to 0 and quality\_id equal to 0. All SEI messages that are referred to in clause ‎D.2 are SEI messages that apply to layer representations with dependency\_id equal to 0 and quality\_id equal to 0.

– Otherwise (the SEI message is included in a scalable nesting SEI message), the SEI message applies to all layer representations of the current access unit for which DQId is equal to any value of ( ( sei\_dependency\_id[ i ] << 4 ) + sei\_quality\_id[ i ] ) with i in the range of 0 to num\_layer\_representations\_minus1, inclusive.

For each value of i in the range of 0 to num\_layer\_representations\_minus1, inclusive, the semantics as specified in clause ‎D.2 apply to the bitstream that would be obtained by invoking the bitstream extraction process as specified in clause ‎G.8.8.1 with dIdTarget equal to sei\_dependency\_id[ i ] and qIdTarget equal to sei\_quality\_id[ i ]. All syntax elements and derived variables that are referred to in the semantics in clause ‎D.2 are syntax elements and variables for layer representations with dependency\_id equal to sei\_dependency\_id[ i ] and quality\_id equal to sei\_quality\_id[ i ]. All SEI messages that are referred to in clause ‎D.2 are SEI messages that apply to layer representations with dependency\_id equal to sei\_dependency\_id[ i ] and quality\_id equal to sei\_quality\_id[ i ].

– Otherwise, if payloadType is equal to 2, 6, 7, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 21, 23, 45, 47, 137, 142, 144, 147, 148, 149, 150, 151, 154, 155, 156, 200, 201, 202, 205, 210, 211, 212 or 218, the following applies:

– If the SEI message is not included in a scalable nesting SEI message, it applies to the dependency representations of the current access unit that have dependency\_id equal to 0.

The semantics as specified in clause ‎D.2 apply to the bitstream that would be obtained by invoking the bitstream extraction process as specified in clause ‎G.8.8.1 with dIdTarget equal to 0. All syntax elements and derived variables that are referred to in the semantics in clause ‎D.2 are syntax elements and variables for dependency representations with dependency\_id equal to 0. All SEI messages that are referred to in clause ‎D.2 are SEI messages that apply to dependency representations with dependency\_id equal to 0.

– Otherwise (the SEI message is included in a scalable nesting SEI message), the scalable nesting SEI message containing the SEI message shall have all\_layer\_representations\_in\_au\_flag equal to 1 or, when all\_layer\_representations\_in\_au\_flag is equal to 0, all values of sei\_quality\_id[ i ] present in the scalable nesting SEI message shall be equal to 0. The SEI message that is included in the scalable nesting SEI message applies to all dependency representations of the current access unit for which dependency\_id is equal to any value of sei\_dependency\_id[ i ] with i in the range of 0 to num\_layer\_representations\_minus1, inclusive.

For each value of i in the range of 0 to num\_layer\_representations\_minus1, inclusive, the semantics as specified in clause ‎D.2 apply to the bitstream that would be obtained by invoking the bitstream extraction process as specified in clause ‎G.8.8.1 with dIdTarget equal to sei\_dependency\_id[ i ]. All syntax elements and derived variables that are referred to in the semantics in clause ‎D.2 are syntax elements and variables for dependency representations with dependency\_id equal to sei\_dependency\_id[ i ]. All SEI messages that are referred to in clause ‎D.2 are SEI messages that apply to dependency representations with dependency\_id equal to sei\_dependency\_id[ i ].

When payloadType is equal to 10 for the SEI message that is included in a scalable nesting SEI message, the semantics for sub\_seq\_layer\_num of the sub-sequence information SEI message is modified as follows:

**sub\_seq\_layer\_num** specifies the sub-sequence layer number of the current picture. When the current picture resides in a sub-sequence for which the first picture in decoding order is an IDR picture, the value of sub\_seq\_layer\_num shall be equal to 0. For a non-paired reference field, the value of sub\_seq\_layer\_num shall be equal to 0. sub\_seq\_layer\_num shall be in the range of 0 to 255, inclusive.

– Otherwise, if payloadType is equal to 0 or 1, the following applies:

– If the SEI message is not included in a scalable nesting SEI message, the following applies. When the SEI message and all other SEI messages with payloadType equal to 0 or 1 not included in a scalable nesting SEI message are used as the buffering period and picture timing SEI messages for checking the bitstream conformance according to Annex ‎C and the decoding process specified in clauses ‎2 to ‎9 is used, the bitstream shall be conforming to this Recommendation | International Standard.

The value of seq\_parameter\_set\_id in a buffering period SEI message not included in a scalable nesting SEI message shall be equal to the value of seq\_parameter\_set\_id in the picture parameter set that is referenced by the layer representation with DQId equal to 0 of the primary coded picture in the same access unit.

– Otherwise (the SEI message is included in a scalable nesting SEI message), the following applies. When the SEI message and all other SEI messages with payloadType equal to 0 or 1 included in a scalable nesting SEI message with identical values of sei\_temporal\_id, sei\_dependency\_id[ i ], and sei\_quality\_id[ i ] are used as the buffering period and picture timing SEI messages for checking the bitstream conformance according to Annex ‎C, the bitstream that would be obtained by invoking the bitstream extraction process as specified in clause ‎G.8.8.1 with tIdTarget equal to sei\_temporal\_id, dIdTarget equal to sei\_dependency\_id[ i ], and qIdTarget equal to sei\_quality\_id[ i ] shall be conforming to this Recommendation | International Standard.

In the semantics of clauses ‎D.2.1 and ‎D.2.3, the syntax elements num\_units\_in\_tick, time\_scale, fixed\_frame\_rate\_flag, nal\_hrd\_parameters\_present\_flag, vcl\_hrd\_parameters\_present\_flag, low\_delay\_hrd\_flag, and pic\_struct\_present\_flag and the derived variables NalHrdBpPresentFlag, VclHrdBpPresentFlag, and CpbDpbDelaysPresentFlag are substituted with the syntax elements vui\_ext\_num\_units\_in\_tick[ i ], vui\_ext\_time\_scale[ i ], vui\_ext\_fixed\_frame\_rate\_flag[ i ], vui\_ext\_nal\_hrd\_parameters\_present\_flag[ i ], vui\_ext\_vcl\_hrd\_parameters\_present\_flag[ i ], vui\_ext\_low\_delay\_hrd\_flag[ i ], and vui\_ext\_pic\_struct\_present\_flag[ i ] and the derived variables VuiExtNalHrdBpPresentFlag[ i ], VuiExtVclHrdBpPresentFlag[ i ], and VuiExtCpbDpbDelaysPresentFlag[ i ].

The value of seq\_parameter\_set\_id in a buffering period SEI message included in a scalable nesting SEI message with the values of sei\_dependency\_id[ i ] and sei\_quality\_id[ i ] shall be equal to the value of seq\_parameter\_set\_id in the picture parameter set that is referenced by the layer representation with DQId equal to (( sei\_dependency\_id[ i ] << 4 ) + sei\_quality\_id[ i ]) of the primary coded picture in the same access unit.

– Otherwise (payloadType is equal to 4 or 5), the corresponding SEI message semantics are not extended.

When an SEI message having a particular value of payloadType equal to 137 or 144, contained in a scalable nesting SEI message, and applying to a particular combination of dependency\_id, quality\_id, and temporal\_id is present in an access unit, the SEI message with the particular value of payloadType applying to the particular combination of dependency\_id, quality\_id, and temporal\_id shall be present a scalable nesting SEI message in the IDR access unit that is the first access unit of the coded video sequence.

All SEI messages having a particular value of payloadType equal to 137 or 144, contained in scalable nesting SEI messages, and applying to a particular combination of dependency\_id, quality\_id, and temporal\_id present in a coded video sequence shall have the same content.

For the semantics of SEI messages with payloadType in the range of 0 to 23, inclusive, or equal to 45, 47, 137, 142, 144, 147, 148, 149, 150, 151, 154, 155, 156, 200, 201, 202, 205, 210, 211, 212 or 218, which are specified in clause ‎D.2, SVC sequence parameter set is substituted for sequence parameter set; the parameters of the picture parameter set RBSP and SVC sequence parameter set RBSP that are in effect are specified in clause ‎G.7.4.1.2.1.

Coded video sequences conforming to one or more of the profiles specified in Annex ‎G shall not include SEI NAL units that contain SEI messages with payloadType in the range of 36 to 44, inclusive, or equal to 46, which are specified in clause ‎H.13, or with payloadType in the range of 48 to 53, inclusive, which are specified in clause ‎I.13.

When an SEI NAL unit contains an SEI message with payloadType in the range of 24 to 35, inclusive, which are specified in clause ‎G.13, it shall not contain any SEI message that has payloadType less than 24 or equal to 45, 47, 137, 142, 144, 147, 148, 149, 150, 151, 154, 155, 156, 200, 201, 202, 205, 210, 211, 212 or 218 that is not included in a scalable nesting SEI message, and the first SEI message in the SEI NAL unit shall have payloadType in the range of 24 to 35, inclusive.

When an SEI NAL unit contains an SEI message with payloadType equal to 24, 28, or 29, it shall not contain any SEI message with payloadType not equal to 24, 28, or 29.

When a scalable nesting SEI message (payloadType is equal to 30) is present in an SEI NAL unit, it shall be the only SEI message in the SEI NAL unit.

The semantics for SEI messages with payloadType in the range of 24 to 35, inclusive, are specified in the following.

*Addition(s) related to large SEI message and versatile SEI RBSP / versatile SEI message*

*[Ed: Proponents of both approaches are asked to work out possible combination]*

X.1 Image format metadata SEI messages

X.1.1 Large supplemental enhancement information message syntax

|  |  |
| --- | --- |
| lsei\_message( ) { | **Descriptor** |
| **lsei\_position** | u(2) |
| **lsei\_relevance** | u(2) |
| **lsei\_reserved** | u(4) |
| **lsei\_payload\_type\_byte** | u(8) |
| **lsei\_payload\_size\_16bits** | u(16) |
| lsei\_payload( lseiPayloadType, lseiPayloadSize ) |  |
| } |  |

X.1.2 Large supplemental enhancement information message semantics

Each large SEI message consists of the variables specifying the type payloadType and size payloadSize of the large SEI message payload. Large SEI message payloads are specified in Annex D. The derived large SEI message payload size payloadSize is specified in bytes and shall be equal to the number of RBSP bytes in the large SEI message payload.

NOTE – The NAL unit byte sequence containing the large SEI message might include one or more emulation prevention bytes (represented by emulation\_prevention\_three\_byte syntax elements). Since the payload size of a large SEI message is specified in RBSP bytes, the quantity of emulation prevention bytes is not included in the size payloadSize of a large SEI payload.

**lsei\_position** indicates if the SEI message corresponds to the PREFIX\_SEI\_NUT and SUFFIX\_SEI\_NUT. **lsei\_position** equal 0 indicates that the SEI message is treated as PREFIX\_SEI\_NUT. **lsei\_position** equal 1 indicates that the SEI message is treated as SUFFIX\_SEI\_NUT. Values 3 and 4 of **lsei\_position** are reserved for future use and shall be ignored.

**lsei\_relevance** indicates the relevance of the SEI message for the target application. **lsei\_relevance** ranges from 0 to 3, with 0 being the least relevant and 3 being the most relevant.

NOTE – The relevance of an SEI message is an arbitrary decision and its use is to be specified by the target application.

**lsei\_reserved** is reserved for future use and shall be ignored.

**lsei\_payload\_type\_byte** is a byte of the payload type of a large SEI message. payloadType = **lsei\_payload\_type\_byte.**

**payload\_size\_16bits** is the payload size in bits of a large SEI message. payloadSize = **payload\_size\_16bits.**

X.2 Versatile SEI RBSP

X.2.1 Versatile SEI RBSP syntax

|  |  |
| --- | --- |
| vsei\_rbsp( ) { | Descriptor |
| **vsei\_importance** | u(1) |
| **vsei\_reserved** | u(7) |
| versatile\_sei\_message( ) |  |
| rbsp\_trailing\_bits( ) |  |
| } |  |

X.2.2 Versatile SEI RBSP semantics

Versatile Supplemental enhancement information RBSP contains information that is not necessary to decode the samples of coded pictures from VCL NAL units. A VSEI RBSP contains one VSEI message.

**vsei\_importance** equal 1 indicates that the versatile SEI message may be an important or required. Vsei\_importance equal 0 indicates that the versatile SEI message does not have particular importance.

NOTE – Entity that is aware of this flag may use this information when it needs to make decision whether or not to deliver / drop the verstatile SEI message.

**vsei\_reserved** is reserved for future use and shall be ignored.

X.2.3 Versatile SEI message syntax

|  |  |
| --- | --- |
| versatile\_sei\_message( ) { | **Descriptor** |
| **vsei\_payload\_type\_byte** | u(8) |
| vsei\_payload( vseiPayloadType ) |  |
| } |  |

X.2.4 Versatile SEI message semantics

Versatile Supplemental enhancement information RBSP contains information that is not necessary to decode the samples of coded pictures from VCL NAL units. A VSEI RBSP contains one VSEI message.

Each versatile SEI message consists of the variables specifying the importance and type payloadType of the SEI message payload. Versatile SEI message payloads are specified in XXX.

NOTE – The NAL unit byte sequence containing the SEI message might include one or more emulation prevention bytes (represented by emulation\_prevention\_three\_byte syntax elements).

**vsei\_payload\_type\_byte** is a byte of the payload type of a versatile SEI message.

*Addition(s) related to alternative persistence signalling in SEI messages*

X.1 Example SEI message to describe alternative persistence signalling for SEI messages

X.1.1 Example SEI message without an ID syntax

|  |  |
| --- | --- |
| example( payloadSize ) { | **Descriptor** |
| **ex\_persistence\_idc** | u(2) |
| if( ex\_persistence\_idc > 0 ) { |  |
| /\* additional syntax elements \*/ |  |
| } |  |
| } |  |

X.1.2 Example SEI message without an ID semantics

**ex\_persistence\_idc** specifies the persistence of the example SEI message for the current layer.

ex\_persistence\_idc equal to 0 indicates that the example SEI message cancels the persistence of any previous example SEI message in output order that applies to the current layer of the current example SEI message.

ex\_persistence\_idc equal to 1 specifies that the example SEI message applies to the current decoded picture only.

ex\_persistence\_idc equal to 2 specifies that the example SEI message applies to the current decoded picture and persists for all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture in the current layer in an AU associated with an example SEI message that follows the current picture in output order.

ex\_persistence\_idc equal to 3 specifies that the example SEI message applies for the entire CLVS.

It is a requirement of bitstream conformance that when ex\_persistence\_idc is equal to 3 and this SEI message is not the first example SEI message in a CLVS, in decoding order, an example SEI message shall be present in the first PU of the CLVS in decoding order.

When ex\_persistence\_idc is equal to 3 in the first PU in the CLVS in decoding order, all example SEI messages in the CLVS shall have identical SEI payload content.

X.1.1.3 Example SEI message with an ID syntax

|  |  |
| --- | --- |
| example( payloadSize ) { | **Descriptor** |
| **ex\_id** | u(6) |
| **ex\_persistence\_idc** | u(2) |
| if( ex\_persistence\_idc > 0 ) { |  |
| /\* additional syntax elements \*/ |  |
| } |  |
| } |  |

X.1.4 Example SEI message with an ID semantics

**ex\_id** specifies an identifier of the example SEI message.

**ex\_persistence\_idc** specifies the persistence of the example SEI message for the current layer.

ex\_persistence\_idc equal to 0 indicates that the example SEI message cancels the persistence of any previous example SEI message in output order that applies to the current layer and has the same ex\_id as the current example SEI message.

ex\_persistence\_idc equal to 1 specifies that the example SEI message applies to the current decoded picture only.

ex\_persistence\_idc equal to 2 specifies that the example SEI message applies to the current decoded picture and persists for all subsequent pictures of the current layer in output order until one or more of the following conditions are true:

– A new CLVS of the current layer begins.

– The bitstream ends.

– A picture in the current layer in an AU associated with an example SEI message with the same ex\_id is output that follows the current picture in output order.

ex\_persistence\_idc equal to 3 specifies that the example SEI message applies for the entire CLVS.

It is a requirement of bitstream conformance that when ex\_persistence\_idc is equal to 3 and this SEI message is not the first example SEI message in a CLVS, in decoding order, an example SEI message with the same value of ex\_id shall be present in the first PU of the CLVS in decoding order.

All example SEI messages in a CLVS that have a particular ex\_id value and ex\_persistence\_idc equal to 3 shall have identical SEI payload content.

When ex\_persistence\_idc is equal to 3 in the first PU in the CLVS in decoding order, all example SEI messages in the CLVS that have a particular ex\_id value shall have identical SEI payload content.